Profitina: Membangun Platform Visibilitas AI secara Terbuka

Irfan Subakti

Profitina
profitina.com

Profitina

Membangun Platform Visibilitas AI secara Terbuka

Irfan Subakti

Hak cipta © 2026 Irfan Subakti. Hak cipta dilindungi undang-undang.

Diterbitkan oleh Profitina · profitina.com

Edisi pertama, September 2026.

Setiap program dalam buku ini sudah dikompilasi dan dijalankan sebelum keluarannya dicetak. Kode sumber lengkapnya ada di folder Code yang menyertai buku ini. Program-program tersebut disediakan untuk keperluan belajar-mengajar; Irfan Subakti dan penerbit tidak bertanggung jawab atas penggunaannya di luar keperluan itu.

Daftar Isi

Kata Pengantar ..... **xxvii**
Untuk siapa buku ini ditulis xxvii
Isi buku ini xxvii
Cara membacanya xxviii
Tentang penulis xxviii
Ucapan terima kasih xxviii

BAB 1 Asal-Usul: Dari Ruang Kelas Seorang Dosen ke Sebuah Platform **1**
1.1 Sebuah Platform dengan Ruang Kelas di Belakangnya 1
1.2 Pendiri dan Tiga Dasawarsa Mengajar 2
1.2.1 Seorang dosen di Surabaya 2
1.2.2 Tempat pengajaran bermula 2
1.2.3 Riset di beberapa negara dan beberapa cabang AI 3
1.3 Ruang Kelas dan Basis Kode Adalah Tempat yang Sama 3
1.3.1 Tiga matakuliah yang menjadi bahan sebuah platform 3
1.3.2 Sebuah pustaka buku ajar di sekitar satu sistem nyata 4
1.3.3 Platform di sebuah kelas nyata 4
1.4 Dari Gagasan ke Produksi: Lini Masa Bertanggal 5
1.4.1 Sebelum Juli 2026 5
1.4.2 Dua belas pekan, pekan demi pekan 5
1.4.3 Menghitung hari: pandangan pertama pada kode 7
1.5 Gagasan-Gagasan Pendiri 8
1.5.1 Misi 8
1.5.2 Visibilitas dan partisipasi dalam satu sistem 8
1.5.3 AI lokal, karena pilihan 9
1.5.4 Diuji sebelum dirilis 9
1.5.5 Indonesia lebih dahulu 9
1.6 Untuk Siapa Buku Ini 10
1.7 Cara Membaca Buku Ini 10
1.7.1 Bentuk buku 10
1.7.2 Setiap bab memiliki bagian yang sama 11
1.7.3 Program, keluaran, dan folder kode 11
1.7.4 Kotak, label, dan tanggal 12
1.8 Menyiapkan VS Code untuk Python dan Java di Windows, macOS, dan Linux 12
1.8.1 Tiga istilah lebih dahulu 13
1.8.2 Visual Studio Code 13
1.8.3 Java: JDK 25 13
1.8.4 Python 3.14 14
1.8.5 Dua ekstensi VS Code 14
1.8.6 Sebuah folder untuk buku ini 15
1.8.7 Memeriksa seluruh meja kerja 15
1.8.8 Program pertama Anda, dalam dua bahasa 18
1.8.9 Program kedua: hari sejak peluncuran 20
1.8.10 Menjalankan dan men-*debug* di VS Code 23
1.9 Profitina dalam Praktik 24
1.10 Kesalahpahaman yang Sering Terjadi 24
1.11 Ringkasan Bab 25
1.12 Soal Latihan 26
1.13 Penyelesaian Soal Latihan 27
Daftar Pustaka 30

BAB 2 Dari Pencarian ke Jawaban: Bangkitnya GEO 31
2.1 Posisi Kita Sejauh Ini 31
2.2 Dua Puluh Lima Tahun Daftar Tautan 31
2.2.1 Merayapi, mengindeks, memeringkat 31
2.2.2 Manusia yang memilih 32
2.3 Datangnya Mesin Jawaban 32
2.3.1 Satu jawaban, bukan sepuluh tautan 32
2.3.2 Apa yang berubah bagi sebuah usaha 33
2.4 Angka-Angka, 2025–2026 34
2.4.1 Lebih sedikit klik ketika jawaban muncul 35
2.4.2 Lebih banyak jawaban, lebih sering 35
2.4.3 Skala mesin-mesin itu 36
2.4.4 Indonesia 36
2.4.5 Perlakukan ramalan dengan hati-hati 36
2.5 Apa Itu GEO: Asal-Usul Akademiknya 36
2.5.1 Makalah yang menamainya 36
2.5.2 Mengukur visibilitas di dalam jawaban 37
2.5.3 Sebuah tolok ukur dan sembilan metode 37
2.6 Definisi GEO Menurut Profitina 38
2.6.1 Dalam satu kalimat, dan dalam sebuah perbandingan 38
2.6.2 Apa yang diukur Profitina, dan apa yang tidak 39
2.6.3 Siklusnya: Pindai, Optimasi, Pantau 39
2.6.4 Harapan yang jujur 40
2.7 Lapisan yang Dapat Dibaca Mesin: Data Terstruktur, robots.txt, dan llms.txt 40
2.7.1 Data terstruktur: schema.org dalam JSON-LD 40
2.7.2 Jagalah kebenaran fakta 41
2.7.3 robots.txt: siapa boleh membaca apa 41
2.7.4 Crawler-crawler AI 42
2.7.5 llms.txt: peta ringkas bagi model bahasa 43
2.8 Lab G6: Generator JSON-LD, robots.txt, dan llms.txt 44
2.8.1 Apa yang akan kita bangun 44
2.8.2 Berkas fakta 44
2.8.3 Generator dalam Python 45
2.8.4 Tiga berkas yang dibangkitkan 49
2.8.5 Generator dalam Java 52
2.8.6 Membacanya kembali: pemeriksa 62
2.8.7 Ketika ada yang salah 72
2.9 Profitina dalam Praktik 73
2.10 Kesalahpahaman yang Sering Terjadi 74
2.11 Ringkasan Bab 75
2.12 Soal Latihan 76
2.13 Penyelesaian Soal Latihan 77
Daftar Pustaka 81

BAB 3 Apa yang Dikerjakan Profitina: Siklus Visibilitas dan Perjalanan Pengguna 83
3.1 Posisi Kita Sekarang 83
3.2 Siklus Visibilitas 84
3.2.1 Empat kata kerja, tiga tab 84
3.2.2 Mengapa siklus, bukan garis lurus 85
3.2.3 Apa yang diukur, dan apa yang tidak 86
3.3 Ukur: Apa yang Diperiksa Sebuah Pindaian 86
3.3.1 Masukan 86
3.3.2 Tiga keluarga sinyal 87
3.3.3 Kemajuan yang dapat dibaca 88
3.3.4 Angka itu, dan maknanya 88

3.4 Jelaskan: Dari Sebuah Angka ke Daftar Perbaikan 88
3.5 Perbaiki: Mesin Konten dan Pendamping Teknis 89
3.5.1 Apa yang disusun Mesin Konten 89
3.5.2 Draf AI hanya bila lebih baik 89
3.5.3 Pendamping Teknis 90
3.6 Pantau: Riwayat, Jadwal, dan Peringatan 90
3.6.1 Tren mingguan 90
3.6.2 Apa arti "berarti" 90
3.7 Melampaui Situs Web: Laporan Peneliti dan Organisasi 91
3.7.1 Laporan Visibilitas Peneliti 91
3.7.2 Visibilitas Organisasi 91
3.7.3 Academic Citation Scan 92
3.8 Satu Teks, Banyak Platform 92
3.9 Perjalanan Pengguna: Profitina Laundry, Langkah demi Langkah 92
3.9.1 Langkah 1: Datang 93
3.9.2 Langkah 2: Mencoba tanpa akun 93
3.9.3 Langkah 3: Mendaftar 93
3.9.4 Langkah 4: Pindai 94
3.9.5 Langkah 5: Optimasi 95
3.9.6 Langkah 6: Pantau 96
3.9.7 Langkah 7: Berpartisipasi 97
3.9.8 Langkah 8: Meningkatkan paket, bila ia mau 97
3.9.9 Langkah 9: Menggunakan haknya 98
3.9.10 Perjalanan yang sama bagi seorang peneliti 99
3.9.11 Satu Teks, Banyak Platform, dari dalam 99
3.10 Lab G1: Pemeriksa Kesiapan AI Mini 100
3.10.1 Apa yang akan kita bangun, dan milik siapa skornya 100
3.10.2 Situs-situs contoh 101
3.10.3 Peladen lokal yang mungil 106
3.10.4 Membaca robots.txt menurut aturan 109
3.10.5 Mengumpulkan fakta sebuah halaman 115
3.10.6 Delapan pemeriksaan 122
3.10.7 Laporan dan baris perintah 131
3.10.8 Menjalankan lab: sebelum dan sesudah 137
3.10.9 Memilih bobot sendiri 142
3.10.10 Ketika pemeriksa berhenti 143
3.10.11 Apa yang tidak dapat dilihat pemeriksa 143
3.11 Profitina dalam Praktik 143
3.12 Kesalahpahaman yang Umum 144
3.13 Ringkasan Bab 145
3.14 Soal Latihan 146
3.15 Penyelesaian Soal Latihan 147
Daftar Pustaka 153

BAB 4 Komunitas yang Dirancang 155
4.1 Posisi Kita Sekarang 155
4.2 Mengapa Komunitas Dibangun di dalam Alat Visibilitas 156
4.2.1 Orang sudah hidup di dalam umpan 156
4.2.2 Apa yang dilakukan Profitina secara berbeda 156
4.3 Peta Komunitas 157
4.4 Waktu-Nyata Tanpa Sihir: Polling, WebSocket, dan Server-Sent Events 158
4.4.1 Masalahnya: web dibangun untuk bertanya 158
4.4.2 Polling: bertanya lagi dan lagi 159
4.4.3 WebSocket: pipa dua arah 160
4.4.4 Server-Sent Events: satu jawaban yang panjang 160

4.4.5 Peristiwa tipis 162
4.5 Umpan, Utas, dan Repost-Kutip 162
4.5.1 Umpan adalah daftar; utas adalah pohon 162
4.5.2 Repost dan tempatnya 162
4.5.3 Jendela sunting lima belas menit 163
4.6 Kanal dan Gerbang Saldo PROFIT 163
4.6.1 Ruang dengan tujuan 163
4.6.2 Gerbangnya 163
4.6.3 Papan peringkat per kanal 164
4.7 Mengikuti Kanal dan Penghitung "N baru" 165
4.7.1 Apa yang ingin diketahui pengikut 165
4.7.2 Definisinya 165
4.7.3 Satu kata, satu angka 165
4.7.4 Menjaga penghitung tetap hidup 166
4.8 Pesan Langsung dan Profil: Privat secara Bawaan 166
4.8.1 Pesan langsung 166
4.8.2 Profil dengan persetujuan per butir 166
4.8.3 Handle netral 167
4.9 Dasbor sebagai Kokpit 167
4.10 Pengiriman Seketika (Optimistis) 168
4.10.1 Menunggu terasa lebih lama daripada sebenarnya 168
4.10.2 Tampilkan dulu, pastikan kemudian 169
4.11 Ketika Waktu-Nyata Keliru: Tiga Pelajaran Rekayasa 169
4.11.1 Pelajaran 1: jalur baca tidak boleh memanggil model 169
4.11.2 Pelajaran 2: tandai basi, jangan ambil ulang semuanya 170
4.11.3 Pelajaran 3: hitung kueri Anda 170
4.11.4 Pelajaran 4 (bonus): jawaban yang terlambat 171
4.12 Lab G2: Umpan Langsung dengan Server-Sent Events 172
4.12.1 Apa yang akan kita bangun 172
4.12.2 Server Python 172
4.12.3 Server Java 177
4.12.4 Halaman peramban 186
4.12.5 Klien konsol 191
4.12.6 Menjalankannya 196
4.12.7 Memeriksa halaman di peramban tanpa layar 197
4.12.8 Apa yang tidak dicakup lab 198
4.13 Profitina dalam Praktik 198
4.14 Kesalahpahaman yang Umum 199
4.15 Ringkasan Bab 200
4.16 Soal Latihan 200
4.17 Penyelesaian Soal Latihan 201
Daftar Pustaka 203

BAB 5 Menjaga Komunitas Tetap Aman: Link Safety, Moderasi, Main Adil 205
5.1 Posisi Kita Sekarang 205
5.2 Mengapa Komunitas Perlu Dilindungi 206
5.2.1 Tiga angka 206
5.2.2 Tiga ancaman, tiga pertahanan 206
5.3 Anatomi Tautan, dan Tempat Trik Bersembunyi 207
5.3.1 Bagian-bagian URL 207
5.3.2 Trik-trik yang biasa 208
5.3.3 Sinyal keras dan sinyal lunak 209
5.3.4 Lima detik untuk memeriksa 209
5.4 Keamanan Tautan sebagai Lapisan 210
5.4.1 Dua lapisan 210

5.4.2 Mengapa lantai lebih dulu ... 210
5.4.3 Tidak pernah di jalur baca ... 211
5.4.4 Lapisan kedua yang lebih senyap: tampilkan teks sebagai teks ... 211
5.5 Mengukur Penyaring: Matriks Kebingungan ... 211
5.5.1 Empat jenis hasil ... 211
5.5.2 Lima ukuran dari empat hitungan ... 212
5.5.3 Tolok ukur publik Profitina ... 213
5.5.4 Apa yang dibuktikan nol dari tiga puluh ... 213
5.5.5 Mengapa laju positif palsu lebih penting daripada kelihatannya ... 214
5.6 Moderator: Pertimbangan Manusia, Wewenang Sempit ... 215
5.6.1 Yang tidak dapat dilakukan perangkat lunak ... 215
5.6.2 Moderator bercakupan kanal ... 215
5.6.3 Pelaporan, dan salinan beku ... 216
5.6.4 Aturan yang dapat dimoderasi ... 216
5.7 Main Adil: Menjaga Hadiah untuk Manusia ... 217
5.7.1 Hadiah mengundang otomatisasi ... 217
5.7.2 Lapisan-lapisannya ... 217
5.7.3 Mengapa lapisan saling mengalikan: ilustrasi terhitung ... 218
5.7.4 Yang sengaja tidak dipublikasikan ... 219
5.8 Lab G3: Penyaring Tautan Berbasis Aturan Milik Kita ... 219
5.8.1 Apa yang akan kita bangun ... 219
5.8.2 Aturan keputusan ... 220
5.8.3 Seberapa jauh “beda satu huruf”? ... 220
5.8.4 Penyaring Python ... 221
5.8.5 Penyaring Java ... 226
5.8.6 Mencobanya pada pesan tunggal ... 230
5.8.7 Himpunan sampel dan evaluasinya ... 231
5.8.8 Membaca hasilnya ... 238
5.8.9 Pandangan lain: titik-titik pada plot ROC ... 239
5.8.10 Apa yang tidak dicakup lab ... 240
5.9 Profitina dalam Praktik ... 240
5.10 Kesalahpahaman yang Umum ... 240
5.11 Ringkasan Bab ... 241
5.12 Soal Latihan ... 242
5.13 Penyelesaian Soal Latihan ... 243
Daftar Pustaka ... 245

BAB 6 AI Lokal: Mesin Konten dan Model Dua Jalur ... 246
6.1 Titik Berangkat Kita ... 246
6.2 Apa yang Dikerjakan Model Bahasa, dengan Kata-Kata Sederhana ... 247
6.2.1 Menebak potongan teks berikutnya ... 247
6.2.2 Parameter, memori, dan kuantisasi ... 248
6.3 Mengapa Hanya Inferensi Lokal: Privasi, Biaya, Kendali ... 248
6.3.1 Privasi ... 248
6.3.2 Biaya ... 249
6.3.3 Kendali ... 250
6.3.4 Biaya jujur dari pilihan lokal ... 250
6.4 Ollama dan Keluarga Qwen ... 251
6.4.1 Ollama: server model di mesin Anda sendiri ... 251
6.4.2 Keluarga Qwen, per September 2026 ... 251
6.4.3 Mengapa Profitina tetap menjalankan Qwen2.5 7B ... 252
6.5 Model Dua Jalur ... 253
6.5.1 Dua jenis pekerjaan ... 253
6.5.2 Mengapa panjang itu penting: sedikit aritmetika ... 254
6.5.3 Satu pekerjaan panjang sekaligus, dan antrean yang jujur ... 254

6.5.4 Pekerjaan yang tidak masuk jalur mana pun 255
6.6 Content Engine dan Gerbang Kualitas 255
6.6.1 Apa yang dihasilkan Content Engine 255
6.6.2 Gerbang: sajikan AI hanya ketika lebih baik 255
6.6.3 Membaca hasil publik: 5 dari 6 257
6.7 Perilaku Jujur Saat "Tidak Ada Model yang Tersedia" 257
6.8 Lab G5: Berbicara dengan Model Lokal melalui HTTP 258
6.8.1 Apa yang akan kita bangun, dan dari mana keluarannya berasal 258
6.8.2 *Stub* yang meniru `/api/generate` 259
6.8.3 Menjalankan *stub* 271
6.8.4 Mengintip kabelnya 271
6.8.5 Klien 272
6.8.6 Dua jalur dalam kode 280
6.8.7 Gerbang kualitas dalam kode 287
6.8.8 Kode yang sama terhadap Ollama sungguhan 293
6.9 Profitina dalam Praktik 293
6.10 Kesalahpahaman yang Umum 294
6.11 Ringkasan Bab 295
6.12 Soal Latihan 296
6.13 Penyelesaian Soal Latihan 297
Daftar Pustaka 301

BAB 7 Siklus Umpan Balik: Manusia Memutuskan, Model Belajar 302
7.1 Titik Berangkat Kita 302
7.2 Mengapa Umpan Balik, dan Mengapa Manusia yang Memutuskan 303
7.2.1 Model belajar dari manusia 303
7.2.2 Mengapa manusia harus memutuskan 303
7.2.3 Aplikasi memberi saran, pengguna yang memutuskan 304
7.2.4 Sebuah kisah dalam lima langkah 304
7.3 *Pipeline* Lima Langkah 305
7.4 Perjalanan Satu Butir Umpan Balik 306
7.5 Kurasi: Mesin Memilah, Manusia Memutuskan 307
7.5.1 Mengapa memilah sebelum manusia melihat 307
7.5.2 Menemukan duplikat dengan kesamaan kata 307
7.5.3 Keputusan manusia 308
7.5.4 Mengurutkan antrean 308
7.6 Dataset: Pasangan Preferensi dalam JSON Lines 308
7.6.1 Seperti apa contoh pelatihan itu 308
7.6.2 JSON Lines 308
7.6.3 Apa yang tidak boleh masuk dataset 309
7.6.4 Versi dan asal-usul 309
7.7 Latih dan Ukur: Sebelum dan Sesudah 310
7.7.1 Himpunan uji terpisah 310
7.7.2 Tingkat kelolosan dan regresi 310
7.7.3 Mengukur dengan jujur 311
7.7.4 Seberapa presisi sebuah tingkat kelolosan? 311
7.8 Rilis: *Pipeline* Memberi Saran, Manusia Memutuskan 312
7.9 PROFIT untuk Memperbaiki Jawaban AI 313
7.9.1 Mengapa memberi hadiah untuk umpan balik 313
7.9.2 Aturan perolehan yang publik 313
7.9.3 Aritmetika hadiah yang dibatasi 314
7.9.4 Membatasi apa yang dapat diperoleh satu akun 315
7.10 Lab: *Pipeline* Umpan Balik Mini 316
7.10.1 Skenario dan data 316
7.10.2 Programnya 317

7.10.3 Menjalankan *pipeline* 329
7.10.4 Hari apa sekarang? UTC lawan WIB 331
7.11 Profitina dalam Praktik 333
7.12 Kesalahpahaman yang Umum 333
7.13 Ringkasan Bab 334
7.14 Soal Latihan 335
7.15 Penyelesaian Soal Latihan 336
Daftar Pustaka 339

BAB 8 Model Bisnis 340
8.1 Titik Berangkat Kita 340
8.2 Apa Itu Model Bisnis 341
8.2.1 Freemium dalam satu gambaran 341
8.2.2 Apa yang sengaja tidak dijual Profitina 342
8.3 Pasar yang Dilayani Profitina 342
8.3.1 Siapa yang daring 343
8.3.2 Bagaimana orang membayar: QRIS 343
8.3.3 Pembanding harga bagi usaha kecil 344
8.4 Dua Tingkat: Free dan Pro 344
8.4.1 Apa isi Free, dan mengapa begitu banyak 344
8.4.2 Apa tambahan Pro 345
8.4.3 Harga per hari 345
8.4.4 Aturan untuk kartu paket 345
8.5 Mengapa Tingkat Agency Dihapus 346
8.5.1 Biaya tersembunyi sebuah tingkat 346
8.5.2 Kesederhanaan bagi pelanggan 347
8.5.3 Bagaimana dengan agensi sekarang 347
8.6 Tanpa Iklan, Selamanya 347
8.6.1 Konflik kepentingan 347
8.6.2 Argumen privasi 348
8.6.3 Argumen kredibilitas 348
8.7 Membayar Pro: QRIS dan PROFIT 348
8.7.1 QRIS, langkah demi langkah 348
8.7.2 Menukarkan PROFIT 349
8.7.3 Perpanjangan otomatis dan pengingat 349
8.8 Ukur Minat sebelum Membangun Pembayaran 350
8.8.1 Temuan nol pembelian 350
8.8.2 Mengapa aturan ini benar 350
8.8.3 Tangga bukti 350
8.8.4 Aturan yang sama di tempat lain 351
8.9 Kuota Harian dan Aturan 00.00 WIB 351
8.9.1 Hari milik siapa? 352
8.9.2 Jebakan tanggal UTC 352
8.9.3 Hari sebagai rumus 352
8.9.4 Berapa pindaian yang tersisa 353
8.10 Praktikum: Paket sebagai Data dan Kalkulator Kuota 353
8.10.1 Rancangan 354
8.10.2 Program Python 354
8.10.3 Program Java 359
8.11 Praktikum: Langganan sebagai State Machine 363
8.11.1 Keadaan, peristiwa, dan transisi 363
8.11.2 Program Python 364
8.11.3 Program Java 369
8.12 Praktikum: Uji yang Mengunci Aturan 374
8.12.1 Apa yang diuji 374

8.12.2 Uji Python 375
8.12.3 Uji yang menangkap bug sungguhan 377
8.12.4 Uji Java, dengan pelari mungil 378
8.13 Profitina dalam Praktik 383
8.14 Kesalahpahaman yang Sering Terjadi 383
8.15 Ringkasan Bab 384
8.16 Soal Latihan 385
8.17 Penyelesaian Soal Latihan 386
Daftar Pustaka 390

BAB 9 Arsitektur, Keandalan, dan Gerbang Rilis 391
9.1 Titik Berangkat Kita 391
9.2 Keandalan dengan Kata-Kata Sederhana 392
9.2.1 Ketersediaan 392
9.2.2 Arti "sembilan" dalam menit 392
9.2.3 Keandalan mencakup kebenaran 393
9.3 Arsitektur pada Tingkat Kotak 393
9.3.1 Dua pintu depan 394
9.3.2 Tepi dan terowongan 394
9.3.3 Satu host, dijaga hidup oleh NSSM 395
9.3.4 Aplikasinya: FastAPI di bawah Uvicorn 395
9.3.5 Waktu nyata: Server-Sent Events 395
9.3.6 Basis data: PostgreSQL, dengan pemisahan yang ditegakkan 396
9.3.7 Jalur AI: lokal saja 396
9.3.8 Apa yang sengaja tidak ada dalam gambar 396
9.4 Mengapa Satu Mesin Cukup, untuk Saat Ini 397
9.5 Suite Uji sebagai Gerbang Rilis 398
9.5.1 Ukuran suite 398
9.5.2 Mengapa gerbang, dan mengapa sekarang 398
9.5.3 Dua pelajaran dari suite 399
9.6 Rilis Bertahap yang Diverifikasi Kesehatannya 399
9.6.1 Gerbang yang berbohong 399
9.6.2 Perbaikannya: berhasil berarti layanan sendiri yang mengatakannya 400
9.6.3 Enam langkah 400
9.6.4 Apa yang dilaporkan endpoint kesehatan 401
9.7 Kisah Kinerja: Berpindah Kanal 401
9.7.1 Gejalanya 401
9.7.2 Tiga akar penyebab 402
9.7.3 Perbaikan dan hasilnya 402
9.7.4 Aturannya, dan balapannya 403
9.8 Menjadwalkan Pindaian Ulang: Priority Queue 404
9.8.1 Masalahnya 404
9.8.2 Irama dan banyaknya pemindaian yang diharapkan 404
9.8.3 Rincian yang membedakan mainan dari alat 405
9.9 Praktikum G4, Bagian 1: Penjadwal 405
9.9.1 Program Python 405
9.9.2 Program Java 409
9.9.3 Uji-ujinya 413
9.10 Praktikum G4, Bagian 2: Endpoint Kesehatan 417
9.10.1 Program Python 417
9.10.2 Program Java 420
9.11 Praktikum G4, Bagian 3: Gerbang Rilis 423
9.11.1 Bagaimana praktikum memetakan rilis sungguhan 423
9.11.2 Gerbang Python 424
9.11.3 Gerbang Java 429

9.12 Profitina dalam Praktik 436
9.13 Kesalahpahaman yang Sering Terjadi 436
9.14 Ringkasan Bab 437
9.15 Soal Latihan 437
9.16 Penyelesaian Soal Latihan 438
Daftar Pustaka 443

BAB 10 Situs Pemasaran sebagai Studi Kasus GEO 444
10.1 Titik Berangkat Kita 444
10.2 Dua Permukaan, Satu Merek 445
10.2.1 Mengapa perlu situs pemasaran tersendiri 445
10.2.2 Mengapa WordPress 445
10.2.3 Tumpukan teknologi, dari pengunjung ke dalam 446
10.2.4 Apa berada di mana 447
10.3 Tema Dwibahasa yang Dikerjakan dengan Benar 447
10.3.1 Dua bahasa, dua himpunan halaman 447
10.3.2 Kutu `$id`: pelajaran tentang nama bersama 448
10.4 robots.txt dari Prinsip Dasar 449
10.4.1 Apa yang sudah ditetapkan Bab 2 449
10.4.2 Dari baris ke grup 449
10.4.3 Memilih grup: penggabungan dan jalur cadangan 450
10.4.4 Memilih aturan: kecocokan terpanjang, `Allow` bila seri 450
10.4.5 Karakter pengganti 451
10.4.6 Ketika berkas tidak dapat dibaca 451
10.4.7 Membaca dan akses adalah dua hal yang berbeda 452
10.5 Pelajaran robots.txt: Ketika Berkas yang Tayang Bukan Berkas Anda 452
10.5.1 Apa yang terjadi 452
10.5.2 Mengapa penyisipan di depan sudah cukup untuk memblokir 452
10.5.3 Bawaan pun bergeser 453
10.5.4 Berkas yang disajikan profitina.com hari ini 454
10.5.5 Pelajaran umumnya 454
10.6 llms.txt: Apa Itu, dan Apa yang Bukan 455
10.6.1 Usulannya 455
10.6.2 Adopsi dan pemakaian 455
10.6.3 Sikap Profitina: kebersihan murah, tanpa janji 456
10.7 Dua Lapisan Cache dan Urutan Purge 456
10.7.1 Mengapa *cache* ada 456
10.7.2 Kesegaran dalam HTTP 457
10.7.3 Apa yang di-*cache* setiap lapisan secara bawaan 458
10.7.4 Urutan *purge* 458
10.7.5 Memilih masa hidup 459
10.8 Data Terstruktur di Situs Pemasaran 460
10.8.1 Apa yang dibawa halaman-halamannya 460
10.8.2 Setelah Mei 2026: *markup* FAQ tanpa hasil FAQ 460
10.8.3 Kesegaran termasuk bagian data 460
10.8.4 Tiga kesalahan yang membuat data terstruktur tak berguna 461
10.9 Lab: Auditor Kebersihan GEO dalam Python dan Java 461
10.9.1 Apa yang akan kita bangun 461
10.9.2 Situs-situs contoh dan servernya 463
10.9.3 Pembaca robots.txt yang mengikuti RFC 9309 472
10.9.4 Auditornya 475
10.9.5 Menjalankan auditor 494
10.9.6 Kejutan di pustaka standar Python 496
10.9.7 Mengarahkan auditor ke situs nyata 497
10.10 Profitina dalam Praktik 498

10.11 Kesalahpahaman yang Sering Terjadi 498
10.12 Ringkasan Bab 499
10.13 Soal Latihan 500
10.14 Penyelesaian Soal Latihan 501
Daftar Pustaka 504

BAB 11 Privasi dan Kepatuhan sejak Rancangan (UU PDP) 506
11.1 Titik Berangkat Kita 506
11.2 UU PDP No. 27/2022 dalam Bahasa Sederhana 507
11.2.1 Satu undang-undang, enam belas bab 507
11.2.2 Apa itu data pribadi? 507
11.2.3 Asas-asasnya 508
11.2.4 Dasar pemrosesan 508
11.2.5 Hak subjek data pribadi 508
11.2.6 Kewajiban pengendali 509
11.2.7 Sanksi 509
11.3 Posisi Hukum pada 2026 510
11.3.1 PP 33/2026, peraturan pelaksananya 510
11.3.2 Lembaga pengawas 510
11.3.3 Artinya bagi tim kecil 511
11.4 Privasi sejak Rancangan sebagai Metode Rekayasa 511
11.4.1 Gagasannya 511
11.4.2 Minimalkan dulu 511
11.4.3 Tuliskan: catatan pemrosesan 512
11.4.4 Privat secara bawaan, di mana-mana 512
11.4.5 Satu rumah untuk teks hukum 512
11.5 Kata Sandi: Mengapa Argon2id 512
11.5.1 Jangan pernah menyimpan kata sandinya 512
11.5.2 Mengapa *hash* yang *cepat* adalah fungsi satu arah yang salah 513
11.5.3 *Salt* 513
11.5.4 Argon2id 514
11.5.5 *Hash* yang tersimpan menyebut resepnya sendiri 514
11.6 Persetujuan yang Bermakna 515
11.6.1 Dari Pasal 22 ke kotak centang 515
11.6.2 Merancang log persetujuan 515
11.6.3 Penarikan 516
11.6.4 Anak 516
11.7 Pembatasan Laju Login 516
11.7.1 Serangannya 516
11.7.2 Jendela tetap dan jendela geser 516
11.7.3 Empat rincian yang penting 517
11.8 Hak yang Benar-Benar Dapat Dijangkau 518
11.8.1 Dari pasal ke tombol 518
11.8.2 Sebuah koreksi yang jujur 519
11.8.3 Ekspor: semuanya, dan hanya milik saya 519
11.8.4 Penghapusan: *purge*, dan tinggalkan *tombstone* hanya di tempat orang lain membutuhkannya 519
11.8.5 Ketika terjadi kegagalan 520
11.9 Halaman Legal HTML dan Satu Sumber Kebenaran 520
11.10 Lab G7: Log Persetujuan, Ekspor, dan Penghapusan 521
11.10.1 Apa yang akan kita bangun 521
11.10.2 *Hashing* kata sandi 522
11.10.3 Pembatas berjendela geser 525
11.10.4 Aplikasinya 526
11.10.5 Demonya 540

11.10.6 Uji-ujinya 548
11.11 Profitina dalam Praktik 557
11.12 Kesalahpahaman yang Sering Terjadi 558
11.13 Ringkasan Bab 559
11.14 Soal Latihan 560
11.15 Penyelesaian Soal Latihan 561

BAB 12 PROFIT: Koin Utilitas di Solana 567
12.1 Posisi Kita Sejauh Ini 567
12.2 Apa Itu Koin Utilitas, dan Apa yang Bukan 568
12.2.1 Kebanggaan yang hampa 568
12.2.2 Empat hal yang bukan PROFIT 568
12.2.3 Kerangka Indonesia 569
12.3 Solana dengan Kata-Kata Sederhana 570
12.3.1 Buku catatan bersama yang tak bisa diubah diam-diam 570
12.3.2 Akun, program, transaksi 570
12.3.3 Tiga klaster, satu untuk berlatih 570
12.3.4 Keadaan jaringan pada 2026 571
12.4 Token di Solana: Mint, Akun, dan Otoritas 572
12.4.1 Mint dan akun token 572
12.4.2 Desimal, dan mengapa jumlah selalu bilangan bulat 573
12.4.3 Dua otoritas 576
12.5 Fakta Publik PROFIT 577
12.6 Emisi yang Menurun 578
12.6.1 Masalah yang diselesaikan jadwal menurun 578
12.6.2 Aturan publiknya 578
12.6.3 Hukum pangkat yang tersembunyi 579
12.6.4 Lab G8, langkah 2: mereproduksi tabel 580
12.6.5 Dari jadwal ke imbalan nyata 583
12.7 Empat Jalan PROFIT Keluar: Pembakaran 583
12.7.1 Arti pembakaran 583
12.7.2 Empat pembakaran 584
12.8 Pasokan sebagai Deret Geometri 585
12.8.1 Model yang cukup sederhana untuk diselesaikan 585
12.8.2 Mengurai relasi rekurens 585
12.8.3 Keadaan tunak dan seberapa cepat dicapai 586
12.8.4 Lab G8, langkah 3: buku besar dalam mikro-PROFIT melawan bentuk tertutup 586
12.9 Kapan Pembakaran Menyusul Emisi? 593
12.9.1 Per pengguna, pertanyaannya menjadi sederhana 593
12.9.2 Lab G8, langkah 4: bentuk tertutup, bagi dua, dan oracle mainan 593
12.10 Oracle: Sebuah Lantai, Bukan Harga 598
12.10.1 Apa yang dihitung oracle 598
12.10.2 Apa yang menggeser lantai, dan apa yang tidak 599
12.11 Membeli, Membelanjakan, dan Mentransfer: Pagar Pengaman 599
12.11.1 Membeli melalui QRIS 599
12.11.2 Membelanjakan untuk layanan nyata 600
12.11.3 Mentransfer antarpengguna 600
12.11.4 Pemberian administrator, secara terbuka 601
12.12 Satu Tingkat Referal, dan Mengapa Tidak Lebih 601
12.12.1 Aturannya 601
12.12.2 Mengapa tingkat kedua adalah hal yang berbeda 601
12.13 Sudah Berjalan, dan di Peta Jalan 602
12.14 Profitina dalam Praktik 603
12.15 Kesalahpahaman yang Sering Terjadi 604
12.16 Ringkasan Bab 604

12.17 Soal Latihan 605
12.18 Penyelesaian Soal Latihan 606
Daftar Pustaka 608

BAB 13 Profitina Group dan Jalan ke Depan 609
13.1 Posisi Kita Sejauh Ini 609
13.2 Satu Payung, Beberapa Rumah 610
13.2.1 Bentuk kelompoknya 610
13.2.2 Enam lapisan bersama 610
13.2.3 Satu aturan untuk berbagi 611
13.3 Profitina Capital: Meja Kuantitatif 612
13.3.1 Apa itu 612
13.3.2 Lima ukuran pada sepuluh hari 612
13.3.3 Apa yang diajarkan Capital 613
13.4 Profitina TTM: Dari Teks ke Film Pendek 614
13.4.1 Alur pemrosesannya 614
13.4.2 Melukis di dua jalur 614
13.4.3 Gagal dengan lantang 615
13.5 Profitina AI: Pendamping AI Local-First 615
13.5.1 Apa yang dikerjakannya 615
13.5.2 Tidak pernah berhenti diam-diam 616
13.6 Profitina Laundry: Tempat Berlatih 616
13.7 Prinsip-Prinsip yang Mengikat 617
13.7.1 AI local-first 617
13.7.2 Aplikasi menyarankan; pengguna memutuskan 618
13.7.3 Tes sebagai gerbang 618
13.7.4 Dan satu lagi: katakan apa yang sebenarnya terjadi 618
13.8 Lab G9: Memeriksa Rekaman Peneliti dengan OpenAlex 619
13.8.1 Fitur di balik lab ini 619
13.8.2 OpenAlex secara singkat 619
13.8.3 Apa yang diperiksa pemeriksa 620
13.8.4 Langkah 1: menghitung ejaan dengan himpunan 621
13.8.5 Langkah 2: pemeriksanya 624
13.8.6 Menjalankannya, dan apa yang terjadi pada jaringan 637
13.8.7 Langkah 3: pengganti lokal untuk API 639
13.8.8 Mengembangkan pemeriksa lebih jauh 643
13.9 Apa Selanjutnya 644
13.9.1 Jalan yang telah dilalui 644
13.9.2 Jalan ke depan 644
13.9.3 Yang tidak akan berubah 645
13.9.4 Lab-lab, sekali lagi 645
13.10 Profitina dalam Praktik 646
13.11 Kesalahpahaman yang Sering Terjadi 647
13.12 Ringkasan Bab 647
13.13 Soal Latihan 648
13.14 Penyelesaian Soal Latihan 649
13.15 Sepatah Kata Penutup 650
Daftar Pustaka 651

Daftar Gambar

Gambar 1.1 Dua belas pekan pertama basis kode sekarang pada sumbu waktu. Titik kuning adalah peluncuran produksi 17 Juli 2026. 7
Gambar 1.2 Tiga belas bab dalam empat kelompok, dengan lab praktik setiap bab dalam huruf miring. Lab bertanda G1 sampai G9 adalah program bernomor buku ini. 11
Gambar 1.3 Meja kerja yang dipakai di seluruh buku. Anda bekerja di editor dan terminal VS Code; dua ekstensi mengajari VS Code tentang Python dan Java; terminal menemukan Python dan JDK melalui PATH sistem operasi. 12
Gambar 2.1 Pertanyaan yang sama di halaman hasil pencarian (kiri) dan di mesin jawaban (kanan). Di kiri, orang memindai daftar berperingkat lalu memilih; di kanan, mesin sudah memilih, dan hanya menyebut satu atau dua sumber. 33
Gambar 2.2 Tiga pengukuran dari **Tabel 2.1**: (a) proporsi kunjungan Google yang mengeklik hasil, dengan dan tanpa ringkasan AI (Pew, Maret 2025); (b) proporsi pencarian Google yang menampilkan AI Overview dengan selang setahun; (c) kunjungan ke Google AI Mode (keduanya Similarweb, dilaporkan Juli 2026). 35
Gambar 2.3 Cara crawler menerapkan `robots.txt` menurut RFC 9309: pilih grup yang menyebut dirinya (atau grup *), lalu biarkan aturan cocok yang terpanjang memutuskan, dengan seri dimenangkan `Allow`. 42
Gambar 2.4 Lab G6. Satu berkas fakta memasok generator yang menulis tiga berkas; pemeriksa terpisah membacanya kembali dan melapor; seorang manusia meninjau hasilnya sebelum apa pun dipublikasikan. 44
Gambar 3.1 Siklus visibilitas: ukur, jelaskan, perbaiki, dan pantau, dikelompokkan ke dalam tiga tab aplikasi, Pindai, Optimasi, dan Pantau (Scan, Optimize, Monitor). 84
Gambar 3.2 Sembilan langkah perjalanan pengguna. Langkah 1 dan 2 tidak memerlukan akun; langkah 4 sampai 6 adalah siklus SOM/POP, dan panah putus-putus menunjukkan bahwa siklus itu berulang. 93
Gambar 3.3 Pendaftaran, sebagai *wireframe*. (1) Kotak persetujuan mula-mula kosong; (2) yang diminta hanya email dan kata sandi; (3) kode yang dikirim melalui email mengaktifkan akun. 94
Gambar 3.4 Tab Pindai, sebagai *wireframe*. (1) Lokasi dan pertanyaan pelanggan terisi otomatis untuk diperiksa pemilik; (2) bila kategori belum pasti, sistem berkata “belum yakin” dan bertanya; (3) kemajuan ditampilkan bersama langkahnya dan sisa waktunya. 94
Gambar 3.5 Hasil pindaian, sebagai *wireframe* dengan nilai ilustratif. (1) Estimasi kesiapan; (2) tautan ke metodenya; (3) perbaikan berprioritas; (4) jalan masuk ke Mesin Konten; (5) umpan balik tentang pindaian. 95
Gambar 3.6 Tab Optimasi, sebagai *wireframe*. (1) Draf jawab-dulu yang langsung menyebut entitasnya; (2) putusan gerbang mutu; (3) JSON-LD yang sesuai; (4) pilihan *crawler* dari Pendamping Teknis; (5) pemilik sendiri yang menerbitkan di situsnya. 96
Gambar 3.7 Tab Pantau, sebagai *wireframe* dengan nilai ilustratif. (1) Tren mingguan; (2) peringatan atas perubahan yang berarti; (3) irama pindai ulang tiap paket; (4) ekspor CSV, fitur Pro. 97
Gambar 3.8 Panel Akun & Data Pribadi, sebagai *wireframe*. (1) Ekspor sebagai JSON; (2) riwayat persetujuan; (3) penghapusan permanen, yang hanya dapat dimulai pengguna; (4) hak-hak ini gratis di semua paket. 98
Gambar 3.9 Laporan Visibilitas Peneliti, sebagai *wireframe*. (1) Nama fiktif tidak ditemukan dan tidak diberi skor; (2) untuk rekaman nyata, laporan mencantumkan celah yang dapat ditutup; (3) hanya rekaman bibliografis yang dipakai. 99
Gambar 3.10 Satu Teks, Banyak Platform, sebagai *wireframe*. (1) Satu platform dipilih pada satu waktu; (2) posisi antrean dan waktu tunggu ditampilkan; (3) panjang teks diperiksa terhadap batas platform; (4) Bagikan ke X membuka X di akun pengguna sendiri. 100
Gambar 3.11 Arsitektur lab G1. Peladen mungil menerbitkan situs contoh di 127.0.0.1; `gather()` mengambil paling banyak empat sumber, `RobotsRules` dan `PageFacts` mengubahnya menjadi `Site`, delapan pemeriksaan menilainya, dan `report()` mencetak hasilnya. 100

Gambar 3.12 Poin tiap pemeriksaan untuk kedua situs contoh, diambil dari keluaran nyata `demo.py`: sebelum perbaikan (skor 13) dan sesudahnya (skor 100). 141
Gambar 4.1 Komunitas pada tingkat kotak. Satu bus peristiwa langsung memasok setiap bagian alun-alun publik dan ruang privat; lapisan keamanan Bab 5 menyaring apa yang ditulis sebelum ditampilkan. 157
Gambar 4.2 Tiga cara kabar sampai ke peramban. Polling bertanya setiap lima detik dan sering tidak mendapat apa-apa; Server-Sent Events membiarkan satu tanggapan tetap terbuka dan mendorong setiap kiriman saat terjadi; WebSocket meningkatkan sambungan agar kedua pihak dapat mengirim kapan saja. 159
Gambar 4.3 Utas adalah pohon balasan. Repost-kutip (#5) menunjuk balik ke kiriman yang dikutipnya dan, kecuali pembuatnya memilih lain, mewarisi kanal kiriman asli; repost polos yang sengaja dibagikan ke Umum (#6) pergi ke sana. 162
Gambar 4.4 Membuat kanal. Saldo sekurang-kurangnya 50 PROFIT diperlukan tetapi tidak pernah dipotong; paling banyak dua kanal baru per hari dan lima kanal aktif per anggota. 164
Gambar 4.5 Penghitung "N baru". Hanya kiriman tingkat atas sesudah penanda terakhir-dilihat yang dihitung, sehingga enam butir baru yang tiga di antaranya balasan menghasilkan lencana "3 baru". 165
Gambar 4.6 Kerangka rancangan kokpit dasbor. Satu permintaan mengisi pita prioritas tiga tingkat dan tiga kartu; setiap butir menaut ke bagiannya sendiri. 168
Gambar 4.7 Riwayat hidup kiriman optimistis. Kiriman langsung tampil sebagai tertunda; keberhasilan mengonfirmasinya dengan nomor aslinya, dan kegagalan menghapusnya serta mengembalikan teks beserta alasannya. 169
Gambar 4.8 Apa yang dilakukan halaman terhadap peristiwa tipis. Peristiwa untuk kanal lain hanya menandai basi *cache* dan memperbarui penghitung; peristiwa untuk kanal yang tampil mengambil satu kiriman; peristiwa untuk kiriman optimistis milik anggota sendiri diabaikan. 170
Gambar 4.9 Satu halaman umpan berisi 30 kiriman sebelum dan sesudah perbaikan September 2026 (skala log; nilai nol digambar sebagai potongan pendek). Waktu turun dari 868 ms menjadi 11 ms dan kueri dari 335 menjadi 12. 171
Gambar 4.10 Balapan jawaban basi. Tanpa penjaga, balasan lambat untuk kanal A menimpa kanal B; dengan nomor urut permintaan, balasan terlambat dikenali dan dibuang. 171
Gambar 4.11 Lab ini. Kedua server menyimpan daftar kiriman dan satu antrean per pendengar; peramban dan klien konsol mengirim permintaan biasa dan menerima peristiwa tipis lewat satu aliran yang terbuka. 172
Gambar 4.12 Halaman lab di Chromium tanpa layar setelah pemeriksaan otomatis (`?lang=id`). Penanda 1: lencana "1 baru" milik kanal yang tidak tampil; 2: sebuah balasan, menjorok; 3: status sambungan; 4: kotak penyusun, langsung kosong saat kiriman dikirim. 198
Gambar 5.1 Tiga ukuran lingkungan ancaman pada 2026: serangan phishing yang teramati di seluruh dunia dalam satu kuartal, laporan penipuan yang diterima Indonesia Anti-Scam Centre sejak dibuka, dan porsi lalu lintas web yang bersifat otomatis. 206
Gambar 5.2 Tautan yang menyembunyikan host sebenarnya. Semua yang ada sebelum `@` adalah informasi pengguna, sekadar umpan; host-nya adalah `receipt-view.click`. Baris bawah menyebut tiga tempat lain trik bersembunyi di dalam host. 207
Gambar 5.3 Keamanan tautan sebagai lapisan. Kedua lapisan berjalan sekali, di jalur tulis, dan putusannya disimpan bersama kiriman; menampilkan umpan hanya membaca putusan tersimpan dan tidak pernah memanggil model. 210
Gambar 5.4 Dua matriks kebingungan. Kiri: tolok ukur Profitina yang diterbitkan untuk lantai aturannya (27 dari 28 penipuan tertangkap, 0 positif palsu pada 30 teks normal). Kanan: penyaring lab Subbab 5.8 pada 60 sampel buku ini dengan $k = 2$. 213
Gambar 5.5 Apa yang dapat dan tidak dapat dilakukan moderator kanal. Tiga wewenang di dalam satu kanal, setiap tindakan dicatat dan pengirim kiriman diberi tahu; tanpa akses ke PROFIT, data pribadi, pesan langsung, atau kanal lain. 215
Gambar 5.6 Main adil pada tingkat kotak. Sebuah kontribusi memperoleh hadiah kecil hanya bila berasal dari manusia terverifikasi, masih di bawah batas harian, bukan duplikat, dan dinilai berguna; penyalahgunaan yang ditemukan belakangan ditarik kembali dan dibakar. 217
Gambar 5.7 Penyaring lab. Tautan diambil dan dipecah, sebelas aturan buku teks diterapkan, dan putusannya BLOCK bila ada aturan keras yang terpicu atau sekurang-kurangnya k aturan lunak terpicu. 219

Gambar 5.8 Presisi, *recall*, dan $F1$ penyaring lab ketika ambang lunak k bervariasi dari 1 sampai 5. k yang lebih rendah menangkap lebih banyak penipuan dan memblokir lebih banyak pesan jujur; mulai $k = 3$ tidak ada positif palsu, tetapi sebagian besar penipuan lolos. 239

Gambar 6.1 Ukuran unduhan model Qwen di pustaka Ollama, dalam gigabita, sebagaimana tercantum pada 24 September 2026 [15], [5], [14]. Dua batang hijau toska adalah model di balik jalur kualitas dan jalur cepat Profitina. 252

Gambar 6.2 Model dua jalur. Pekerjaan panjang menuju jalur kualitas, pekerjaan pendek ke jalur cepat, keduanya dilayani Ollama di mesin yang sama. Setiap jalur memiliki jalan kegagalan yang jujur. 253

Gambar 6.3 Gerbang kualitas sebagaimana dibangun di lab bab ini. Draf harus lolos semua cek wajib, lalu skornya harus benar-benar lebih tinggi daripada templat; bila tidak ada model yang tersedia, pengguna diberi tahu. 256

Gambar 6.4 Panggilan *streaming* ke `/api/generate`. Klien mengirim satu permintaan JSON; server menjawab dengan satu objek JSON per baris, dan objek terakhir bertanda `"done": true` serta membawa jumlah token dan durasinya. 272

Gambar 7.1 *Pipeline* umpan balik lima langkah. Kotak gelap adalah langkah-langkahnya; kotak abu-abu menunjukkan apa yang dilihat pemberi umpan balik; catatan berwarna kuning menandai dua tempat manusia memutuskan. Setelah rilis, siklus dimulai lagi dengan umpan balik baru. 305

Gambar 7.2 Perjalanan satu butir umpan balik. Perpindahan bergaris putus-putus bersifat otomatis dan dapat dijelaskan; perpindahan berwarna kuning adalah keputusan manusia. Hanya butir yang diterima yang dapat dirilis. 306

Gambar 7.3 Pertanyaan uji terpisah lab sebelum dan sesudah dataset. Enam pertanyaan diperbaiki, satu tetap lolos, dan E2 mengalami regresi: tingkat kelolosan naik dari 2/8 menjadi 7/8, tetapi *pipeline* tetap menyarankan penahanan. 311

Gambar 7.4 Batas harian lab sebanyak 3 butir berhadiah per hari WIB untuk pengguna u1. F10 pukul 23.30 WIB terkena batas; F11 pukul 00.10 WIB jatuh pada hari WIB yang baru. Batas yang dikaitkan pada tanggal UTC baru akan diatur ulang pukul 07.00 WIB. 315

Gambar 8.1 Pasar pada 2026. (a) Penetrasi internet per wilayah menurut survei APJII, dengan angka nasional 81,72% sebagai garis putus-putus; sumbu dimulai dari 60%. (b) Pengguna internet (APJII, 2026) di samping pengguna QRIS dan merchant QRIS (Bank Indonesia, Juni 2026). 343

Gambar 8.2 Struktur tingkat layanan sebelum dan sesudah 6–7 Agustus 2026. Ketiga fitur Agency sudah terbukti berjalan; tingkat itu dihapus demi kesederhanaan, dan sikap tanpa iklan selamanya dipertahankan. 346

Gambar 8.3 Ukur minat sebelum membangun pembayaran, sebagai tangga. Setiap anak tangga hanya dibangun bila anak tangga di bawahnya menunjukkan permintaan nyata; temuan Agustus 2026 berupa nol pembelian PROFIT menahan platform di anak tangga bawah. 351

Gambar 8.4 Satu hari pada dua jam. Pengaturan ulang yang benar terjadi pada 00.00 WIB; kode yang memakai tanggal UTC justru mengatur ulang pada 07.00 WIB, sehingga pengguna mendapati kuota kemarin yang sudah habis selama tujuh jam pertama hari mereka. 352

Gambar 8.5 State machine langganan dalam praktikum. Panah hijau toska menuju Pro, panah merah kembali ke Free, dan simpul pada Pro adalah pengingat serta perpanjangan otomatis; setiap keadaan juga menerima detak jam (`tick`). 363

Gambar 9.1 Arsitektur Profitina pada tingkat kotak. Peramban dan situs pemasaran menjangkau satu host Windows 11 melalui tepi Cloudflare dan terowongan yang hanya mengarah keluar; di dalamnya, NSSM menjaga Uvicorn dan FastAPI tetap hidup, dengan PostgreSQL dan AI lokal di sampingnya. Mesin jawaban adalah target yang diukur, tidak pernah dipanggil. 394

Gambar 9.2 Rilis enam langkah yang diverifikasi kesehatannya. Uji yang gagal atau muatan yang berubah selama gerbang berjalan membuat rilis ditolak tanpa menyentuh produksi; versi baru yang tidak menjawab pemeriksaan kesehatan tepat waktu dikembalikan (*rollback*) 400

Gambar 9.3 Berpindah kanal di feed komunitas, sebelum dan sesudah perbaikan September 2026: waktu untuk halaman 30 unggahan, kueri basis data, koneksi baru, dan panggilan AI per halaman. 402

Gambar 9.4 Balapan respons basi. Balasan terlambat untuk permintaan yang lebih awal tidak boleh menimpa balasan untuk permintaan terbaru; nomor urut pada setiap permintaan, yang diperiksa saat balasan tiba, membuang balasan yang basi. 403

Gambar 9.5 Min-heap pindaian terjadwal yang diurutkan menurut waktu jatuh tempo. Akar adalah tugas berikutnya; entri bergaris putus-putus adalah entri basi yang tertinggal akibat penjadwalan ulang, dan dilewati ketika mencapai akar. 404

Gambar 10.1 Tumpukan teknologi situs pemasaran. Permintaan melewati *edge* Cloudflare (lapisan *cache* 2) dan LiteSpeed Cache milik *host* (lapisan *cache* 1) sebelum WordPress dan tema dwibahasanya menyusun halaman; aplikasi hanya dijangkau melalui widget yang disematkan dan tautan legal. 446

Gambar 10.2 Blok terkelola yang disisipkan di *edge*. Grup `*` milik origin masih mengizinkan semua pihak, tetapi *crawler* yang disebut di blok terkelola kini memiliki grup sendiri dan tidak pernah melihat `*`; *crawler* yang tidak disebut tetap kembali ke `*`. 453

Gambar 10.3 Adopsi dan pemakaian `llms.txt` pada 2026. Sekitar satu dari sebelas situs teratas menerbitkan berkas itu, tetapi pada Mei 2026 hampir tidak ada berkas yang menerima satu permintaan pun 456

Gambar 10.4 Mengosongkan dua lapisan *cache*. Dengan urutan yang salah, pengunjung yang datang di antara dua *purge* membuat *edge* mengisi dirinya kembali dari *cache* dalam yang masih basi, dan salinan basi itu hidup selama satu masa hidup *edge* penuh; dengan urutan yang benar, jendela basi berakhir pada *purge* kedua. 458

Gambar 10.5 Permintaan-permintaan auditor untuk satu situs. Dua perbandingan membedakan salinan *cache* yang basi dari berkas yang ditulis ulang oleh suatu lapisan; permintaan terakhir mengambil halaman beranda yang segar untuk membandingkan `dateModified`. 462

Gambar 11.1 UU PDP dari pengesahan sampai peraturan pelaksananya, bersama tonggak kerja kepatuhan Profitina sendiri. Tanggal lembaga adalah perkiraan seorang pakar, bukan jadwal. 511

Gambar 11.2 Menyimpan kata sandi. *Salt* acak dan pengaturan biaya masuk ke fungsi penurunan kunci yang lambat; string yang disimpan mencatat resep, biaya, *salt*, dan hasilnya, sehingga login dapat menghitung ulang dan resep yang usang dapat ditingkatkan. 514

Gambar 11.3 Pembatas berjendela geser dengan nilai-nilai lab. Tiga kegagalan pada 09:00:05, 09:00:10, dan 09:00:15 memblokir percobaan keempat pada 09:00:20, bahkan dengan kata sandi yang benar; blokir terangkat ketika kegagalan pertama keluar dari jendela pada 09:01:05. 517

Gambar 11.4 Riwayat hidup data pribadi di Profitina: pengumpulan minimal, pemakaian yang terlindungi, hak yang dijalankan dari satu panel, penghapusan yang disusul *purge* produksi, dan kewajiban pemberitahuan kegagalan pelindungan yang melatari semuanya. 518

Gambar 11.5 Penghapusan di lab G7. Baris pengguna hilang; tulisan yang dibalas pengguna lain menjadi cangkang anonim; peristiwa persetujuan hanya menyimpan pengenal anonim, versi, dan waktu, dan satu peristiwa `withdrawn` ditambahkan. 520

Gambar 12.1 Sebuah token di Solana: satu akun mint menggambarkan token (supply, decimals, dan dua otoritas), setiap pemegang memiliki akun token yang menunjuk ke mint, dan akun metadata memberinya nama dan logo. Token Program menegakkan aturan untuk setiap perubahan 572

Gambar 12.2 Emisi harian menurut jadwal linear dan menurut jadwal menurun dari whitepaper, pada sumbu logaritmik. Tiga titik contoh whitepaper terletak pada garis menurun; pada sejuta pengguna selisihnya 64 kali lipat. 580

Gambar 12.3 Bagaimana PROFIT masuk dan keluar dari peredaran. Perolehan dan pembelian QRIS menambah peredaran; empat pembakaran menghapus PROFIT secara permanen; lantai oracle membagi cadangan Rupiah dengan jumlah yang beredar. 584

Gambar 12.4 Pasokan beredar di bawah emisi tetap 3.125 PROFIT per hari dan tiga laju pembakaran harian, dari buku besar bilangan bulat Lab G8. Setiap kurva mendekati keadaan tunaknya sendiri E/b; laju pembakaran yang lebih tinggi memberi keadaan tunak yang lebih rendah dan tercapai lebih cepat. Nilai ilustratif saja. 592

Gambar 12.5 Emisi per pengguna $m(u)$ turun seiring tumbuhnya komunitas; pembakaran per pengguna yang konstan β adalah garis mendatar. Pembakaran menyusul emisi di sebelah kanan setiap perpotongan $u*$. Nilai β ilustratif. 598

Gambar 13.1 Profitina Group. Lapisan bersama berada di puncak dan disalin ke bawah ke setiap produk, tidak pernah kembali ke atas. Profitina Laundry, tempat berlatih, hanya meminjam gagasan dan pola, tidak pernah kode atau data. 610

Gambar 13.2 Tiga prinsip sebagai satu siklus. Data masuk ke model lokal; model memberi saran beserta alasannya; manusia memutuskan; tes berdiri di antara setiap perubahan dan rilisnya, yang baru dianggap selesai setelah pemeriksaan kesehatan mengonfirmasinya. 617

Gambar 13.3 Lab G9. Sebuah nama dicari; hanya kecocokan persis setelah normalisasi yang lolos penjaga, dan selain itu dilaporkan sebagai tidak ditemukan tanpa skor. Ketika jaringan diblokir, sampel tersimpan atau pengganti lokal menjawab tiga panggilan yang sama. 620
Gambar 13.4 Jalan yang telah dilalui (garis utuh) dan jalan ke depan (garis putus-putus). Segala hal di sebelah kiri September 2026 telah terjadi; segala hal di sebelah kanannya direncanakan dan bergerbang. 644

Daftar Tabel

Tabel 1.1 Tonggak bertanggal Profitina dari 3 Juli hingga 24 September 2026, beserta bab buku ini yang membahas masing-masing. Baris terakhir adalah rencana, bukan fakta. 6
Tabel 1.2 Lima jenis pembaca, apa yang akan ditemukan masing-masing dalam buku ini, dan bab mana yang sebaiknya dibaca lebih dahulu. 10
Tabel 1.3 Memasang VS Code di setiap sistem operasi. 13
Tabel 1.4 Memasang JDK 25 di setiap sistem operasi. 14
Tabel 1.5 Memasang Python di setiap sistem operasi. 14
Tabel 1.6 Ekstensi VS Code yang dipakai dalam buku ini. 15
Tabel 1.7 Perintah pemeriksaan, apa yang seharusnya dicetak masing-masing, dan perbaikan yang lazim bila tidak. 15
Tabel 2.1 Pengukuran pergeseran dari daftar ke jawaban, 2025–2026, dengan tanggal dan sumber masing-masing. “jt” berarti juta. 34
Tabel 2.2 Hasil terpilih makalah GEO pada hitungan kata yang disesuaikan posisi: tiga metode terbaik dan penjejalan kata kunci dibandingkan dengan dasar yang tidak diubah [2]. 38
Tabel 2.3 SEO dan GEO dibandingkan. Fondasinya sama; yang berbeda adalah siapa yang memilih sumber dan apa yang dihitung sebagai keberhasilan. 39
Tabel 2.4 Nama-nama crawler yang dipakai dalam `robots.txt` lab G6, dengan uraian tugas crawler dari perusahaan masing-masing. 43
Tabel 3.1 Empat kata kerja siklus dan tiga tab aplikasi: SOM dalam bahasa Inggris, POP dalam bahasa Indonesia. 85
Tabel 3.2 Tiga keluarga sinyal teramati di balik estimasi kesiapan, dan standar publik yang memungkinkan masing-masing diamati. 87
Tabel 3.3 Dua paket sebagaimana digambarkan kartu paket aplikasi pada September 2026. 98
Tabel 3.4 Delapan pemeriksaan lab G1 beserta aturannya. Ambang-ambangnya adalah pilihan ajar buku ini. 101
Tabel 4.1 Bagian-bagian komunitas Profitina dan kapan masing-masing mulai berjalan di produksi (tanggal dari *whitepaper* publik dan riwayat rilis). 158
Tabel 4.2 Perbandingan polling, WebSocket, dan Server-Sent Events. 161
Tabel 4.3 Aturan pesan langsung dan alasan masing-masing. 166
Tabel 4.4 Antarmuka server lab (sama persis dalam Python dan Java). 172
Tabel 5.1 Trik yang umum dalam tautan berbahaya, dengan contoh rekaan. 208
Tabel 5.2 Kesebelas aturan penyaring lab. Daftar-daftar di baliknya adalah contoh milik buku ini dan dimaksudkan untuk disunting. 220
Tabel 5.3 Tabel jarak suntingan untuk `pay` dan `pya`, diisi baris demi baris dengan Persamaan (5.10); jawabannya ada di sel kanan bawah. 221
Tabel 5.4 Beberapa dari 60 sampel berlabel (30 penipuan, 30 normal) dalam `data/samples.tsv`. 232
Tabel 6.1 Generasi Qwen yang relevan untuk inferensi lokal, per 24 September 2026. Model “MoE” (*mixture of experts*) seperti 30B-A3B memiliki total 30 miliar parameter tetapi hanya memakai sekitar 3 miliar untuk setiap token. 252
Tabel 6.2 Jalur kualitas dan jalur cepat pada rilis September 2026. 254
Tabel 7.1 Keadaan-keadaan butir umpan balik di lab bab ini, siapa yang memindahkan butir ke setiap keadaan, dan apa yang dilihat pemberi umpan balik. 306

Tabel 7.2 Aturan perolehan PROFIT yang publik (*whitepaper* v24, September 2026). Jumlahnya dikalikan lagi dengan pengali emisi yang menurun dan dengan penilaian kebermanfaatan oleh AI. 313
Tabel 7.3 Sebelas butir umpan balik lab dalam `submissions.tsv`. Waktu dalam UTC; tambahkan tujuh jam untuk WIB. 317
Tabel 8.1 Dua tingkat layanan sebagaimana dijelaskan kartu paket pada rilis September 2026. 344
Tabel 8.2 Apa yang sudah berjalan dan apa yang belum, untuk semua bagian model bisnis yang melibatkan pembayaran. 348
Tabel 8.3 Tabel transisi mesin langganan. Peristiwa apa pun yang tidak tercantum untuk keadaan saat ini ditolak dengan galat. 364
Tabel 8.4 Kombinasi fitur-dan-tingkat serta perpindahan tingkat berarah untuk $F = 12$. 389
Tabel 9.1 Waktu mati yang diperbolehkan pada target ketersediaan yang lazim, menurut Persamaan (9.2). 392
Tabel 9.2 Komponen-komponen **Gambar 9.1**, pilihan untuk masing-masing, dan alasan utamanya. 397
Tabel 9.3 Suite otomatis yang menjadi gerbang setiap rilis, pada saat tulisan ini dibuat (24 September 2026). 398
Tabel 9.4 Bagaimana gerbang praktikum menirukan setiap langkah rilis sungguhan. 423
Tabel 10.1 Dua permukaan publik merek Profitina dan tanggung jawab masing-masing. 447
Tabel 10.2 Cara RFC 9309 meminta *crawler* memperlakukan setiap jenis respons atas permintaan `robots.txt`. Google juga memperlakukan status 429 seperti galat server. 451
Tabel 10.3 Direktif dan *header caching* HTTP yang dibaca auditor di lab, sebagaimana didefinisikan RFC 9111 untuk *cache* bersama. 457
Tabel 10.4 Empat situs contoh di lab. Dua di antaranya juga membuka origin-nya di *port* kedua, sehingga auditor dapat membandingkan apa yang diterima pengunjung dengan apa yang disimpan origin. 462
Tabel 11.1 Kecepatan terukur empat cara meng-*hash* kata sandi pada satu inti CPU mesin yang dipakai untuk buku ini (September 2026). Penyerang sungguhan jauh lebih cepat dengan perangkat keras khusus; yang penting adalah perbandingannya. 513
Tabel 11.2 Hak subjek data menurut UU PDP dan fitur swalayan, atau saluran, yang dipakai pengguna Profitina untuk menjalankan masing-masing hak. 518
Tabel 12.1 Tiga klaster publik Solana, sebagaimana dijelaskan dalam dokumentasi resmi pada September 2026. 571
Tabel 12.2 Fakta publik PROFIT, dari whitepaper publik v24 dan aplikasi, per September 2026. 577
Tabel 12.3 Tabel emisi ilustratif dari whitepaper: pengali menjadi separuh untuk setiap pertumbuhan pengguna sepuluh kali lipat. 578
Tabel 12.4 Tindakan perolehan, imbalan dasar, dan batasnya sebagaimana tercantum dalam whitepaper publik. Setiap imbalan dasar dikalikan pengali emisi dan skor kebergunaan; batas harian diatur ulang pukul 00.00 WIB. 583
Tabel 12.5 Apa yang sudah berjalan dan apa yang ada di peta jalan untuk PROFIT, per September 2026. 602
Tabel 13.1 Lab-lab dalam buku ini, gagasan yang diajarkan masing-masing, dan prinsip yang paling dilayaninya. 646

Daftar Kode

Kode 1.1 `check_setup.py`: melaporkan sistem operasi, Python yang menjalankan program, dan alat meja kerja mana yang dapat dijangkau PATH. 16
Kode 1.2 `CheckSetup.java`: laporan yang sama dalam Java, dengan penelusuran PATH kecil yang ditulis sendiri. 17
Kode 1.3 `hello.py`: program pertama; membuktikan bahwa editor, terminal, dan Python bekerja bersama. 18
Kode 1.4 `Hello.java`: program pertama dalam Java. 19

Kode 1.5 `milestones.py`: membaca tonggak, mencetak jaraknya dari peluncuran, dan menerapkan Persamaan (1.1) pada banyaknya hari hidup. 20
Kode 1.6 `Milestones.java`: tabel yang sama dalam Java, dengan `LocalDate` dan `ChronoUnit.DAYS`. 21
Kode 1.7 `sol_countdown.py`: hari, pekan, dan nama hari di antara dua tanggal. 28
Kode 1.8 `SolCountdown.java`: perhitungan yang sama dengan `LocalDate` dan `ChronoUnit`. 28
Kode 2.1 `business.json`: fakta-fakta yang dipelihara pemilik Profitina Laundry, satu-satunya sumber bagi ketiga berkas yang dibangkitkan. 45
Kode 2.2 `geo_generator.py`: membaca `business.json`, memvalidasinya, lalu menulis `jsonld.html`, `robots.txt`, dan `llms.txt` ke sebuah folder. 45
Kode 2.3 `MiniJson.java`: pengurai dan pencetak rapi JSON yang minimal untuk lab ini, dengan posisi galat dilaporkan sebagai baris dan kolom. 52
Kode 2.4 `GeoGenerator.java`: generator yang sama dalam Java, memakai `MiniJson` dan peta berurutan. 58
Kode 2.5 `geo_check.py`: mengurai JSON-LD, menerapkan `robots.txt` pada sederet crawler dan jalur, dan memeriksa struktur `llms.txt`. 62
Kode 2.6 `Robots.java`: pembaca `robots.txt` kecil yang menerapkan pemilihan grup dan aturan cocok terpanjang RFC 9309 (tanpa *wildcard* `*` dan `$`). 66
Kode 2.7 `GeoCheck.java`: pemeriksaan yang sama dalam Java, memakai `MiniJson` dan `Robots`. 68
Kode 2.8 `sol_rule_order.py`: cara baca cocok-pertama dan cocok-terpanjang untuk satu grup `robots.txt`, berdampingan. 78
Kode 2.9 `SolRuleOrder.java`: perbandingan yang sama dalam Java. 78
Kode 3.1 `sites/before/robots.txt`: berkas yang dikelola panel, yang memblokir dua *crawler* AI dan satu *crawler* lain berdasarkan nama, dan mengizinkan semua yang lain. 102
Kode 3.2 `sites/before/index.html`: beranda sebelum perbaikan. *Title*-nya "Home", ada dua judul `h1`, JSON-LD-nya berakhir dengan koma, dan nama usaha hanya muncul di *footer*. 102
Kode 3.3 `sites/after/robots.txt`: setiap *crawler* boleh membaca halaman publik; `/admin/` dan `/api/` dijauhkan, `CCBot` diblokir atas pilihan pemilik, dan sitemap disebutkan. 104
Kode 3.4 `sites/after/llms.txt`: judul H1, ringkasan, dan daftar halaman utama situs. 104
Kode 3.5 `sites/after/index.html`: beranda setelah perbaikan, dengan pembuka jawab-dulu, kerangka judul yang rapi, dan satu graf JSON-LD yang sah. 104
Kode 3.6 `serve_site.py`: peladen berkas statis untuk satu situs contoh, dengan satu baris log per permintaan. 106
Kode 3.7 `ServeSite.java`: peladen yang sama dengan `com.sun.net.httpserver`, termasuk penjaga *path traversal*. 108
Kode 3.8 `robots_rules.py`: mengurai `robots.txt` menjadi grup dan memutuskan berdasarkan kode status, pemilihan grup, dan kecocokan terpanjang. 110
Kode 3.9 `RobotsRules.java`: pengurai dan prosedur keputusan yang sama dalam Java, dengan *record* untuk aturan, grup, dan keputusan. 112
Kode 3.10 `page_facts.py`: pengumpul *title*, tag meta, judul, blok JSON-LD, dan kata terlihat berbasis peristiwa, dibangun di atas `html.parser`. 115
Kode 3.11 `PageFacts.java`: pengumpul yang sama dalam Java, dengan pemecah token berbasis ekspresi reguler dan penanganan teks mentah untuk `script` dan `style`. 117
Kode 3.12 `checks.py`: delapan pemeriksaan. Masing-masing menerima sebuah `Site` dan mengembalikan poin, bukti, dan saran. 122
Kode 3.13 `Checks.java`: *record* `Site` dan `CheckResult`, antarmuka `Check`, dan delapan pemeriksaan. 126
Kode 3.14 `readiness.py`: mengambil dengan sopan, menjalankan pemeriksaan, mencetak laporan, dan menghitung skor ajar. 131
Kode 3.15 `Readiness.java`: pengambilan, pelaporan, dan penilaian yang sama dengan `java.net.http.HttpClient`. 133
Kode 3.16 `demo.py`: menyajikan kedua situs contoh di porta 7030 dan 7031, memeriksanya, dan selalu mematikan peladennya. 137
Kode 3.17 `Demo.java`: demonstrasi yang sama dalam Java. 138

Kode 3.18 `robots_compare.py`: keputusan RFC 9309 dari `RobotsRules` di samping keputusan `urllib.robotparser`.....148
Kode 3.19 `RobotsCompare.java`: keputusan RFC 9309 untuk enam kasus yang sama.....149
Kode 3.20 `robots_patterns.py`: mencocokkan jalur aturan yang memuat `*` dan `$` di akhir.....151
Kode 3.21 `RobotsPatterns.java`: pencocokan yang sama dalam Java, dengan `Pattern.quote` untuk bagian-bagian biasa.....151
Kode 4.1 `feed_server.py`: server umpan langsung dengan Server-Sent Events, hanya pustaka standar.....173
Kode 4.2 `FeedServer.java`: server yang sama dengan `com.sun.net.httpserver`, memakai satu antrean pemblokir per pendengar.....178
Kode 4.3 `MiniJson.java`: JSON secukupnya untuk lab, sesuai dengan keluaran Python.....183
Kode 4.4 `feed.html`: halaman langsung dengan lencana kanal, pengiriman optimistis, pengambilan ulang satu butir, dan penjaga balasan basi.....186
Kode 4.5 `feed_client.py`: mendengarkan aliran dan mengirim, dalam Python.....191
Kode 4.6 `FeedClient.java`: klien yang sama dengan `java.net.http.HttpClient` dan `BodyHandlers.ofLines()`.....193
Kode 5.1 `link_screener.py`: penyaring tautan berbasis aturan dengan sebelas aturan buku teks dan aturan keputusan Persamaan (5.9).....221
Kode 5.2 `LinkScreener.java`: penyaring yang sama dalam Java, menghasilkan keluaran yang identik bita demi bita.....226
Kode 5.3 `evaluate.py`: matriks kebingungan, lima ukuran, daftar kesalahan, dan penyapuan atas *k*.....232
Kode 5.4 `Evaluate.java`: evaluasi yang sama dalam Java.....234
Kode 6.1 `stub_ollama.py`: pengganti `/api/version`, `/api/tags`, dan `/api/generate` milik Ollama, dengan *streaming* berupa JSON yang dipisah baris baru.....259
Kode 6.2 `StubOllama.java`: *stub* yang sama dengan `com.sun.net.httpserver`; `sendResponseHeaders(200, 0)` memilih *streaming* bertahap (*chunked*).....262
Kode 6.3 `MiniJson.java`: pembaca JSON *recursive-descent* yang minimal beserta penyusunnya, cukup untuk lab ini.....267
Kode 6.4 `ollama_client.py`: klien pustaka standar untuk antarmuka HTTP Ollama, dengan satu eksepsi jujur untuk setiap cara model bisa tidak ada.....272
Kode 6.5 `OllamaClient.java`: klien yang sama dengan `java.net.http.HttpClient`; *streaming* memakai `BodyHandlers.ofLines()`.....275
Kode 6.6 `two_lanes.py`: jalur kualitas yang menjalankan satu pekerjaan panjang sekaligus dan melaporkan posisi antrean, jalur cepat untuk label, dan lantai aturan di bawahnya.....280
Kode 6.7 `TwoLanes.java`: pengarah dua jalur dalam Java, dengan `record`, `ReentrantLock`, dan *thread*.....283
Kode 6.8 `quality_gate.py`: cek wajib, skor lunak, dan perbandingan ketat terhadap templat yang benar.....287
Kode 6.9 `QualityGate.java`: gerbang yang sama dalam Java; `show` meniru `textwrap.fill` Python sehingga keduanya mencetak baris yang identik.....289
Kode 6.10 `json_label.py`: minta JSON, validasi bentuk dan nilai yang diizinkan, coba ulang sekali, lalu serahkan kepada manusia.....298
Kode 6.11 `JsonLabel.java`: validasi dan percobaan ulang yang sama dalam Java, memakai `MiniJson`.....299
Kode 7.1 `feedback_pipeline.py`: kirim, kurasi dengan keputusan manusia, ekspor dataset JSONL, beri hadiah di bawah batas harian WIB, ukur sebelum dan sesudah, lalu tunggu keputusan rilis dari manusia.....317
Kode 7.2 `FeedbackPipeline.java`: *pipeline* yang sama dalam Java, dengan *record*, `java.time` untuk WIB, satuan mikro berupa bilangan bulat, dan penyusun JSON Lines buatan sendiri.....322
Kode 7.3 `day_key.py`: tanggal UTC dan tanggal WIB dari empat saat.....331
Kode 7.4 `DayKey.java`: perbandingan yang sama dengan `OffsetDateTime` dan `ZoneOffset.ofHours(7)`.....332
Kode 8.1 `tiers.py`: paket sebagai data, buku kuota yang menghitung pindaian per hari WIB, tambahan kuota untuk hari yang sama, dan bug tanggal UTC sebagai pembanding.....354

Kode 8.2 `Tiers.java`: paket, buku kuota, dan peragaan yang sama dalam Java, memakai `java.time`. 359
Kode 8.3 `subscription.py`: state machine langganan, dengan pesanan QRIS, verifikasi, penukaran PROFIT, pengingat, dan perpanjangan otomatis. 364
Kode 8.4 `Subscription.java`: state machine yang sama dalam Java, dengan `EnumMap` sebagai tabel transisi. 369
Kode 8.5 `test_tiers.py`: delapan uji untuk tabel paket, hari WIB, batas harian, tambahan kuota, monitor, dan riwayat. 375
Kode 8.6 `test_subscription.py`: sepuluh uji untuk state machine langganan. 376
Kode 8.7 `MiniTest.java`: pelari uji dalam lima puluh baris, sehingga uji Java tidak memerlukan apa pun selain JDK. 378
Kode 8.8 `TestTiers.java`: uji kuota dalam Java. 379
Kode 8.9 `TestSubscription.java`: uji langganan dalam Java. 381
Kode 8.10 `until_reset.py`: detik hingga 00.00 WIB berikutnya. 386
Kode 8.11 `UntilReset.java`: perhitungan yang sama dengan `Duration.between`. 386
Kode 8.12 `late_renewal.py`: aturan lama dan aturan yang adil untuk perpanjangan yang sangat terlambat. 388
Kode 8.13 `LateRenewal.java`: perbandingan yang sama dalam Java. 388
Kode 9.1 `scheduler.py`: penjadwal min-heap dengan irama per tingkat, penghapusan malas, pengejaran yang melompat alih-alih menyembur, dan jam simulasi. 406
Kode 9.2 `Scheduler.java`: penjadwal yang sama dengan `java.util.PriorityQueue` dan *record* 409
Kode 9.3 `test_scheduler.py`: tujuh uji untuk penjadwal. Gerbang rilis menjalankan persis berkas ini. 413
Kode 9.4 `TestScheduler.java`: tujuh uji yang sama dalam Java. 415
Kode 9.5 `health.py`: *endpoint* kesehatan yang melaporkan versi yang berjalan dan tiga pemeriksaan, serta membedakan mati dari menurun. 417
Kode 9.6 `HealthServer.java`: *endpoint* yang sama dengan `com.sun.net.httpserver`. 420
Kode 9.7 `release_gate.py`: gerbang rilis enam langkah yang menolak bila uji gagal atau muatan berubah, dan menyatakan berhasil hanya ketika `/api/health` menjawab versi baru. 424
Kode 9.8 `ReleaseGate.java`: gerbang yang sama dalam Java, dengan `ProcessBuilder`, `MessageDigest`, dan `java.net.http.HttpClient` 429
Kode 9.9 `weekly_trend.py`: rata-rata per minggu ISO dari skor satu monitor, dengan peringatan perubahan antarminggu 439
Kode 9.10 `WeeklyTrend.java`: tren yang sama dalam Java. 440
Kode 9.11 `latest_only.py`: penjaga nomor urut terhadap balasan basi. 441
Kode 9.12 `LatestOnly.java`: penjaga yang sama dalam Java. 441
Kode 10.1 `site_server.py`: menyajikan situs-situs contoh, meniru lapisan *edge* (`_edge/`) dan *cache* basi (`_cache/`) di depan setiap origin. 463
Kode 10.2 `SiteServer.java`: server situs contoh yang sama di atas server HTTP bawaan JDK. 466
Kode 10.3 `robots_rules.py`: pembaca yang menerapkan Persamaan (10.1) (penggabungan, kembali ke *) dan Persamaan (10.2) (kecocokan terpanjang, `Allow` bila seri), lengkap dengan karakter pengganti `*` dan `$`. 472
Kode 10.4 `RobotsRules.java`: pembaca RFC 9309 yang sama dalam Java. 473
Kode 10.5 `auditor.py`: memeriksa `robots.txt`, `llms.txt`, *header cache*, dan JSON-LD setiap situs, lalu mencetak laporan dan tabel ringkasan. 475
Kode 10.6 `MiniJson.java`: pembaca JSON kecil dan ketat untuk auditor Java; apa pun yang bukan JSON sah, seperti koma di akhir, melemparkan eksepsi. 483
Kode 10.7 `Auditor.java`: auditor yang sama dalam Java; mencetak laporan yang persis sama dengan **Kode 10.5**. 485
Kode 10.8 `robots_quirk.py`: dua pertanyaan yang sama diajukan kepada `urllib.robotparser` dan kepada pembaca RFC 9309 di **Kode 10.3**. 496
Kode 11.1 `passwords.py`: PBKDF2-HMAC-SHA256 dari pustaka standar, dan Argon2id melalui paket opsional `argon2-cffi`; setiap *hash* yang tersimpan menyebut resepnya. 522
Kode 11.2 `Pbkdf2Hasher.java`: resep PBKDF2 yang sama dengan `SecretKeyFactory` milik JDK. 523

Kode 11.3 `rate_limit.py`: pembatas berjendela geser untuk login yang gagal, dengan jam yang diberikan dari luar agar uji dapat mengendalikan waktu. 525
Kode 11.4 `SlidingWindowLimiter.java`: pembatas yang sama dalam Java. 525
Kode 11.5 `privacy_app.py`: pendaftaran dengan persetujuan eksplisit, login berbatas laju dengan peningkatan *hash*, ekspor, pembetulan, dan penghapusan dengan *tombstone*. 526
Kode 11.6 `PrivacyApp.java`: aplikasi yang sama dalam Java. 531
Kode 11.7 `Json.java`: penghasil JSON kecil dengan tata letak yang sama seperti `json.dumps(indent=2)` di Python. 539
Kode 11.8 `demo.py`: persetujuan, *hashing*, pembatasan laju, ekspor, pembetulan, dan penghapusan dalam satu skenario. 541
Kode 11.9 `Demo.java`: skenario yang sama dalam Java. 543
Kode 11.10 `test_privacy_app.py`: sepuluh uji, termasuk uji jawaban-diketahui untuk PBKDF2 dan penelusuran setiap tabel setelah penghapusan. 548
Kode 11.11 `PrivacyAppTest.java`: sepuluh sifat yang sama dalam Java, dengan pelari uji minimal. 552
Kode 11.12 `sol_policy.py`: meminta persetujuan lagi setelah kebijakan berubah, di atas lab G7. 562
Kode 11.13 `SolPolicy.java`: hal yang sama dalam Java (kolom `policyVersion` pada `PrivacyApp` sengaja tidak `final`). 562
Kode 11.14 `sol_sqlite.py`: tabel-tabel lab G7 di SQLite, dengan penghapusan sebagai satu transaksi. 564
Kode 12.1 `01_units.py`: jumlah PROFIT sebagai bilangan bulat mikro-PROFIT, dengan Persamaan (12.1) diterapkan ke dua arah. 573
Kode 12.2 `Units.java`: konversi yang sama dalam Java, dengan hitungan mikro-PROFIT bertipe `long`. 574
Kode 12.3 `02_emission.py`: pengali menurun dari Persamaan (12.2), emisi harian dari Persamaan (12.3), dan bentuk hukum pangkatnya Persamaan (12.5). 580
Kode 12.4 `Emission.java`: tabel yang sama dalam Java, dengan format yang identik. 581
Kode 12.5 `03_supply_sim.py`: relasi rekurens Persamaan (12.6) dijalankan sebagai buku besar bilangan bulat, diperiksa terhadap bentuk tertutup Persamaan (12.8), dengan keadaan tunak Persamaan (12.9) dan $t90$ dari Persamaan (12.10). 586
Kode 12.6 `SupplySim.java`: simulasi buku besar yang sama dalam Java, dengan larik mikro-PROFIT bertipe `long`. 588
Kode 12.7 `04_breakeven_oracle.py`: jumlah pengguna titik impas menurut Persamaan (12.11) dan menurut metode bagi dua, serta lantai oracle mainan. 594
Kode 12.8 `BreakEvenOracle.java`: dua perhitungan yang sama dalam Java. 595
Kode 13.1 `normalize.py`: dua penormal dan sebuah himpunan yang menghitung ejaan yang benar-benar berbeda. 621
Kode 13.2 `Normalize.java`: penormal yang sama dalam Java. 622
Kode 13.3 `researcher_check.py`: pemeriksa rekaman peneliti OpenAlex, dengan penjaga nama persis, laju menurut Persamaan (13.6), dan kegagalan jaringan yang jujur. 624
Kode 13.4 `Json.java`: pembaca dan penyusun JSON *recursive descent* yang kecil, cukup untuk lab ini. 628
Kode 13.5 `ResearcherCheck.java`: pemeriksa yang sama dalam Java, dengan antarmuka `Source` dan dua implementasi. 631
Kode 13.6 `sample_server.py`: pengganti lokal untuk tiga panggilan OpenAlex, menjawab dari sampel tersimpan. 639
Kode 13.7 `SampleServer.java`: server pengganti yang sama dalam Java. 640

Daftar Pseudokode

Pseudokode 3.1 Memutuskan apakah sebuah *crawler* boleh mengambil suatu jalur menurut RFC 9309: kode status lebih dulu, lalu pemilihan grup, lalu aturan cocok yang terpanjang, dengan hasil seri dimenangkan `Allow`. 109

Daftar Persamaan

(1.1) 7
(2.1) 35
(2.2) 37
(3.1) 90
(3.2) 101
(3.3) 110
(3.4) 115
(3.5) 125
(3.6) 125
(3.7) 125
(4.1) 159
(4.2) 159
(4.3) 160
(4.4) 163
(4.5) 163
(4.6) 165
(4.7) 170
(5.1) 212
(5.2) 212
(5.3) 212
(5.4) 212
(5.5) 213
(5.6) 214
(5.7) 214
(5.8) 218
(5.9) 220
(5.10) 220
(6.1) 248
(6.2) 249
(6.3) 249
(6.4) 254
(6.5) 254
(6.6) 254
(6.7) 256
(6.8) 257
(7.1) 307
(7.2) 308
(7.3) 310
(7.4) 310
(7.5) 311

(7.6) 312
(7.7) 314
(7.8) 314
(7.9) 314
(7.10) 314
(7.11) 315
(8.1) 342
(8.2) 345
(8.3) 346
(8.4) 349
(8.5) 353
(8.6) 353
(8.7) 353
(9.1) 392
(9.2) 392
(9.3) 402
(9.4) 403
(9.5) 404
(9.6) 405
(9.7) 405
(9.8) 424
(10.1) 450
(10.2) 450
(10.3) 457
(10.4) 457
(10.5) 459
(10.6) 459
(11.1) 513
(11.2) 517
(11.3) 517
(12.1) 573
(12.2) 578
(12.3) 579
(12.4) 579
(12.5) 579
(12.6) 585
(12.7) 585
(12.8) 586
(12.9) 586
(12.10) 586
(12.11) 593
(12.12) 598
(12.13) 599
(12.14) 601
(13.1) 612
(13.2) 612
(13.3) 613
(13.4) 613
(13.5) 613
(13.6) 620

Kata Pengantar

Tanyakan kepada asisten AI binatu mana di Surabaya yang buka sampai malam, klinik mana yang menerima pasien tanpa janji, atau buku mana yang paling jelas menerangkan algoritma, dan ia akan menjawab dalam satu atau dua kalimat. Ia tidak memberi Anda sepuluh tautan biru. Ia memberi Anda sebuah *nama*. Apakah nama itu nama usaha Anda bergantung pada sesuatu yang dua tahun lalu hampir tidak dikenal oleh pemilik usaha di Indonesia: seberapa siap situs web Anda untuk dibaca, dipahami, dan dikutip oleh mesin penjawab. Profitina ada untuk mengukur kesiapan itu, menjelaskannya dengan kata-kata sederhana, dan membantu memperbaikinya.

Buku ini menuturkan bagaimana Profitina dibangun secara terbuka, dari ruang kelas seorang dosen di Surabaya hingga menjadi platform yang berjalan di lingkungan produksi di app.profitina.com. Isinya kisah sebuah produk, tetapi ditulis sebagai buku ajar: setiap gagasan dijelaskan dari dasar, dan setiap bab memiliki praktikum yang Anda bangun sendiri, dalam **Python** dan **Java**, sehingga Anda memahami rekayasanya, bukan hanya kabar utamanya.

Untuk siapa buku ini ditulis

Mahasiswa informatika yang ingin melihat bagaimana mata kuliah yang mereka pelajari — struktur data, pemrograman web, basis data, keamanan, kecerdasan buatan — bertemu di dalam sebuah produk nyata. Pemilik usaha dan pemasar yang ingin memahami mengapa mesin penjawab AI penting dan apa yang sungguh-sungguh dapat dilakukan. Pengembang yang menginginkan cerita jujur tentang keputusan, pertimbangan, dan kekeliruan di balik sebuah platform kecil dengan pengguna sungguhan. Anda tidak memerlukan pengetahuan awal apa pun: Bab 1 dimulai dari komputer kosong dan berakhir dengan program-program pertama Anda berjalan.

Isi buku ini

Bab 1 menuturkan kisah awal dan menyiapkan meja kerja Anda: VS Code, Python, dan Java di Windows, macOS, atau Linux. Bab 2 menjelaskan bagaimana pencarian berubah menjadi jawaban dan apa itu Generative Engine Optimization (GEO). Bab 3 menelusuri apa yang dikerjakan Profitina, dari pemindaian pertama hingga hasil yang dipantau, dan mengajak Anda membangun pemeriksa kesiapan AI sendiri. Bab 4 dan 5 membahas komunitas waktu-nyata dan cara menjaganya tetap aman. Bab 6 dan 7 membahas AI lokal — Mesin Konten, model dua jalur, dan siklus umpan balik tempat manusia memutuskan dan model belajar. Bab 8 membahas model bisnis. Bab 9 membahas arsitektur, keandalan, dan gerbang rilis. Bab 10 mengkaji situs pemasaran sebagai studi kasus GEO tersendiri. Bab 11 membahas privasi dan kepatuhan sejak rancangan menurut undang-undang pelindungan data pribadi Indonesia. Bab 12 menjelaskan PROFIT, koin utilitas di Solana, dengan menyatakan secara terang apa yang sudah berjalan dan apa yang masih rencana. Bab 13 menempatkan Profitina di dalam Profitina Group dan menatap ke depan.

Setiap praktikum sudah dijalankan sebelum keluarannya dicetak. Program-program itu dikumpulkan, bab demi bab, dalam folder `Code` yang menyertai buku ini, dengan README di setiap folder bab yang menjelaskan cara menjalankan setiap program di terminal dan di VS Code, pada Windows, macOS, dan Linux.

Cara membacanya

Berurutan, untuk pertama kali; sesudah itu, setiap bab dapat berdiri sendiri. Ketikkan praktikumnya, jangan hanya membacanya, ubah satu hal, lalu jalankan lagi. Semua soal latihan di akhir tiap bab dilengkapi penyelesaian lengkap — gunakan sesudah usaha Anda sendiri, bukan sebagai penggantinya.

Gambar, tabel, kode, dan persamaan yang bernomor semuanya adalah tautan hidup: pada edisi elektronik, klik "Kode 3.2" atau "[4]" di bagian mana pun dan Anda langsung dibawa ke sana. Daftar Isi serta daftar-daftar di bagian depan buku bekerja dengan cara yang sama.

Tentang penulis

Irfan Subakti menggambarkan dirinya dalam satu baris — *Father | Educator | Entrepreneur | Bridging Tech & Nature: CS Lecturer × Crypto/Digital Business × AI/ML Specialist × Fruit Farmer | Storyteller* — ayah, pendidik, wirausahawan, penjembatan teknologi dan alam: dosen ilmu komputer, pelaku bisnis kripto dan digital, spesialis AI/ML, petani buah, dan pencerita. Setiap bagian dari baris itu ikut masuk ke dalam buku ini. Ia mengajar di Departemen Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS), Surabaya, sejak 1998. Ia membimbing dan menguji tugas akhir mahasiswa sarjana, mengajar MLOps dalam program IBM SkillsBuild, dan membangun produk: ia mendirikan Profitina dan Profitina Group. Jauh dari papan ketik, ia merawat kebun buah dan durian keluarga — di sanalah ia belajar bahwa perangkat lunak yang baik, seperti kebun yang baik, ditumbuhkan dengan sabar, satu bagian yang terawat demi satu bagian. Jiwa penceritanya adalah alasan buku ini memilih kisah ketimbang daftar, setiap kali sebuah kisah bisa mengantarkan maknanya.

Ucapan terima kasih

Kepada para anggota komunitas Profitina, yang pertanyaan, laporan, dan kesabarannya membentuk produk ini lebih dari rencana mana pun; kepada para mahasiswa ITS yang mencoba versi-versi paling awalnya dan berterus terang tentangnya; serta kepada rekan-rekan yang bersama-sama membahas dan mempertajam gagasan-gagasan dalam buku ini. Standar buku ini berutang pada penelitian yang telah terbit tentang mesin penjawab dan pada dokumentasi resmi setiap teknologi yang dipakainya, yang dikutip di setiap tempat penggunaannya.

Surabaya, September 2026

Irfan Subakti · yifana@gmail.com

profitina.com

Bab 1

Asal-Usul: Dari Ruang Kelas Seorang Dosen ke Sebuah Platform

Di akhir bab ini Anda akan mengetahui dari mana Profitina berasal dan mengapa bentuknya seperti sekarang: siapa yang membangunnya, apa yang telah ia ajarkan selama hampir tiga puluh tahun sebelum menulis baris kodenya yang pertama, bagaimana sebuah gagasan menjadi platform yang hidup pada 17 Juli 2026 lalu tumbuh pekan demi pekan, dan gagasan apa saja yang sudah ada sejak awal. Anda akan tahu apakah buku ini ditulis untuk Anda, dan bab mana yang perlu dibaca lebih dahulu bila waktu Anda sempit. Anda juga akan memiliki meja kerja pemrogram yang siap pakai di komputer Anda sendiri, entah Windows, macOS, atau Linux: Visual Studio Code, Java Development Kit, dan Python, yang masing-masing sudah diperiksa, dengan dua program pertama yang sudah ditulis, dijalankan, dan dipahami dalam Java maupun Python. Setiap bab berikutnya ditutup dengan sebuah lab praktik, dan setiap lab mengandaikan bahwa meja kerja Subbab 1.8 sudah berdiri.

1.1 Sebuah Platform dengan Ruang Kelas di Belakangnya

Kebanyakan produk teknologi diperkenalkan dengan legenda pendirian: sebuah garasi, kamar asrama, akhir pekan tanpa tidur, atau ilham mendadak yang dicoret di selembar tisu. Profitina tidak memilikinya, dan bab ini tidak akan mengarangnya. Yang dimilikinya justru kurang dramatis, tetapi jauh lebih berguna bagi pembaca yang ingin belajar: jejak panjang yang terdokumentasi dengan baik, dari sebuah ruang kelas di Surabaya, melewati hampir tiga dasawarsa pengajaran dan penelitian kecerdasan buatan, hingga sebuah aplikasi web produksi yang dapat dibuka siapa saja hari ini di `app.profitina.com`.

Profitina adalah platform Visibilitas AI Indonesia. Disiplin yang dijalankannya punya nama, GEO, singkatan dari *Generative Engine Optimization*, dan Bab 2 menjelaskannya dengan cermat. Dalam satu kalimat: Profitina mengukur seberapa siap sebuah usaha, situs web, peneliti, atau organisasi untuk *ditemukan dan dikutip* oleh mesin jawaban AI seperti ChatGPT, Gemini, dan Perplexity, lalu membantu memperbaiki apa yang ditemukannya [1]. Pengukuran itu dilakukan dari sinyal yang dapat diamati siapa pun di web publik, misalnya apakah teks sebuah halaman dapat diekstraksi dengan bersih, apakah halaman itu memuat data terstruktur dalam kosakata schema.org, dan apakah fondasi teknisnya kokoh. Tagline Indonesianya menyebutnya *aplikasi manajemen & optimalisasi visibilitas AI nomor 1 di Indonesia*, dan semboyan tiga katanya adalah *Bersama · Berkembang · Berbagi* (dalam bahasa Inggris, *Gather · Grow · Give*) [1].

Dua fakta tentang asal-usul Profitina membentuk seluruh isi buku ini.

Fakta pertama: Profitina dibangun oleh seorang dosen yang tidak pernah berhenti mengajar. Orang yang merancang arsitektur Profitina juga mengajar algoritma, pemrograman web, dan pemrograman berorientasi objek kepada mahasiswa sarjana setiap semester. Karena itu produk ini bukan hanya sebuah usaha; ia juga kasus ajar yang ditelaah dari berbagai sudut dalam sebuah pustaka buku ajar, dan sejak 31 Agustus 2026 ia menjadi tempat sebuah kelas berisi sekitar 250 mahasiswa benar-benar bekerja.

Fakta kedua: Profitina dibangun secara terbuka dan dengan cepat. Basis kode yang sekarang dimulai dengan versi 0.11.0 pada 3 Juli 2026. Dua pekan kemudian, pada 17 Juli 2026, platform ini masuk produksi. Pada 24 September 2026, hari bab ini selesai ditulis, versinya sudah 0.14.5 dan whitepaper publiknya sudah mencapai versi kedua puluh empat. Setiap langkah itu bertanggal, dan Subbab 1.4 memaparkannya.

Bab ini menuturkan kisah itu secara berurutan. Subbab 1.2 memperkenalkan pendiri dan perjalanan akademiknya. Subbab 1.3 menunjukkan mengapa ruang kelas dan basis kode, dalam arti yang tepat, adalah tempat yang sama. Subbab 1.4 menelusuri lini masa yang bertanggal. Subbab 1.5 menyebut gagasan-gagasan pendiri yang terus dirujuk oleh bagian lain buku ini. Subbab 1.6 dan 1.7 menjelaskan untuk siapa buku ini dan bagaimana membacanya. Subbab 1.8, yang terpanjang, menyiapkan komputer Anda. Sesudah itu datang subbab penutup yang tetap: Profitina dalam praktik, kesalahpahaman yang sering terjadi, ringkasan, soal latihan, dan penyelesaian lengkapnya.

1.2 Pendiri dan Tiga Dasawarsa Mengajar

1.2.1 Seorang dosen di Surabaya

Profitina didirikan, dan hingga kini dibangun, oleh Ir. M.M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil., IPM, dosen Departemen Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS) di Surabaya, salah satu perguruan tinggi teknik terkemuka di Indonesia. Di bagian lain buku ini ia cukup disebut “pendiri”.

Deretan huruf di sekitar namanya layak diurai sekali, karena setiap bagian mencatat jenis pendidikan yang berbeda. *S.Kom.* (Sarjana Komputer) adalah gelar sarjana ilmu komputer di Indonesia. *M.Sc.Eng.* adalah Master of Science in Engineering, dan *M.Phil.* adalah Master of Philosophy, gelar riset. *Ir.* adalah gelar insinyur profesional yang diperoleh melalui Program Profesi Insinyur di perguruan tinggi, dan *IPM* (Insinyur Profesional Madya) adalah sertifikasi insinyur profesional jenjang madya. Bersama-sama, gelar itu menggambarkan karier yang dibangun di sekitar komputasi dan kecerdasan buatan, yang diakui baik secara akademik maupun profesional.

1.2.2 Tempat pengajaran bermula

Perkuliahan paling awal yang terdokumentasi diampu pendiri di ITS berlangsung dari September 1997 hingga Januari 1998: sebuah bagian tentang *general inference engine* (mesin inferensi umum) dalam matakuliah sistem berbasis pengetahuan. *Mesin inferensi* adalah bagian dari sistem pakar yang mengambil sekumpulan aturan, seperti “jika demamnya tinggi dan ruamnya merah, curigai campak”, dan sekumpulan fakta, lalu menurunkan fakta baru darinya, satu aturan demi satu aturan, sampai tiba pada kesimpulan. Ini salah satu gagasan tertua dalam kecerdasan buatan terapan, dan dalam wujud modernnya masih menjadi jantung sistem yang harus *menjelaskan* keputusannya. Tugas akhir sarjananya sendiri, pada 1998–1999, adalah simulator Flexible Manufacturing Systems, sebuah program yang memodelkan mesin, antrean, dan jadwal pabrik sehingga orang dapat bereksperimen tanpa menghentikan lini produksi sungguhan.

Menarik garis lurus dari awal mula itu ke Profitina memang menggoda, dan sebagian garis itu memang berlaku. Pemindaian kesiapan yang menghasilkan skor *beserta alasannya* adalah, secara semangat, keturunan sistem pakar yang harus memperlihatkan aturannya. Platform yang proses rilis, penjadwal pindai ulang, dan umpan komunitasnya semuanya berupa antrean dan jadwal

adalah sepupu simulator manufaktur itu. Buku ini tidak akan melebih-lebihkan kaitan tersebut, tetapi akan menunjukkannya di tempat kaitan itu benar-benar membantu Anda memahami sebuah pilihan rancangan.

1.2.3 Riset di beberapa negara dan beberapa cabang AI

Antara 2007 dan 2010 pendiri menjalani masa riset di luar negeri, di National Chiao Tung University (NCTU) di Hsinchu, Taiwan, pada 2007–2009, dan di Comenius University di Bratislava, Slovakia, pada 2009–2010. Publikasinya selama bertahun-tahun mencakup sistem pakar, ontologi, logika fuzzy, algoritma genetika, jaringan saraf tiruan, serta pengolahan dan temu kembali citra. Bila dihimpun, daftar itu hampir menyerupai daftar isi kecerdasan buatan klasik: bernalar dengan aturan, merepresentasikan pengetahuan sebagai konsep terstruktur, bernalar dengan derajat kebenaran, mencari dengan meniru evolusi, belajar dari contoh, dan memahami gambar.

Hal ini penting untuk cara membaca bagian lain buku ini. Profitina bukan kisah seorang pebisnis yang menyewa keahlian AI untuk menghiasi sebuah ide produk. Justru sebaliknya: karier yang dihabiskan untuk membangun dan mengajarkan sistem AI dari fondasinya, diterapkan pada produk nyata yang juga dioperasikan oleh orang yang sama. Ketika Bab 6 menjelaskan mengapa Profitina menjalankan model bahasanya secara *lokal*, di perangkat kerasnya sendiri, alih-alih menyewa layanan AI komersial, Anda akan melihat bahwa keputusan itu diambil oleh seseorang yang terbiasa memahami sebuah model, bukan sekadar memanggilnya.

> **WAWASAN KUNCI:** Keistimewaan Profitina bukan bahwa ia memakai AI; hampir setiap produk baru melakukannya. Keistimewaannya adalah bahwa perancang platform ini telah mengajarkan fondasi AI, algoritma, dan konstruksi perangkat lunak selama sekitar tiga dasawarsa, dan bahwa produknya sengaja dibangun agar dapat *diajarkan*: gagasannya dijelaskan secara terbuka, angkanya bertanggal, dan lab praktiknya dapat dibangun ulang oleh seorang mahasiswa hanya dengan sebuah laptop.

1.3 Ruang Kelas dan Basis Kode Adalah Tempat yang Sama

1.3.1 Tiga matakuliah yang menjadi bahan sebuah platform

Bukti paling nyata tentang kaitan antara pengajaran pendiri dan platform ini sederhana dan dapat diperiksa. Pada tahun akademik 2025/2026 ia mengajar tiga matakuliah sarjana di ITS: EF234405 *Perancangan dan Analisis Algoritma* (PAA, dalam bahasa Inggris *Design and Analysis of Algorithms*, DAA), EF234301 *Pemrograman Web* (WebPro), dan EF234302 *Pemrograman Berorientasi Objek* (PBO, dalam bahasa Inggris OOP). Ketiganya bukan matakuliah pilihan yang kebetulan berkaitan. Masing-masing adalah matakuliah tentang berpikir dalam algoritma dan biayanya, matakuliah tentang membangun aplikasi web, dan matakuliah tentang menyusun perangkat lunak menjadi objek-objek yang bekerja sama. Platform seperti Profitina terbuat dari tepat tiga bahan itu: biaya kueri dan jadwalnya adalah pertanyaan PAA, aplikasinya dan widget komunitas yang tertanam adalah artefak Pemrograman Web, dan banyaknya pemeriksaan independen dalam pemindai, yang masing-masing melaporkan hasilnya sendiri, adalah sebuah rancangan berorientasi objek.

Sebuah rincian kecil menunjukkan bahwa ini adalah catatan yang terus dipelihara, bukan sekadar sebaris dalam riwayat hidup. Kode matakuliah berubah dari skema lama yang diawali “IF” (misalnya IF184301 pada 2022/2023) ke skema sekarang yang diawali “EF” (misalnya EF234405 pada 2025/2026) sebagai bagian dari revisi kurikulum ITS.

1.3.2 Sebuah pustaka buku ajar di sekitar satu sistem nyata

Buku ini termasuk dalam sebuah seri, Profitina Course Library. Buku-bukunya antara lain *Perancangan dan Analisis Algoritma*, *Pemrograman Web*, *Pemrograman Berorientasi Objek*, *Struktur Data*, *Pemrograman Berbasis Kerangka Kerja*, dan *Dasar Pemrograman*, ditambah buku tentang pemrograman kompetitif, sistem cerdas, metode numerik, dan interaksi manusia dan komputer. Sebagian besar memakai Profitina sebagai kasus nyata yang berjalan. Beberapa contoh memperlihatkan rentangnya: pembatasan laju login adalah algoritma *sliding window* (jendela geser) dalam buku algoritma; hashing kata sandi dengan Argon2id adalah contoh fungsi yang sengaja dibuat *memory-hard* (berat memori); emisi koin PROFIT yang meluruh adalah relasi rekurens peluruhan eksponensial dalam buku metode numerik; dan papan peringkat platform adalah latihan praktis pengurutan dan pemeringkatan.

Buku Pemrograman Web dan Pemrograman Berbasis Kerangka Kerja melangkah lebih jauh dan membangun sebuah aplikasi kecil yang lengkap, **Profitina Laundry**, sebagai studi kasusnya. Profitina Laundry adalah usaha binatu fiktif di Surabaya dengan tiga gerai, basis data pelanggan, layanan, dan tagihan, serta versi dalam beberapa bahasa dan kerangka kerja. Ia tidak berhubungan dengan kode maupun data produk yang sebenarnya; ia ada agar setiap latihan praktik memiliki usaha yang ramah dan konkret untuk dikerjakan. Anda juga akan bertemu dengannya di buku ini: di Bab 2 kita menulis data terstruktur dan `robots.txt`-nya, dan lab-lab berikutnya memindai, menjadwalkan, dan melindunginya.

1.3.3 Platform di sebuah kelas nyata

Pada 31 Agustus 2026 hubungan antara ruang kelas dan platform tidak lagi sekadar soal contoh. Kelas pertama ITS dengan sekitar 250 mahasiswa mulai memakai komunitas Profitina untuk pekerjaan perkuliahan mereka, yang diatur dalam kanal-kanal kelas. Beberapa fitur yang kini menjadi bagian produk dibangun tepat untuk situasi itu, dalam pekan-pekan sebelum dan sesudahnya:

- **Moderator yang lingkupnya kanal** (13 Agustus 2026). Seorang moderator, misalnya dosen atau koordinator mahasiswa, dapat menyembunyikan unggahan, menyematkan pengumuman, atau membisukan peserta untuk sementara *hanya di kanalnya sendiri*. Moderator tidak pernah menyentuh saldo PROFIT atau data pribadi siapa pun, setiap tindakan dicatat, dan orang yang terdampak diberi tahu.
- **Handle netral** (14 Agustus 2026). Akun baru menerima handle acak, huruf `u` diikuti delapan digit heksadesimal, yang dapat diubah pengguna. Handle diperiksa agar mahasiswa tidak tanpa sengaja memublikasikan nomor induk mahasiswanya sebagai nama publik.
- **Hak swalayan** (19 Agustus 2026). Panel *Akun & Data Pribadi* memungkinkan setiap pengguna mengekspor datanya, mengubah e-mail atau kata sandinya, atau menghapus akunnya secara permanen, tanpa harus meminta kepada siapa pun. Bab 11 menuturkan kisah yang mendidik tentang mengapa panel ini mendesak.

Platform yang dipakai sebuah kelas nyata menerima umpan balik nyata, dengan sangat cepat. Satu keputusan dari masa itu adalah contoh yang baik tentang disiplin yang berkali-kali dirujuk buku ini. Pada pertengahan September 2026 tim mengevaluasi gagasan untuk memberikan konten buatan AI secara gratis kepada seluruh 250 mahasiswa. Gagasan itu diukur dan, berdasarkan bukti, *dibekukan* pada 15 September 2026. Gagasan yang menarik dikesampingkan karena hasil pengukuran berkata demikian. Bab 7 menjelaskan kaidah umumnya: manusia yang memutuskan, dan mereka memutuskan berdasarkan bukti.

1.4 Dari Gagasan ke Produksi: Lini Masa Bertanggal

1.4.1 Sebelum Juli 2026

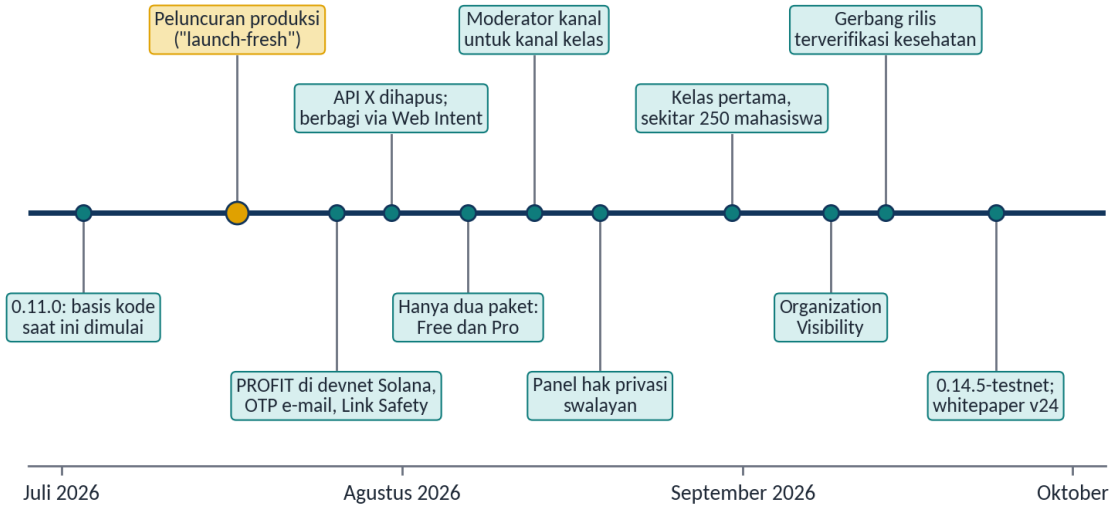
Gagasan Profitina lebih tua daripada basis kodenya yang sekarang. Sebelum Juli 2026 proyek ini sudah melewati dua belas versi whitepaper, beberapa versi tema WordPress khusus untuk situs pemasaran, dan metodologi awal bernama "3P". Produk pada waktu itu juga memiliki tingkat langganan ketiga, *Agency*, untuk konsultan yang mengelola banyak klien; seperti ditunjukkan lini masa, tingkat ini kemudian dihapus. Sejarah yang lebih awal berada di luar basis kode yang dibahas buku ini, sehingga kisah bertanggal di bawah dimulai pada 3 Juli 2026.

1.4.2 Dua belas pekan, pekan demi pekan

Tabel 1.1 memuat tonggak-tonggak dari versi pertama basis kode sekarang hingga hari bab ini ditulis, dan **Gambar 1.1** menggambarkan periode yang sama pada sumbu waktu. Bacalah tabel itu perlahan: ia sekaligus peta buku ini, karena hampir setiap barisnya menjadi pokok bahasan salah satu bab berikutnya.

Tabel 1.1 Tonggak bertanggal Profitina dari 3 Juli hingga 24 September 2026, beserta bab buku ini yang membahas masing-masing. Baris terakhir adalah rencana, bukan fakta.

Tanggal (2026)	Tonggak	Bab
3 Jul	Versi 0.11.0: pindai GEO, konsep PROFIT, tema, dan whitepaper v12 dibawa ke basis kode sekarang	1
Jul	Fase 1 (akun, pindai) dan Fase 1.5 (pelindungan data pribadi: Argon2id, persetujuan, pembatasan laju, hak subjek data, halaman hukum)	3, 11
17 Jul	**Peluncuran produksi.** Basis data produksi dikosongkan tepat sebelum pengguna publik pertama ("launch-fresh"), sehingga tidak ada data uji yang terbawa	9
akhir Jul	Ekonomi PROFIT: verifikasi e-mail dengan kode sekali pakai, pemeriksaan fair play, Link Safety, oracle PROFIT–rupiah dan pembelian lewat QRIS, tema terang dan gelap, antarmuka ID/EN penuh, PROFIT sebagai token di devnet Solana	5, 8, 12
29 Jul	Dasbor historis, penjadwal pindai ulang, dan peringatan; versi 0.12.0	9
31 Jul	API X (Twitter) dihapus; berbagi beralih ke Web Intent X, diunggah pengguna dari akunnya sendiri; bilah kemajuan yang rinci	3
1–5 Agu	Dua jalur AI lokal: jalur kualitas di CPU dan jalur cepat di GPU	6
6–7 Agu	Hanya dua paket, Free dan Pro: tingkat Agency dihapus; tanpa iklan, selamanya	8
12 Agu	Unggahan dapat disunting selama 15 menit, dengan imbalan dihitung ulang sepenuhnya	4
13 Agu	Moderator berlingkup kanal; hibah administrator dengan riwayat publik; peran basis data terpisah untuk pengembangan dan produksi	4, 9
19 Agu	Panel hak privasi swalayan	11
20 Agu	Program referal satu tingkat, halaman profil dengan persetujuan per butir, pesan langsung	4, 8
30 Agu	PROFIT dapat dibelanjakan untuk layanan nyata; satu rumah tunggal bagi Kebijakan Privasi	11, 12
31 Agu	Kelas ITS pertama dengan sekitar 250 mahasiswa; Researcher Visibility Report (OpenAlex)	3, 13
1 Sep	*One Text, Many Platforms*	3
9 Sep	Organization Visibility	3, 13
14 Sep	Proyek bergabung ke payung Profitina Group; gerbang rilis terverifikasi kesehatan	9, 13
15–16 Sep	Dasbor sebagai "kokpit", unggahan instan, pusat komando administrator	4, 9
23–24 Sep	Perpindahan kanal instan (satu halaman umpan dari 868 ms menjadi 11 ms); versi 0.14.3 sampai 0.14.5-testnet; whitepaper v24	9
K4 (rencana)	Fase 3: PROFIT di jaringan utama Solana dan bursa nonkustodian, bergantung pada perizinan dan audit	12



Gambar 1.1 Dua belas pekan pertama basis kode sekarang pada sumbu waktu. Titik kuning adalah peluncuran produksi 17 Juli 2026.

Tiga pola menonjol dalam **Gambar 1.1**.

Pola pertama adalah **kecepatan**. Antara 3 Juli dan 24 September 2026 basis kode menerima 391 *commit*, pada 35 hari yang berbeda, dan bergerak dari versi 0.11 ke 0.14. Kemampuan baru sampai ke pengguna kira-kira setiap pekan.

Pola kedua adalah **pengurangan**. Beberapa tonggak justru *menghapus* sesuatu: API X, tingkat Agency, iklan. Produk yang masih muda tergoda untuk menjanjikan segalanya. Lini masa Profitina menunjukkan kebiasaan sebaliknya: ketika sebuah fitur tidak dapat diverifikasi, tidak dapat dipertahankan, atau mengalihkan perhatian dari tugas utama, fitur itu dikeluarkan, dan alasannya dituliskan. Bab 8 mengutip aturan internal yang lahir dari kebiasaan ini: *jangan pernah menuliskan janji tentang fitur yang belum dibangun di samping harga berbayar.*

Pola ketiga adalah **memberi tanggal pada segalanya**. Setiap baris **Tabel 1.1** bertanggal karena platform ini menyimpan catatan tertulis untuk setiap perubahan, dan whitepaper-nya diberi versi dengan cara yang sama. Buku ini menganut disiplin yang sama. Setiap angka yang dapat berubah, seperti versi, jumlah tes, atau statistik, dicetak bersama tanggal ketika angka itu benar, biasanya "pada saat buku ini disusun (24 September 2026)". Ketika Anda membaca ini kelak, sebagian angka itu sudah bergeser. Itu wajar, dan tanggalnya memberi tahu seberapa jauh dunia telah bergerak sejak itu.

1.4.3 Menghitung hari: pandangan pertama pada kode

Sudah berapa lama Profitina hidup? Pada 24 September 2026 jawabannya 69 hari. Menghitung hari dengan tangan melintasi bulan yang panjangnya berbeda-beda adalah jenis tugas kecil yang rawan salah dan lebih baik dikerjakan program daripada manusia, sehingga cocok menjadi program kedua di Subbab 1.8.9: program itu membaca tonggak-tonggak **Tabel 1.1** dari sebuah berkas dan mencetak jarak masing-masing dari peluncuran. Jika d adalah banyaknya hari, banyaknya pekan utuh w dan sisa hari r adalah

$$w = \left\lfloor \frac{d}{7} \right\rfloor, \qquad r = d - 7w \tag{1.1}$$

dengan $\lfloor x \rfloor$, dibaca "lantai dari x", yaitu bilangan bulat terbesar yang tidak lebih besar daripada x. Untuk $d = 69$: $69/7 = 9.857\ldots$, jadi $w = 9$ dan $r = 69 - 63 = 6$. Profitina telah hidup selama 9

pekan 6 hari. Persamaan (1.1) adalah apa yang dihitung operator `//` dan `%` di Python, serta `/` dan `%` pada bilangan bulat di Java, untuk setiap $d \geq 0$.

1.5 Gagasan-Gagasan Pendiri

1.5.1 Misi

Profitina menyatakan misinya dalam satu kalimat, yang aslinya berbahasa Inggris:

> **MISI.** "Democratize AI visibility for every Indonesian business and creator through a transparent, inclusive, self-reliant platform — so anyone (from a warung to a SaaS startup to a solo creator) can be found and quoted by AI, without paid APIs or gatekeepers." Artinya: mendemokratisasi visibilitas AI bagi setiap usaha dan kreator Indonesia melalui platform yang transparan, inklusif, dan mandiri, agar siapa pun (dari warung hingga startup SaaS hingga kreator perorangan) dapat ditemukan dan dikutip oleh AI, tanpa API berbayar atau penjaga gerbang.

Tiga kata di dalamnya memikul bobot. *Transparan* berarti metodenya diperlihatkan: pemindaian tidak hanya memberi skor antara 0 dan 100, tetapi juga alasannya dan daftar perbaikan yang diprioritaskan, dan platform menyatakan terus terang bahwa hasilnya adalah "perkiraan metodologis, bukan jaminan atas keluaran mesin AI pihak ketiga yang sebenarnya" [1], [2]. *Inklusif* berarti produk yang sama melayani *warung*, startup perangkat lunak, dan seorang kreator perorangan, dengan metode yang sama. *Mandiri* berarti AI platform berjalan di infrastruktur yang dioperasikannya sendiri dan tidak bergantung pada layanan AI sewaan pihak ketiga.

1.5.2 Visibilitas dan partisipasi dalam satu sistem

Gagasan pendiri yang kedua kurang kentara dan lebih orisinal. Selama dua dasawarsa, platform sosial menghimpun nilai yang sangat besar dari partisipasi orang biasa, seperti unggahan, balasan, rekomendasi, dan ulasan, dan membalasnya sebagian besar dengan tanda suka dan pengikut. Tesis pendirian Profitina, yang diungkapkan melalui rancangannya dan bukan melalui slogan, adalah bahwa kontribusi yang berguna dapat *diukur*, dan bahwa ukuran itu dapat dikaitkan dengan imbalan yang nyata, walaupun kecil.

Di dalam produk, gagasan ini mengambil tiga wujud. Sebuah komunitas, dengan kanal, utas, dan pesan langsung, hidup di dalam aplikasi yang sama dengan alat visibilitas (Bab 4). Kontribusi yang membantu, seperti umpan balik yang memperbaiki pemindaian, diakui dan dicantumkan, dan setiap keputusan tentang apa yang dirilis diambil oleh manusia (Bab 7). Dan sebuah koin utilitas, PROFIT, diperoleh untuk kontribusi yang berguna dan dapat dibelanjakan untuk layanan platform itu sendiri (Bab 12). Lingkungan akademik pendiri mengajukan pertanyaan yang bertetangga pada waktu yang sama: tugas akhir sarjana yang ia bimbing pada 2026 meneliti bagaimana pemegang besar memengaruhi volatilitas token di Ethereum, bagaimana token "soulbound" yang tidak dapat dipindahtangankan dapat merepresentasikan identitas profesional, dan bagaimana strategi *trend-following* dan *mean-reversion* berperilaku di pasar spot Solana. Tidak ada dokumen yang menyatakan bahwa tugas akhir itu membentuk rancangan PROFIT, dan buku ini pun tidak menyatakannya; tugas akhir itu menunjukkan bahwa ruang kelas dan produk sedang menggarap masalah yang bertetangga pada bulan-bulan yang sama.

> **PERINGATAN: YANG SUDAH BERJALAN DAN YANG MASIH RENCANA.** PROFIT saat ini ada sebagai token di *devnet* Solana, jaringan uji publik yang tokennya tidak bernilai uang; kontraknya dapat diperiksa siapa saja di penjelajah blok. Membeli PROFIT dengan rupiah

> melalui QRIS, memperolehnya, dan membelanjakannya di dalam platform sudah berjalan. Menjual PROFIT dinonaktifkan, dan peluncuran di jaringan utama Solana adalah rencana Fase 3 yang bergantung pada perizinan dan audit. Tidak ada isi buku ini yang merupakan nasihat investasi atau janji tentang harga apa pun.

1.5.3 AI lokal, karena pilihan

Gagasan pendiri yang ketiga bersifat arsitektural. Setiap fitur AI di Profitina berjalan pada model bahasa yang dieksekusi di mesin milik Profitina sendiri, melalui *runtime* bernama Ollama, dalam dua jalur: *jalur kualitas* di prosesor dengan model Qwen2.5 7B untuk teks panjang seperti draf konten, dan *jalur cepat* di prosesor grafis dengan model yang sangat kecil, qwen3:1.7b, untuk tugas klasifikasi pendek [1]. Jalur kode yang dahulu memanggil layanan AI terhosting telah dihapus seluruhnya. Dua akibat mengikutinya. Data pengguna tidak pernah meninggalkan server untuk pemrosesan AI, hal yang penting menurut undang-undang pelindungan data pribadi Indonesia (Bab 11). Dan platform dapat jujur tentang batasnya: ketika tidak ada model yang tersedia, aplikasi *menyatakannya* alih-alih diam-diam menyajikan templat seolah-olah ditulis oleh model.

Sebuah prinsip yang berkaitan mudah disalahpahami, sehingga perlu dinyatakan sejak awal. Profitina **tidak** mengukur visibilitas sebuah situs dengan mengajukan pertanyaan tentang situs itu kepada ChatGPT, Gemini, atau Perplexity. Mesin-mesin itu adalah *sasaran* yang kesiapannya diperkirakan, bukan komponen yang menjadi sandaran platform. Alasannya adalah kemandirian, privasi, dan aturan internal yang melarang API pihak ketiga berbayar. Bab 2 menjelaskan apa yang diukur sebagai gantinya.

1.5.4 Diuji sebelum dirilis

Gagasan pendiri yang keempat adalah disiplin rilis. Versi baru Profitina sampai ke pengguna hanya melalui gerbang rilis yang lebih dahulu menjalankan seluruh rangkaian tes otomatis. Pada saat buku ini disusun (24 September 2026, versi 0.14.5) rangkaian itu mencatat **1.577 tes lulus dan 1 dilewati** pada jalur uji SQLite, dan subkumpulan prioritas berisi **440 tes** lulus pada PostgreSQL 16, basis data produksi. Gerbang itu baru menyatakan berhasil setelah server yang baru dipasang menjawab titik akhir kesehatannya dengan nomor versi yang baru. Bab 9 menuturkan bagaimana gerbang itu menjadi seketat ini, termasuk hari ketika sebuah skrip rilis melaporkan keberhasilan untuk versi yang sesungguhnya belum terpasang.

1.5.5 Indonesia lebih dahulu

Gagasan pendiri yang terakhir bersifat geografis. Profitina dibangun untuk Indonesia lebih dahulu dengan cara yang konkret, bukan sebagai slogan: antarmukanya sepenuhnya dwibahasa, Indonesia dan Inggris; penanganan data pribadinya mengikuti Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi (UU PDP); dan pembayarannya berjalan di atas QRIS, standar nasional Indonesia untuk pembayaran dengan kode QR. Ekspansi ke kawasan adalah rencana Fase 4 yang baru datang setelah kepemimpinan di dalam negeri tercapai [1]. Pasar dalam negeri itu besar: menurut survei 2026 Asosiasi Penyelenggara Jasa Internet Indonesia (APJII), Indonesia memiliki sekitar 235,3 juta pengguna internet dari populasi sekitar 287,3 juta, penetrasi 81,72 persen [3].

1.6 Untuk Siapa Buku Ini

Buku ini ditulis agar dapat dibaca sejak halaman pertama oleh seseorang yang belum tahu apa pun tentang GEO, crawler web, model bahasa, atau blockchain, dan agar pembaca itu pada akhirnya mampu menjelaskan masing-masing serta membangun contoh kecil yang berjalan untuk masing-masing. Buku ini juga ditulis bagi pembaca yang sudah mengenal sebagian hal itu dan ingin melihat bagaimana semuanya menyatu dalam satu sistem nyata. **Tabel 1.2** menyebut lima jenis pembaca dan menyarankan tempat masing-masing memulai.

Tabel 1.2 Lima jenis pembaca, apa yang akan ditemukan masing-masing dalam buku ini, dan bab mana yang sebaiknya dibaca lebih dahulu.

Jika Anda…	Anda akan menemukan…	Mulailah dari
Mahasiswa informatika atau bidang terkait	Sistem produksi nyata yang ditelaah dari ujung ke ujung, dengan setiap lab dalam Python dan Java, serta soal latihan beserta penyelesaian lengkap	1, 2, 3, lalu urutan bebas
Pengembang perangkat lunak atau insinyur	Arsitektur, keandalan, AI lokal, umpan waktu nyata, pengujian sebagai gerbang rilis, dan alasan di balik setiap pilihan	2, 3, 6, 9
Pemilik usaha atau kreator	Apa itu GEO, mengapa jawaban AI mengubah cara pelanggan menemukan Anda, dan apa yang dapat Anda lakukan pekan ini	2, 3, 10
Mitra atau pihak yang sedang menilai Profitina	Apa yang sudah berjalan dan apa yang masih rencana, model bisnis, kepatuhan, dan koin PROFIT yang diuraikan secara faktual	2, 8, 11, 12
Dosen	Studi kasus yang menghubungkan algoritma, pemrograman web, rancangan berorientasi objek, keamanan, dan hukum privasi	1, 3, 9, 11

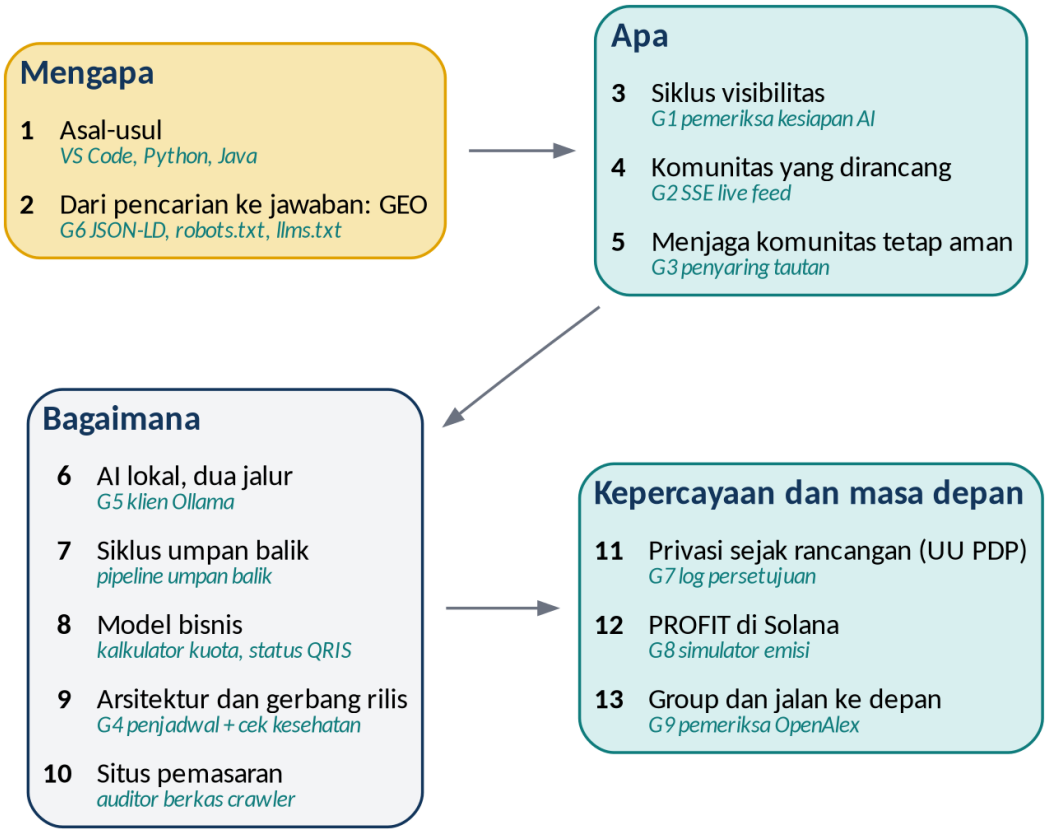
Baris mana pun yang menggambarkan Anda, bacalah Bab 1 dan 2 secara utuh. Keduanya adalah fondasi yang diandaikan oleh setiap bab berikutnya: Bab 1 memberi Anda kisah dan meja kerja yang siap pakai, Bab 2 memberi kosakata dan alasan-alasannya.

Dua kelompok patut disapa secara langsung. Jika Anda **pemilik usaha** tanpa latar belakang pemrograman, Anda dapat membaca bagian kisah di setiap bab dan melewati kodenya; teks di sekitar setiap kode selalu menjelaskan dengan kata-kata sederhana apa yang dikerjakan program itu dan apa arti keluarannya. Jika Anda mempertimbangkan Profitina sebagai **mitra atau investor**, buku ini sengaja bukan dokumen penjualan. Buku ini menyatakan terus terang di mana platform berdiri hari ini dan ke mana peta jalannya mengarah, termasuk bahwa koin PROFIT saat ini tidak dapat diperdagangkan dan bahwa peluncurannya di jaringan utama bergantung pada regulator dan audit. Tidak ada isi buku ini yang merupakan nasihat keuangan.

1.7 Cara Membaca Buku Ini

1.7.1 Bentuk buku

Buku ini terdiri atas tiga belas bab, yang dikelompokkan menurut pertanyaan yang dijawabnya. **Gambar 1.2** memperlihatkan kelompok-kelompok itu: *mengapa* Profitina ada (Bab 1 dan 2), *apa* yang dikerjakannya (Bab 3 sampai 5), *bagaimana* ia bekerja (Bab 6 sampai 10), dan mengapa ia dapat *dipercaya*, beserta ke mana ia menuju (Bab 11 sampai 13). Di bawah setiap judul bab, gambar itu menyebut lab praktiknya.



Gambar 1.2 Tiga belas bab dalam empat kelompok, dengan lab praktik setiap bab dalam huruf miring. Lab bertanda G1 sampai G9 adalah program bernomor buku ini.

1.7.2 Setiap bab memiliki bagian yang sama

Setiap bab dibuka dengan satu paragraf yang menyatakan apa yang akan mampu Anda lakukan di akhir bab, dan ditutup dengan lima subbab yang sama: *Profitina dalam Praktik* menghubungkan bab dengan produk yang berjalan; *Kesalahpahaman yang Sering Terjadi* menghimpun kekeliruan yang paling sering dibuat pembaca terhadap gagasan bab; *Ringkasan Bab* menegaskan kembali intinya; *Soal Latihan* memberi Anda kesempatan berlatih; dan *Penyelesaian Soal Latihan* mengerjakan setiap soal secara lengkap. *Daftar Pustaka* sebuah bab memuat sumber-sumbernya, dinomori menurut urutan kemunculan pertamanya dalam teks, seperti ini [1].

1.7.3 Program, keluaran, dan folder kode

Setiap lab ditulis dua kali, dalam **Python** dan dalam **Java**, dengan perilaku yang sama dan, sedapat mungkin, keluaran yang sama persis sampai ke tiap byte, sehingga Anda dapat membaca bahasa yang Anda kuasai dan mempelajari bahasa lainnya dengan membandingkan. Kode hanya memakai pustaka standar masing-masing bahasa, sehingga tidak ada tambahan yang perlu dipasang. Setiap kode adalah program lengkap, dicetak persis seperti di folder kode buku (hanya komentarnya yang diterjemahkan dalam edisi ini), dan setiap blok keluaran di bawahnya adalah apa yang benar-benar dicetak program itu ketika dijalankan untuk buku ini. Folder kode memiliki satu subfolder per bab, `Chapter01`, `Chapter02`, dan seterusnya, masing-masing dengan subfolder `python` dan `java` serta `README.md` yang mengulang petunjuk menjalankan untuk Windows, macOS, dan Linux.

Program-program itu dijalankan untuk buku ini di Linux dengan Python 3.11 dan JDK 21, dan ditulis untuk rilis terkini, Python 3.14 dan JDK 25. Bila sebuah program mencetak sesuatu yang

bergantung pada mesin Anda, seperti nama folder atau versi Java Anda, teks akan menyebutkannya.

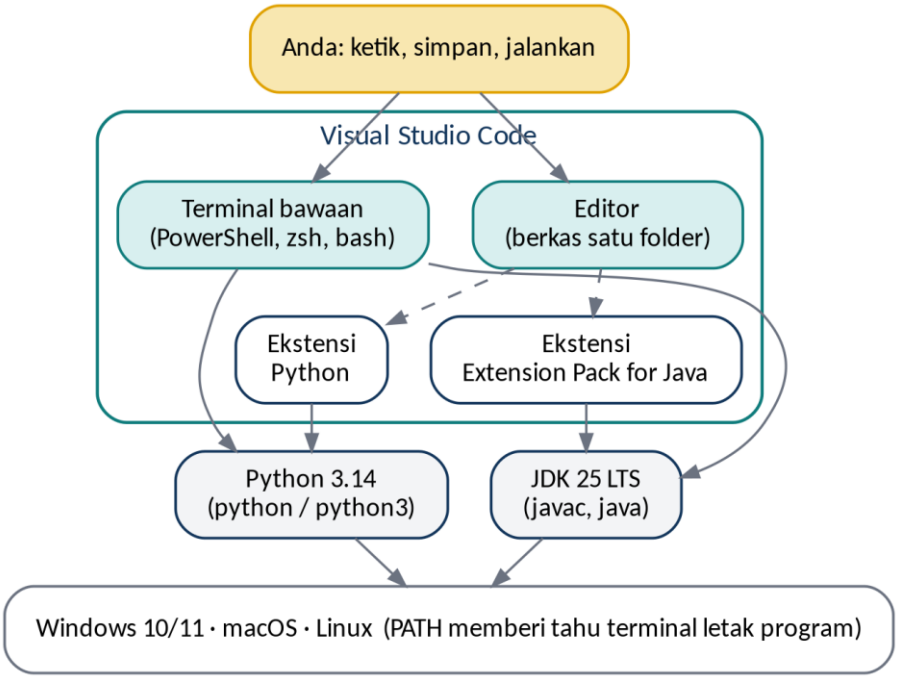
1.7.4 Kotak, label, dan tanggal

Empat jenis kotak muncul dalam teks. Kotak **definisi** memperkenalkan istilah secara tepat; kotak **wawasan kunci** menyatakan gagasan yang layak diingat; kotak **peringatan** menandai jebakan, dan juga setiap tempat ketika perbedaan antara *sudah berjalan* dan *masih rencana* penting; dan kotak **contoh** mengerjakan sebuah kasus atau mengutip pernyataan kata demi kata. Matematika ditulis dengan notasi baku dan setiap simbol dijelaskan saat pertama kali muncul; rumus-rumus yang penting diberi nomor, seperti Persamaan (1.1).

Profitina berubah setiap pekan. Ketika buku ini menyebut versi, jumlah, atau harga, tanggalnya ikut disebut. Ketika buku ini menguraikan sesuatu yang direncanakan, hal itu dinyatakan dalam kalimat yang sama. Dan ketika sebuah angka berasal dari luar Profitina, seperti survei atau statistik industri, daftar pustaka menyebut siapa yang mengukurnya dan kapan.

1.8 Menyiapkan VS Code untuk Python dan Java di Windows, macOS, dan Linux

Semua isi buku ini berjalan di laptop biasa, dan tidak ada yang perlu dibeli. Subbab ini memasang seluruh meja kerja sekali saja dengan cermat: sebuah editor, Java Development Kit, dan Python. Kemudian subbab ini membuktikan bahwa setiap bagiannya bekerja dengan menjalankan program pertama yang sama dalam kedua bahasa. **Gambar 1.3** memperlihatkan bagaimana bagian-bagian itu saling terhubung.



Gambar 1.3 Meja kerja yang dipakai di seluruh buku. Anda bekerja di editor dan terminal VS Code; dua ekstensi mengajari VS Code tentang Python dan Java; terminal menemukan Python dan JDK melalui PATH sistem operasi.

1.8.1 Tiga istilah lebih dahulu

Terminal adalah jendela tempat Anda mengetik perintah dan membaca jawabannya. Di Windows terminal itu adalah **PowerShell** (atau Command Prompt yang lebih lama); di macOS adalah aplikasi **Terminal**, yang menjalankan *shell* bernama `zsh`; di Linux juga disebut Terminal dan biasanya menjalankan `bash`. VS Code memiliki terminal bawaan, yang dibuka dengan **Terminal → New Terminal**. Dalam buku ini, baris yang harus diketik di terminal dicetak dengan huruf berlebar tetap, dan Anda menekan Enter di ujungnya.

Manajer paket memasang perangkat lunak dengan perintah alih-alih dengan mengeklik penginstal: `winget` di Windows 10 dan 11, **Homebrew** (perintah `brew`, dari `https://brew.sh`) di macOS, dan `apt` di Ubuntu dan Debian. Setiap langkah di bawah memberikan perintah manajer paket sekaligus alternatif grafisnya; keduanya sama baiknya.

PATH adalah daftar folder tempat terminal mencari program. Ketika Anda mengetik `javac`, terminal menelusuri setiap folder dalam PATH, berurutan, untuk mencari program bernama `javac` (di Windows, `javac.exe`). Pesan "command not found" atau "is not recognized as an internal or external command" hampir selalu berarti salah satu dari dua hal: programnya terpasang tetapi foldernya tidak ada dalam PATH, atau terminal dibuka *sebelum* folder itu ditambahkan. Membuka terminal **baru** adalah obat pertama yang perlu dicoba.

> **DEFINISI.** *PATH* adalah variabel lingkungan yang menyimpan daftar folder, dipisahkan oleh `;` di Windows dan oleh `:` di macOS dan Linux. Untuk menjalankan perintah yang disebut dengan namanya, terminal mencoba setiap folder dalam daftar secara bergiliran dan menjalankan program pertama dengan nama itu yang ditemukannya.

1.8.2 Visual Studio Code

Visual Studio Code, singkatnya VS Code, adalah editor gratis dari Microsoft yang tampil dan bekerja sama di ketiga sistem dan menjadi lingkungan lengkap untuk sebuah bahasa melalui **ekstensi**. Editor ini diperbarui setiap bulan; rilis terbaru pada saat buku ini disusun adalah versi 1.139, yang dirilis pada 23 September 2026 [4]. Versi terkini mana pun sudah memadai. **Tabel 1.3** memberikan cara pemasangannya.

Tabel 1.3 Memasang VS Code di setiap sistem operasi.

	Windows 10/11	MacOS	Linux (Ubuntu/Debian)
Pasang	Penginstal dari `https://code.visualstudio.com`; centang **Add to PATH**. Atau `winget install Microsoft.VisualStudioCode`	Unduh, ekstrak, lalu seret *Visual Studio Code* ke **Applications**. Atau `brew install --cask visual-studio-code`	Paket `.deb` dari situs web, lalu `sudo apt install ./code_*.deb`. Atau `sudo snap install code --classic`
Agar `code` dapat dipakai di terminal	Sudah dilakukan penginstal	Di VS Code: **Cmd+Shift+P**, lalu *Shell Command: Install 'code' command in PATH*	Sudah dilakukan paketnya
Periksa	`code --version`	`code --version`	`code --version`

1.8.3 Java: JDK 25

Program Java dikompilasi dengan **JDK**, Java Development Kit, yang berisi kompilator `javac` dan peluncur `java`. Pasanglah **JDK 25**, rilis dukungan jangka panjang (*long-term support*, LTS)

September 2025, yang menerima pembaruan selama bertahun-tahun. JDK 27, yang dirilis pada 15 September 2026, adalah rilis fitur terbaru, tetapi hanya didukung sampai rilis fitur berikutnya menggantikannya pada Maret 2027 [5], [6]. Untuk sebuah buku, rilis LTS adalah pilihan yang tenang. Build **Eclipse Temurin** dari proyek Adoptium gratis dan identik di ketiga sistem [7]. Program-program buku ini memerlukan Java 17 atau yang lebih baru dan dikompilasi di sini dengan JDK 21. **Tabel 1.4** memberikan cara pemasangannya.

Tabel 1.4 Memasang JDK 25 di setiap sistem operasi.

	Windows 10/11	MacOS	Linux (Ubuntu/Debian)
Pasang	`winget install EclipseAdoptium.Temurin.25.JDK`, atau penginstal `.msi` dari `adoptium.net` dengan **Set JAVA_HOME** dicentang	`brew install --cask temurin@25`, atau penginstal `.pkg` dari `adoptium.net`	`sudo apt install openjdk-25-jdk`, atau Temurin dari repositori apt Adoptium
Periksa	`javac -version` dan `java -version`	sama	sama

Kedua pemeriksaan harus menjawab, dan dengan nomor versi yang sama. Jika `java` menjawab tetapi `javac` tidak, yang terpasang hanyalah *runtime* Java, yang dapat menjalankan program Java tetapi tidak dapat mengompilasinya: pasanglah JDK. Jika keduanya menjawab dengan versi yang *berbeda*, ada dua JDK yang terpasang dan yang lebih lama berada lebih dahulu dalam PATH; pindahkan folder `bin` JDK yang lebih baru ke urutan teratas PATH, atau copot JDK yang lebih lama.

1.8.4 Python 3.14

Pasanglah **Python 3.14**, seri terkini; rilis terbarunya pada saat buku ini disusun adalah 3.14.7, tertanggal 5 Agustus 2026 [8]. Program Python buku ini memerlukan Python 3.10 atau yang lebih baru dan dijalankan di sini dengan Python 3.11. **Tabel 1.5** memperlihatkan cara memasangnya dan perbedaan perintahnya antarsistem.

Tabel 1.5 Memasang Python di setiap sistem operasi.

	Windows 10/11	MacOS	Linux (Ubuntu/Debian)
Pasang	Penginstal dari `https://www.python.org`; di layar pertamanya centang **Add python.exe to PATH**. Atau `winget install Python.Python.3.14`	Penginstal dari `python.org`, atau `brew install python@3.14`	`python3` bawaan sistem: `sudo apt install python3`; rilis Ubuntu terbaru sudah menyertakannya
Menjalankan program	`python hello.py`	`python3 hello.py`	`python3 hello.py`
Periksa	`python --version`	`python3 --version`	`python3 --version`

Di Windows perintahnya `python`; di macOS dan Linux perintahnya `python3`, karena pada sistem yang lebih lama nama `python` berarti Python 2 yang sudah lama pensiun. Di tempat buku ini menulis `python3`, pengguna Windows mengetik `python`.

1.8.5 Dua ekstensi VS Code

Bukalah tampilan Extensions (ikon empat kotak di bilah kiri, atau **Ctrl+Shift+X**, **Cmd+Shift+X** di Mac), cari setiap nama dalam **Tabel 1.6**, lalu tekan **Install**. Atau ketik dua perintah di bawah tabel di terminal mana saja.

Tabel 1.6 Ekstensi VS Code yang dipakai dalam buku ini.

Ekstensi	Penerbit	Pengenal	Memberi Anda
Python	Microsoft	`ms-python.python`	Penyuntingan, menjalankan, *debugging*, dan lingkungan virtual Python [9]
Extension Pack for Java	Microsoft	`vscjava.vscode-java-pack`	Penyuntingan, menjalankan, *debugging*, pengujian, dan tampilan proyek Java [10]

```
code --install-extension ms-python.python
code --install-extension vscjava.vscode-java-pack
```

Ekstensi Python juga memasang *Pylance*, yang menggarisbawahi kesalahan saat Anda mengetik, dan paket Java memasang ekstensi *Language Support for Java*, sebuah *debugger*, dan penjalan tes. Saat pertama kali Anda membuka berkas Java, ekstensi Java mungkin memerlukan satu menit untuk menyala; bilah status di bagian bawah jendela memperlihatkan kemajuannya.

1.8.6 Sebuah folder untuk buku ini

Buatlah satu folder untuk buku ini, misalnya `Profitina` di folder Documents Anda, lalu salin folder kode buku ke dalamnya, sehingga Anda memiliki `Profitina/Chapter01/python`, `Profitina/Chapter01/java`, dan seterusnya. Selalu buka sebuah **folder** di VS Code (**File → Open Folder…**), jangan satu berkas saja. Dengan begitu terminal VS Code dimulai di dalam folder itu, sehingga perintah seperti `python3 milestones.py` menemukan berkas-berkas di sebelahnya. Ketika VS Code bertanya apakah Anda memercayai berkas di dalam folder, jawablah ya untuk folder buku ini: itu salinan milik Anda sendiri.

1.8.7 Memeriksa seluruh meja kerja

Bukalah terminal *baru*, agar terminal itu melihat PATH yang sudah diperbarui, lalu ketik setiap perintah dalam **Tabel 1.7**. Di Windows, ketik `python` di tempat tabel menulis `python3`.

Tabel 1.7 Perintah pemeriksaan, apa yang seharusnya dicetak masing-masing, dan perbaikan yang lazim bila tidak.

Perintah	Jawaban yang baik diawali dengan	Jika gagal
`code --version`	`1.`	Pasang ulang VS Code dengan **Add to PATH**, atau jalankan *Shell Command: Install 'code' command in PATH*
`javac -version`	`javac 25`	Hanya *runtime* yang terpasang, atau JDK yang lebih lama berada lebih dahulu dalam PATH
`java -version`	`openjdk version "25`	Pasang JDK; periksa `JAVA_HOME`
`python3 --version`	`Python 3.14`	Windows: jalankan ulang penginstal dan centang **Add python.exe to PATH**

Di mesin Linux tempat program-program buku ini dijalankan, pemeriksaan mencetak hasil berikut. Mesin itu sedikit tertinggal dari versi yang dianjurkan, dengan JDK 21 dan Python 3.11, yang sekaligus menunjukkan bahwa program-program buku ini juga berjalan di sana.

```
$ javac -version
javac 21.0.10
$ java -version
openjdk version "21.0.10" 2026-01-20
$ python3 --version
Python 3.11.15
```

Mengetik empat perintah memang cepat, tetapi sebuah program dapat melakukan pemeriksaan yang sama dan menjelaskan jawabannya, dan menulis program itu adalah cara yang baik untuk melihat PATH bekerja. **Kode 1.1** menanyakan kepada sistem operasi mana di antara enam alat yang dapat ditemukannya, persis seperti yang dilakukan terminal.

Kode 1.1 `check_setup.py`: melaporkan sistem operasi, Python yang menjalankan program, dan alat meja kerja mana yang dapat dijangkau PATH.

```python
# check_setup.py - laporkan alat meja kerja mana yang terlihat oleh terminal ini
import os
import platform
import shutil

TOOLS = ["code", "python3", "python", "java", "javac", "git"]


def main():
    print("Operating system :", platform.system(), platform.release())
    print("Python running   :", platform.python_version())
    print("Working folder   :", os.getcwd())
    print()
    missing = 0
    for tool in TOOLS:
        where = shutil.which(tool)       # menelusuri PATH, seperti terminal
        if where is None:
            missing += 1
            print(f"  {tool:<8} NOT FOUND on the PATH")
        else:
            print(f"  {tool:<8} {where}")
    print()
    if missing == 0:
        print("Every tool was found.")
    else:
        print(f"{missing} tool(s) not found; see Chapter 1 for the usual fixes.")


if __name__ == "__main__":
    main()
```

Fungsi `shutil.which(tool)` melakukan apa yang dilakukan terminal ketika Anda mengetik sebuah perintah: menelusuri folder-folder PATH secara berurutan dan mengembalikan jalur lengkap program pertama yang cocok, atau `None` bila tidak ada. Format `{tool:<8}` menambahkan spasi pada nama hingga delapan karakter, rata kiri, agar kolom kedua sejajar. Jalankan dengan `python3 check_setup.py` di folder `Chapter01/python`:

```
Operating system : Linux 6.18.44-fc-v37
Python running   : 3.11.15
Working folder   : /home/claude/kit/Profitina/Code_new/Chapter01/python

  code     NOT FOUND on the PATH
  python3  /usr/bin/python3
  python   /usr/bin/python
  java     /usr/bin/java
  javac    /usr/bin/javac
  git      /usr/bin/git

1 tool(s) not found; see Chapter 1 for the usual fixes.
```

Keluaran Anda akan berbeda dalam rinciannya, dan memang seharusnya begitu. Mesin yang dipakai untuk buku ini adalah server tanpa desktop grafis, sehingga tidak memiliki VS Code, dan program menyatakannya dengan jujur alih-alih berpura-pura. Di laptop Anda `code` seharusnya ditemukan, dan di Windows jalurnya berbentuk `C:\Program Files\...`. Alat `git`, sebuah sistem

kendali versi, tidak diperlukan untuk buku ini, tetapi pemeriksaan itu menunjukkan apakah Anda memilikinya.

Java tidak memiliki `which` siap pakai, sehingga **Kode 1.2** menulisnya sendiri. Program ini membaca PATH dengan `System.getenv("PATH")`, memecahnya dengan `File.pathSeparator`, yaitu `;` di Windows dan `:` di sistem lain, dan di Windows juga mencoba akhiran `.exe`, `.cmd`, dan `.bat`, karena perintah Windows bernama `javac` sesungguhnya adalah berkas bernama `javac.exe`.

Kode 1.2 `CheckSetup.java`: laporan yang sama dalam Java, dengan penelusuran PATH kecil yang ditulis sendiri.

```
// CheckSetup.java - laporkan alat meja kerja mana yang terlihat oleh terminal ini
import java.io.File;
import java.util.List;

public class CheckSetup {
    static final List<String> TOOLS =
            List.of("code", "python3", "python", "java", "javac", "git");

    // Cari alat itu di setiap folder PATH, seperti yang dilakukan terminal.
    static String which(String tool) {
        String path = System.getenv("PATH");
        if (path == null) {
            return null;
        }
        boolean windows = System.getProperty("os.name").startsWith("Windows");
        String[] endings = windows ? new String[] {".exe", ".cmd", ".bat"}
                                   : new String[] {""};
        for (String dir : path.split(File.pathSeparator)) {
            for (String end : endings) {
                File f = new File(dir, tool + end);
                if (f.isFile() && f.canExecute()) {
                    return f.getPath();
                }
            }
        }
        return null;
    }

    public static void main(String[] args) {
        System.out.println("Operating system : " + System.getProperty("os.name")
                + " " + System.getProperty("os.version"));
        System.out.println("Java running     : " + Runtime.version());
        System.out.println("Working folder   : " + System.getProperty("user.dir"));
        System.out.println();
        int missing = 0;
        for (String tool : TOOLS) {
            String where = which(tool);
            if (where == null) {
                missing++;
                System.out.printf("  %-8s NOT FOUND on the PATH%n", tool);
            } else {
                System.out.printf("  %-8s %s%n", tool, where);
            }
        }
        System.out.println();
        if (missing == 0) {
```

Kode 1.2 (Lanjutan) *CheckSetup.java: laporan yang sama dalam Java, dengan penelusuran PATH kecil yang ditulis sendiri.*

```
            System.out.println("Every tool was found.");
        } else {
            System.out.println(missing
                    + " tool(s) not found; see Chapter 1 for the usual fixes.");
        }
    }
}
```

Kompilasi dan jalankan di folder `Chapter01/java` dengan `javac CheckSetup.java` lalu `java CheckSetup`:

```
Operating system : Linux 6.18.44-fc-v37
Java running     : 21.0.10+7-Ubuntu-124.04
Working folder   : /home/claude/kit/Profitina/Code_new/Chapter01/java

  code     NOT FOUND on the PATH
  python3  /usr/bin/python3
  python   /usr/bin/python
  java     /usr/bin/java
  javac    /usr/bin/javac
  git      /usr/bin/git

1 tool(s) not found; see Chapter 1 for the usual fixes.
```

Kedua program sepakat, sebagaimana mestinya: keduanya mengajukan pertanyaan yang sama kepada PATH yang sama. Baris `Java running` memperlihatkan string versi lengkap Java yang sedang berjalan; dengan Temurin 25 string itu diawali `25`.

1.8.8 Program pertama Anda, dalam dua bahasa

Buatlah berkas `hello.py` di folder `Chapter01/python`, ketik **Kode 1.3**, lalu simpan.

Kode 1.3 `hello.py`: program pertama; membuktikan bahwa editor, terminal, dan Python bekerja bersama.

```
# hello.py - program pertama buku Profitina
import sys

try:
    name = input("What is your name? ")
except EOFError:                          # tak ada yang bisa dibaca
    name = ""
name = name.strip()
if not name:
    name = "friend"                       # tanpa masukan: sapa seorang teman
print(f"Hello, {name}! Welcome to Profitina.")
print(f"Language version: Python {sys.version_info.major}.{sys.version_info.minor}")
```

Bukalah terminal di VS Code dengan **Terminal → New Terminal** dan jalankan program dengan `python3 hello.py` (`python hello.py` di Windows). Program menanyakan nama Anda; ketik sebuah nama lalu tekan Enter:

```
What is your name? Sari
Hello, Sari! Welcome to Profitina.
Language version: Python 3.11
```

Bacalah program itu baris demi baris. `import sys` memberi akses ke modul `sys`, yang antara lain mengetahui versi Python yang sedang berjalan. `input(...)` mencetak argumennya sebagai prompt dan menunggu satu baris dari papan ketik. Bila tidak ada papan ketik sama sekali,

misalnya ketika masukan program berasal dari berkas kosong, `input` tidak dapat membaca apa pun dan memunculkan eksepsi `EOFError` (*end of file*, akhir berkas); `try` dan `except` menangkapnya dan memakai nama kosong sebagai gantinya. `strip()` membuang spasi di kedua ujung, sehingga nama yang diketik " Sari " disapa sebagai "Sari". Nama kosong menjadi "friend". *f-string* `f"Hello, {name}! ..."` membangun string dengan nilai `name` disisipkan di antara kurung kurawal. Dengan Python 3.14 baris terakhir berbunyi `Python 3.14`.

Sekarang versi Java, **Kode 1.4**. Simpanlah sebagai `Hello.java` di folder `Chapter01/java`. Di Java, kelas publik harus berada dalam berkas yang namanya persis sama, termasuk huruf kapitalnya.

Kode 1.4 `Hello.java`: program pertama dalam Java.

```java
// Hello.java - program pertama buku Profitina
import java.util.Scanner;

public class Hello {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("What is your name? ");
        String name = in.hasNextLine() ? in.nextLine().strip() : "";
        if (name.isEmpty()) {
            name = "friend";                  // tanpa masukan: sapa seorang teman
        }
        System.out.println("Hello, " + name + "! Welcome to Profitina.");
        System.out.println("Language version: Java " + Runtime.version().feature());
    }
}
```

Kompilasi dan jalankan dalam dua langkah:

```
javac Hello.java
java Hello
```

```
What is your name? Sari
Hello, Sari! Welcome to Profitina.
Language version: Java 21
```

Perintah pertama menjalankan kompilator, yang membaca `Hello.java` dan menulis `Hello.class`, berkas *bytecode* yang dipahami mesin virtual Java. Jika `javac` tidak mencetak apa pun, berarti berhasil: diam adalah kabar baik. Perintah kedua menyalakan mesin virtual dan memintanya menjalankan kelas `Hello`; perhatikan bahwa yang Anda tulis adalah nama kelas, tanpa `.class`. Sejak Java 11 Anda juga boleh mengetik `java Hello.java`, dan peluncur akan mengompilasi serta menjalankan program satu-berkas dalam satu langkah.

Program Java itu mencerminkan program Python. Sebuah `Scanner` membaca dari `System.in`, papan ketik. `in.hasNextLine()` menanyakan apakah ada baris yang dapat dibaca, yang berperan seperti pemeriksaan `EOFError` di Python; ekspresi `condition ? a : b` memilih `a` bila kondisinya benar dan `b` bila tidak. `strip()` membuang spasi di kedua ujung, seperti di Python. `Runtime.version().feature()` adalah versi utama Java yang sedang berjalan, 21 di sini dan 25 dengan JDK 25.

Kedua program menangani keadaan tanpa masukan sama sekali. Jalankan `python3 hello.py < empty.txt` dengan berkas kosong, atau di macOS dan Linux cukup `python3 hello.py < /dev/null`, dan program menyapa "friend". Program yang berperilaku wajar ketika masukannya tidak ada adalah contoh kecil dari kebiasaan yang mengalir di seluruh buku ini, dan di seluruh Profitina: *gagallah dengan lantang atau dengan anggun, tetapi jangan pernah gagal diam-diam.*

1.8.9 Program kedua: hari sejak peluncuran

Program pertama membuktikan bahwa perkakasnya bekerja. Program kedua mengerjakan sesuatu yang berguna: membaca tonggak-tonggak **Tabel 1.1** dari sebuah berkas kecil dan mencetak, untuk masing-masing, nama harinya dan banyaknya hari dari peluncuran pada 17 Juli 2026. Berkas `milestones.csv` adalah berkas *CSV* (*comma-separated values*, nilai yang dipisahkan koma): baris pertama menamai kolom, dan setiap baris berikutnya adalah satu rekaman. Isi berkas dibiarkan dalam bahasa Inggris, bahasa yang juga dipakai keluaran program.

```
date,event
2026-07-03,First commit of the current code base (version 0.11.0)
2026-07-17,Production launch at app.profitina.com
2026-07-31,Sharing to X moves to Web Intent; no X API
2026-08-07,Two plans only: Free and Pro
2026-08-19,Self-service privacy rights panel
2026-08-31,First university class of about 250 students
2026-09-01,One Text Many Platforms
2026-09-09,Organization Visibility
2026-09-24,Version 0.14.5-testnet
```

Kode 1.5 membacanya dengan modul `csv` Python, mengubah setiap tanggal menjadi objek `date`, mengurutkan rekaman menurut tanggal, lalu mencetak tabel. Argumen baris perintah yang opsional memberikan tanggal acuan (“per tanggal”); tanpa argumen, program memakai tanggal hari ini.

Kode 1.5 `milestones.py`: membaca tonggak, mencetak jaraknya dari peluncuran, dan menerapkan Persamaan (1.1) pada banyaknya hari hidup.

```
# milestones.py - baca tonggak Profitina dan hitung hari sejak peluncuran
import csv
import sys
from datetime import date

LAUNCH = date(2026, 7, 17)


def load(filename):
    rows = []
    with open(filename, encoding="utf-8", newline="") as f:
        for row in csv.DictReader(f):
            rows.append((date.fromisoformat(row["date"]), row["event"]))
    rows.sort()                                 # terlama dulu, apa pun urutannya
    return rows


def main():
    if len(sys.argv) > 1:
        as_of = date.fromisoformat(sys.argv[1])
    else:
        as_of = date.today()
    rows = load("milestones.csv")
    print(f"{'Date':<11}{'Day':<5}{'From launch':>12}  Event")
    for when, event in rows:
        offset = (when - LAUNCH).days       # negatif: sebelum peluncuran
        day = when.strftime("%a")           # Mon, Tue, ...
        print(f"{when.isoformat():<11}{day:<5}{offset:>+12}  {event}")
    live = (as_of - LAUNCH).days
    print(f"\nAs of {as_of.isoformat()}, Profitina has been live for {live} days"
          f" ({live // 7} weeks and {live % 7} days).")


if __name__ == "__main__":
    main()
```

```
Date       Day   From launch  Event
2026-07-03 Fri           -14  First commit of the current code base (version 0.11.0)
2026-07-17 Fri            +0  Production launch at app.profitina.com
2026-07-31 Fri           +14  Sharing to X moves to Web Intent; no X API
2026-08-07 Fri           +21  Two plans only: Free and Pro
2026-08-19 Wed           +33  Self-service privacy rights panel
2026-08-31 Mon           +45  First university class of about 250 students
2026-09-01 Tue           +46  One Text Many Platforms
2026-09-09 Wed           +54  Organization Visibility
2026-09-24 Thu           +69  Version 0.14.5-testnet

As of 2026-09-24, Profitina has been live for 69 days (9 weeks and 6 days).
```

Itulah keluaran `python3 milestones.py 2026-09-24`. Beberapa rincian patut diperhatikan.

- `csv.DictReader` mengubah setiap baris menjadi kamus (*dictionary*) yang kuncinya berasal dari baris judul, sehingga kode dapat menulis `row["date"]` alih-alih mengingat bahwa tanggal ada di kolom 0.
- `date.fromisoformat("2026-07-17")` mengubah teks dalam format internasional *tahun-bulan-hari* (ISO 8601) menjadi tanggal. Mengurangkan dua tanggal menghasilkan `timedelta`, dan `.days`-nya adalah selisih dalam hari utuh. Aritmetika kalender, dengan bulan yang panjangnya 28 sampai 31 hari dan tahun kabisat, dikerjakan oleh pustaka dengan benar.
- `rows.sort()` mengurutkan daftar pasangan menurut unsur pertamanya, yaitu tanggal, dan memakai unsur kedua hanya untuk memecah seri. Karena itu program mencetak tonggak dalam urutan tanggal meskipun berkasnya disunting dengan ceroboh.
- Format `{offset:>+12}` meratakan-kanan bilangan dalam dua belas karakter dan selalu menampilkan tandanya, sehingga tonggak sebelum peluncuran tampil sebagai `-14`.
- `live // 7` dan `live % 7` adalah *w* dan *r* pada Persamaan (1.1).

Kode 1.6 adalah program yang sama dalam Java.

Kode 1.6 `Milestones.java`: tabel yang sama dalam Java, dengan `LocalDate` dan `ChronoUnit.DAYS`.

```java
// Milestones.java - baca tonggak Profitina dan hitung hari sejak peluncuran
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.time.LocalDate;
import java.time.format.TextStyle;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class Milestones {
    static final LocalDate LAUNCH = LocalDate.of(2026, 7, 17);

    record Milestone(LocalDate when, String event) { }

    static List<Milestone> load(String filename) throws IOException {
        List<Milestone> rows = new ArrayList<>();
        List<String> lines = Files.readAllLines(Path.of(filename),
                StandardCharsets.UTF_8);
        for (String line : lines.subList(1, lines.size())) {   // lewati baris judul
            if (line.isBlank()) {
                continue;
            }
            int comma = line.indexOf(',');
            rows.add(new Milestone(LocalDate.parse(line.substring(0, comma)),
                    line.substring(comma + 1)));
        }
        rows.sort((a, b) -> a.when().compareTo(b.when()));  // terlama dulu
        return rows;
    }

    public static void main(String[] args) throws IOException {
        LocalDate asOf = args.length > 0 ? LocalDate.parse(args[0])
                                         : LocalDate.now();
        List<Milestone> rows = load("milestones.csv");
        System.out.printf("%-11s%-5s%12s  %s%n",
                "Date", "Day", "From launch", "Event");
        for (Milestone m : rows) {
            long offset = ChronoUnit.DAYS.between(LAUNCH, m.when());
            String day = m.when().getDayOfWeek()
                    .getDisplayName(TextStyle.SHORT, Locale.ENGLISH);
            System.out.printf("%-11s%-5s%+12d  %s%n",
                    m.when(), day, offset, m.event());
        }
        long live = ChronoUnit.DAYS.between(LAUNCH, asOf);
        System.out.printf("%nAs of %s, Profitina has been live for %d days"
                + " (%d weeks and %d days).%n", asOf, live, live / 7, live % 7);
    }
}
```

Kode 1.6 (Lanjutan) *Milestones.java: tabel yang sama dalam Java, dengan LocalDate dan ChronoUnit.DAYS.*

Dijalankan dengan `javac Milestones.java` dan `java Milestones 2026-09-24`, program ini mencetak sebelas baris yang persis sama dengan versi Python. Program Java ini memperlihatkan tiga fitur yang akan Anda jumpai lagi. `record` mendeklarasikan kelas kecil yang tugasnya hanya menyimpan nilai, di sini sebuah tanggal dan sebuah peristiwa, dalam satu baris. `Files.readAllLines` membaca seluruh berkas teks menjadi daftar string, dan `lines.subList(1, lines.size())` melewati baris judul. Dan `ChronoUnit.DAYS.between(a, b)` menghitung hari dari `a` ke `b`, negatif bila `b` lebih dahulu.

Program ini pendek, tetapi bukan mainan. Ia memiliki setiap bagian program di bab-bab berikutnya: membaca data dari berkas, mengubah teks menjadi nilai bertipe, menghitung sesuatu yang mudah keliru bila dikerjakan dengan tangan, dan mencetak hasil yang dapat diperiksa manusia.

1.8.10 Menjalankan dan men-*debug* di VS Code

Menjalankan dari editor. Dengan ekstensi pada **Tabel 1.6** terpasang, berkas Python atau Java menampilkan segitiga **Run** di kanan atas editor. Menekannya menjalankan berkas di terminal VS Code. Untuk berkas Java, ekstensi lebih dahulu mengompilasinya untuk Anda. Untuk berkas Python, pada kali pertama, VS Code menanyakan interpreter Python mana yang dipakai; pilihlah 3.14 yang Anda pasang. Bilah status menampilkan interpreter terpilih, dan mengekliknya memungkinkan Anda menggantinya. Program yang menerima argumen baris perintah, seperti `milestones.py 2026-09-24`, paling mudah dijalankan dari terminal.

Memberi masukan dari berkas. Program yang membaca papan ketik dapat membaca berkas sebagai gantinya melalui *pengalihan* (*redirection*): `python3 hello.py < name.txt` atau `java Hello < name.txt` di macOS, Linux, dan Command Prompt Windows, dan `Get-Content name.txt | python hello.py` di PowerShell, yang tidak memahami `<`.

Debugging: mengamati program berpikir. Ketika sebuah program memberikan jawaban yang salah, cara tercepat untuk mengetahui sebabnya adalah *mengamatinya*. Klik di margin sebelah kiri nomor baris: muncul titik merah, sebuah **breakpoint**. Tekan **F5** dan pilih *Python Debugger* atau *Java*. Program berjalan sampai mencapai titik merah itu lalu berhenti. Panel **Variables** kemudian memperlihatkan nilai setiap variabel pada saat itu, dan Anda dapat mengarahkan tetikus ke nama mana pun dalam kode untuk melihat nilainya. **F10** menjalankan satu baris, **F11** masuk ke dalam pemanggilan fungsi, **Shift+F11** keluar lagi, dan **F5** berlanjut ke breakpoint berikutnya. Pasanglah breakpoint pada `print` di dalam gelung **Kode 1.5** dan amati `when` dan `offset` berubah dari baris ke baris. Itulah cara terbaik untuk memahami gelung yang ditulis orang lain.

> **PERINGATAN: KETIKA ADA YANG TIDAK BERES.** *"python is not recognized"* (Windows): Python dipasang tanpa **Add python.exe to PATH**; jalankan ulang penginstal, pilih *Modify*, lalu centang pilihan itu; kemudian buka terminal baru. *Mengetik `python` malah membuka Microsoft Store* (Windows): matikan *App execution aliases* untuk `python.exe` di Settings, atau pasang Python dari python.org. *`javac` tidak ditemukan tetapi `java` ada*: Anda memasang *runtime*; pasanglah JDK. *`error: class Hello is public, should be declared in a file named Hello.java`*: ganti nama berkas agar persis sama dengan nama kelas, termasuk huruf kapital. *`Error: Could not find or load main class Hello.class`*: Anda mengetik `java Hello.class`; ketiklah `java Hello`. *`FileNotFoundError: milestones.csv`*: terminal berada di folder yang berbeda dengan berkasnya; pindah dengan `cd` ke folder itu, atau buka folder itu di VS Code dengan **File → Open Folder**. *Garis bergelombang merah di seluruh berkas Java, padahal dapat dikompilasi*: tunggu ekstensi Java selesai menyala, atau jalankan *Java: Clean Java Language Server Workspace* dari Command Palette (**Ctrl+Shift+P**).

Dengan meja kerja yang sudah berdiri, setiap kode dalam bagian lain buku ini hanya berjarak satu salinan dan satu perintah untuk menjadi milik Anda.

1.9 Profitina dalam Praktik

Kisah bab ini terlihat di dalam produk itu sendiri, dan beberapa tempat layak dibuka di peramban sambil Anda membaca.

Wajah-wajah publik. `profitina.com` adalah situs pemasaran dwibahasa; `app.profitina.com` adalah aplikasinya, yang judul jendelanya berbunyi "Profitina — Dasbor GEO". Antarmuka aplikasi dapat beralih antara bahasa Indonesia dan Inggris serta antara tema terang dan gelap, dan pilihan yang sama berlaku di situs. Situs pemasaran memublikasikan berkas `/llms.txt`, ringkasan pendek yang dapat dibaca mesin untuk model bahasa, yang dibangun Bab 2 untuk Profitina Laundry dan ditelaah Bab 10 secara rinci.

Whitepaper yang berversi. Whitepaper publik, versi 24 pada September 2026, disajikan oleh aplikasi itu sendiri dan tersedia dalam bahasa Indonesia dan Inggris [1]. Dokumen itu adalah sumber tunggal paling aman bagi fakta-fakta dalam buku ini, dan setiap bab yang mengutipnya menyebutkannya. Penafiannya adalah salah satu gagasan pendiri dalam satu kalimat: hasil visibilitas adalah perkiraan metodologis, bukan jaminan tentang apa yang benar-benar akan dikatakan mesin AI pihak ketiga mana pun.

String versi yang jujur. Versi aplikasi pada saat buku ini disusun adalah `0.14.5-testnet`. Angka *0.x* menyatakan bahwa platform ini masih muda dan permukaannya masih berubah; *testnet* menyatakan bahwa koin PROFIT hidup di jaringan uji. Keduanya adalah pernyataan fakta yang dibuat produk tentang dirinya sendiri, di tempat yang dapat dilihat setiap pengguna.

Kanal kelas. Kanal kelas dalam komunitas, dengan moderator, pengumuman tersemat, dan handle netralnya, adalah bagian platform yang paling mungkin dijumpai mahasiswa lebih dahulu. Aturan yang mengaturnya, seperti apa yang boleh dan tidak boleh dilakukan moderator dan mengapa handle bukan nomor induk mahasiswa, diuraikan dalam Bab 4 dan 11.

Sebuah kebiasaan lokal dalam kode. Program sekecil `milestones.py` pun menyentuh salah satu kebiasaan Profitina: zona waktu. Kuota dan batas harian platform direset pada pukul 00.00 WIB (Waktu Indonesia Barat, UTC+7), bukan pada tengah malam UTC, yang berarti pukul 07.00 di Jakarta. Tanggal tanpa zona waktu, seperti yang ada di `milestones.csv`, sudah cukup untuk menghitung hari antartonggak; sebuah *saat* seperti "kuota direset sekarang" memerlukan zona. Lab penjadwal di Bab 9 membuat perbedaan itu menjadi konkret.

Ke mana menulis. Untuk pertanyaan tentang platform, kontak resminya adalah `info@profitina.com` [2].

1.10 Kesalahpahaman yang Sering Terjadi

"Profitina bertanya kepada ChatGPT tentang usaha saya lalu melaporkan jawabannya." Tidak. Mesin jawaban adalah sasaran yang *kesiapannya* untuk mengutip Anda diperkirakan dari sinyal yang dapat diamati; Profitina tidak mengajukan kueri kepadanya dan tidak bergantung padanya. Bab 2 menjelaskan apa yang diukur sebagai gantinya, dan Bab 3 memperlihatkan pemindaiannya.

"Skor 100 menjamin AI akan merekomendasikan saya." Tidak ada skor yang dapat menjamin apa yang akan dikatakan mesin pihak ketiga, dan platform menyatakan hal itu dalam syarat dan ketentuannya [2]. Skor memperkirakan kesiapan; perbaikan menaikkan peluang.

"AI Profitina hanyalah pembungkus chatbot komersial." Setiap fitur AI berjalan pada model lokal dalam dua jalur di mesin milik Profitina sendiri; jalur kode AI terhosting telah dihapus.

"PROFIT adalah mata uang kripto yang dapat saya perdagangkan." PROFIT hidup di devnet Solana, jaringan uji yang tokennya tidak bernilai uang. Penjualan dinonaktifkan. Peluncuran di jaringan utama adalah rencana yang bergantung pada perizinan dan audit.

"Pendiri membangun produknya lebih dahulu, lalu menulis bahan ajar untuk mempromosikannya." Tanggalnya justru berjalan ke arah sebaliknya: karier mengajar dimulai pada 1997, dan buku-buku ajar memakai produk ini sebagai satu kasus di antara banyak kasus karena matakuliah-matakuliah itulah bahan pembuat produk ini.

"Kalau `java -version` berjalan, berarti Java sudah terpasang." Mungkin yang terpasang hanya *runtime*. Anda juga memerlukan `javac`; periksa keduanya, dan periksa bahwa keduanya melaporkan versi yang sama.

"PATH langsung diperbarui begitu penginstal selesai." Terminal membaca PATH ketika dinyalakan. Setelah memasang apa pun, bukalah terminal *baru*, dan di Windows kadang juga jendela VS Code baru, sebelum menyimpulkan bahwa ada yang gagal.

"Saya bisa membuka satu berkas `.py` atau `.java` saja di VS Code dan bekerja dari situ." Bisa, tetapi terminal akan dimulai di folder lain dan program tidak akan menemukan berkas datanya. Bukalah folder bab yang bersangkutan.

1.11 Ringkasan Bab

- **Profitina** adalah platform Visibilitas AI (GEO) Indonesia. Platform ini memperkirakan seberapa siap sebuah usaha, situs, peneliti, atau organisasi untuk ditemukan dan dikutip oleh mesin jawaban AI, dari sinyal yang dapat diamati, dan membantu memperbaiki apa yang ditemukannya. Ia hidup di `app.profitina.com` sejak **17 Juli 2026**.
- Profitina didirikan dan dibangun oleh seorang **dosen Departemen Teknik Informatika ITS** yang mengajar sejak 1997–1998 dan risetnya mencakup sistem pakar, ontologi, logika fuzzy, algoritma genetika, jaringan saraf tiruan, dan pengolahan citra.
- Pendiri mengajar **PAA, Pemrograman Web, dan PBO**, tiga matakuliah yang menjadi bahan platform ini, dan sebuah **pustaka buku ajar** memakai Profitina sebagai kasus; **Profitina Laundry** adalah usaha fiktif untuk latihan praktik. Sebuah kelas berisi sekitar **250 mahasiswa** memakai platform ini sejak 31 Agustus 2026.
- **Lini masa** pada **Tabel 1.1** dan **Gambar 1.1** memperlihatkan kecepatan (versi 0.11 sampai 0.14 dalam dua belas pekan), pengurangan (API X, tingkat Agency, dan iklan dihapus), dan kebiasaan memberi tanggal pada segalanya.
- **Gagasan pendiri** adalah misi yang transparan, inklusif, dan mandiri; visibilitas dan partisipasi yang berguna dalam satu sistem; AI lokal dalam dua jalur; rilis yang teruji (1.577 tes lulus pada versi 0.14.5); dan Indonesia lebih dahulu (dwibahasa, UU PDP, QRIS).
- **PROFIT** sudah berjalan di devnet Solana; penjualan dinonaktifkan; peluncuran di jaringan utama masih rencana. Tidak ada isi buku ini yang merupakan nasihat investasi.
- Buku ini terdiri atas **tiga belas bab** dalam empat kelompok (**Gambar 1.2**); setiap lab ditulis dalam **Python dan Java** dengan keluaran nyata; setiap angka yang dapat berubah disertai tanggalnya.
- **Meja kerja** adalah VS Code dengan ekstensi Python dan Extension Pack for Java, **JDK 25**, dan **Python 3.14**, yang diperiksa dengan perintah pada **Tabel 1.7** atau dengan `check_setup`. Hari dipecah menjadi pekan dan hari dengan Persamaan (1.1).

1.12 Soal Latihan

Soal 1.1. Dengan kata-kata Anda sendiri dan paling banyak tiga kalimat untuk masing-masing, jelaskan (a) apa yang diukur Profitina, (b) apa yang *tidak* dipakainya untuk mengukur, dan (c) mengapa platform menyebut hasilnya "perkiraan".

Soal 1.2. Dengan memakai **Tabel 1.1**, daftarlah tonggak-tonggak yang *menghapus* sesuatu dari produk, dan untuk masing-masing berikan alasan yang disiratkan bab ini. Apa kesamaan mereka?

Soal 1.3. Pasangkan masing-masing dari tiga matakuliah EF234405, EF234301, dan EF234302 dengan satu bagian konkret Profitina yang diuraikan dalam bab ini, dan jelaskan pasangan itu dalam satu kalimat untuk masing-masing.

Soal 1.4. Manakah dari pernyataan berikut yang merupakan fakta yang sudah berjalan pada 24 September 2026, dan mana yang masih rencana? (a) PROFIT dapat dibeli dengan rupiah melalui QRIS. (b) PROFIT dapat dijual untuk rupiah. (c) PROFIT ada di jaringan utama Solana. (d) PROFIT dapat dibelanjakan untuk tambahan kuota pindai. (e) Ekspansi regional ke luar Indonesia.

Soal 1.5. Terapkan Persamaan (1.1) dengan tangan pada $d = 45$, $d = 7$, $d = 0$, dan $d = 183$. Tunjukkan w dan r untuk masing-masing, dan periksa bahwa $7w + r = d$.

Soal 1.6. Seorang teman pengguna Windows melaporkan tiga pesan. Jelaskan masing-masing dan berikan perbaikannya. (a) `'python' is not recognized as an internal or external command`, padahal penginstal selesai tanpa galat. (b) `java -version` mencetak `openjdk version "25.0.1"`, tetapi `javac -version` mencetak `javac 17.0.2`. (c) Setelah mengetik `java Hello.class`: `Error: Could not find or load main class Hello.class`.

Soal 1.7. Jalankan `hello.py` dan `Hello.java` dengan berkas masukan kosong, lalu dengan berkas yang hanya berisi spasi. Tebaklah kedua keluaran sebelum menjalankannya, dan jelaskan baris mana di setiap program yang menentukan hasilnya.

Soal 1.8. Tulislah program, dalam Python dan Java, yang menerima dua tanggal berbentuk `YYYY-MM-DD` di baris perintah dan mencetak berapa hari di antara keduanya, nama hari masing-masing, dan selisihnya dalam pekan dan hari menurut Persamaan (1.1). Gunakan program itu untuk mencari berapa hari setelah peluncuran Profitina peraturan pelaksana undang-undang pelindungan data pribadi Indonesia, PP Nomor 33 Tahun 2026, mulai berlaku pada 16 Januari 2027.

Soal 1.9. Tambahkan tonggak "2026-09-14,Health-verified release gate" di *akhir* `milestones.csv` lalu jalankan lagi `milestones.py 2026-09-24`. Di mana baris baru itu muncul dalam keluaran, dan baris program mana yang menempatkannya di sana? Apa yang terjadi tanpa baris itu?

Soal 1.10. Ubahlah `check_setup.py` atau `CheckSetup.java` agar juga melaporkan *versi* Python dan Java yang akan dijalankan terminal, dengan menjalankan `python3 --version` dan `javac -version` sebagai proses anak. (Petunjuk: `subprocess.run` di Python dengan `capture_output=True, text=True`; `ProcessBuilder` di Java dengan `redirectErrorStream(true)`.) Mengapa versi yang dijalankan terminal dapat berbeda dari versi yang menjalankan `check_setup` itu sendiri?

Soal 1.11. Dengan memakai **Tabel 1.2**, pilihlah baris yang paling tepat menggambarkan Anda dan tulislah rencana membaca buku ini: bab mana, dalam urutan apa, dan lab mana yang akan Anda jalankan. Kemudian tulislah satu pertanyaan yang Anda harapkan dijawab buku ini; periksa di akhir Bab 13 apakah pertanyaan itu terjawab.

Soal 1.12. Bab ini menyatakan bahwa *tanggal* sudah cukup untuk menghitung hari antartonggak, tetapi sebuah *saat* memerlukan zona waktu. Berikan contoh konkret, dengan WIB (UTC+7) dan UTC, tentang kuota harian yang akan direset pada jam lokal yang salah bila program memakai tanggal UTC.

1.13 Penyelesaian Soal Latihan

Penyelesaian 1.1. (a) Profitina mengukur seberapa *siap* sebuah usaha, situs web, peneliti, atau organisasi untuk ditemukan dan dikutip oleh mesin jawaban AI, dari sinyal yang dapat diamati siapa pun di web publik: apakah teksnya dapat diekstraksi dengan bersih, apakah data terstruktur menggambarkan usaha itu, dan apakah fondasi teknisnya kokoh. Kemudian platform menyarankan perbaikan yang diprioritaskan. (b) Profitina tidak bertanya kepada mesin jawaban itu sendiri; ChatGPT, Gemini, dan Perplexity adalah sasaran, bukan komponen, karena alasan kemandirian, privasi, dan aturan yang melarang API pihak ketiga berbayar. (c) Karena tidak ada pihak di luar sebuah mesin yang dapat tahu apa yang akan dikatakannya besok; mesin-mesin itu terus berubah. Skor berbasis sinyal memperkirakan peluang dan dapat dijelaskan, tetapi tidak dapat menjadi jaminan, dan mengatakannya adalah bagian dari misi yang transparan.

Penyelesaian 1.2. API X dihapus pada 31 Juli 2026, karena platform tidak dapat memverifikasi kegiatan berbagi melaluinya dan tidak ingin bergantung pada API pihak ketiga berbayar; berbagi beralih ke Web Intent, tempat pengguna mengunggah dari akunnya sendiri. Tingkat Agency dihapus pada 6–7 Agustus 2026, demi kesederhanaan: dua paket lebih mudah dipahami, didukung, dan dijaga kejujurannya. Iklan ditiadakan selamanya pada waktu yang sama. Kesamaannya: setiap penghapusan mengurangi apa yang *dijanjikan* produk menjadi apa yang dapat *diberikan dan diverifikasi*-nya, dan setiap penghapusan disertai alasan tertulis.

Penyelesaian 1.3. EF234405, *Perancangan dan Analisis Algoritma*: biaya kueri dan jadwal, misalnya halaman umpan yang turun dari 868 ms menjadi 11 ms setelah banyaknya kueri basis data per halaman berkurang. EF234301, *Pemrograman Web*: aplikasi itu sendiri dan widget komunitas yang tertanam di situs pemasaran. EF234302, *Pemrograman Berorientasi Objek*: banyaknya pemeriksaan independen dalam pemindai, masing-masing berupa objek yang melaporkan hasil dan penjelasannya sendiri di balik satu antarmuka bersama.

Penyelesaian 1.4. (a) Sudah berjalan. (b) Rencana; penjualan dinonaktifkan sampai jaringan utama dan kejelasan regulasi. (c) Rencana; hari ini PROFIT berada di devnet. (d) Sudah berjalan. (e) Rencana (Fase 4).

Penyelesaian 1.5. Untuk $d = 45$: $45/7 = 6.43\ldots$, $w = 6$, $r = 45 - 42 = 3$, dan $7 \cdot 6 + 3 = 45$. Untuk $d = 7$: $w = 1$, $r = 0$, dan $7 + 0 = 7$. Untuk $d = 0$: $w = 0$, $r = 0$. Untuk $d = 183$: $183/7 = 26.14\ldots$, $w = 26$, $r = 183 - 182 = 1$, dan $182 + 1 = 183$. Pemeriksaan $7w + r = d$ berlaku karena definisi r pada Persamaan (1.1); yang dijamin oleh fungsi lantai adalah fakta lainnya, $0 \leq r < 7$.

Penyelesaian 1.6. (a) Kotak **Add python.exe to PATH** pada penginstal tidak dicentang, atau terminal dibuka sebelum pemasangan. Bukalah terminal baru; bila pesan itu tetap muncul, jalankan ulang penginstal, pilih *Modify* dan centang kotak itu, atau tambahkan folder Python ke Path pengguna secara manual. (b) Ada dua JDK yang terpasang, dan JDK 17 yang lebih lama berada lebih dahulu dalam PATH untuk `javac` dibandingkan JDK 25 untuk `java`. Letakkan folder `bin` JDK 25 paling depan dalam PATH (dan atur `JAVA_HOME` ke JDK 25), atau copot JDK 17, lalu buka terminal baru. (c) Peluncur menerima *nama kelas*, bukan nama berkas; ketiklah `java Hello`.

Penyelesaian 1.7. Dengan berkas kosong, `input` Python memunculkan `EOFError`, `except` mengisi `name` dengan `""`, dan program mencetak `Hello, friend! Welcome to Profitina.`; `in.hasNextLine()` di Java mengembalikan `false`, namanya `""`, dan keluarannya sama. Dengan berkas yang hanya berisi spasi, kedua program *memang* membaca satu baris, tetapi `strip()` mengubahnya menjadi `""`, sehingga keduanya kembali menyapa "friend". Baris penentunya adalah `name = name.strip()` diikuti `if not name:` di Python, dan `in.nextLine().strip()` diikuti `if (name.isEmpty())` di Java. Tanpa `strip()`, keduanya akan menyapa nama yang terdiri atas spasi: `Hello,    ! Welcome to Profitina.`

Penyelesaian 1.8. **Kode 1.7** dan **Kode 1.8** memeriksa banyaknya argumen, mengurai kedua tanggal (melaporkan tanggal yang keliru alih-alih macet), dan menerapkan Persamaan (1.1) pada selisih mutlaknya. `divmod(a, 7)` di Python mengembalikan pasangan $(\lfloor a/7 \rfloor, a \bmod 7)$ dalam satu pemanggilan.

Kode 1.7 `sol_countdown.py`: hari, pekan, dan nama hari di antara dua tanggal.

```python
# sol_countdown.py - hari, pekan, dan nama hari di antara dua tanggal (Soal 1.8)
import sys
from datetime import date


def main():
    if len(sys.argv) != 3:
        print("usage: python3 sol_countdown.py FROM TO   (dates as YYYY-MM-DD)")
        sys.exit(1)
    try:
        start = date.fromisoformat(sys.argv[1])
        end = date.fromisoformat(sys.argv[2])
    except ValueError as e:
        print("Not a date:", e)
        sys.exit(1)
    days = (end - start).days
    weeks, rest = divmod(abs(days), 7)       # pembagian lantai dan sisa
    word = "after" if days >= 0 else "before"
    print(f"{end.isoformat()} ({end.strftime('%A')}) is {abs(days)} days {word}"
          f" {start.isoformat()} ({start.strftime('%A')}),")
    unit = "day" if rest == 1 else "days"
    print(f"that is {weeks} weeks and {rest} {unit}.")


if __name__ == "__main__":
    main()
```

Kode 1.8 `SolCountdown.java`: perhitungan yang sama dengan `LocalDate` dan `ChronoUnit`.

```java
// SolCountdown.java - hari, pekan, dan nama hari di antara dua tanggal (Soal 1.8)
import java.time.LocalDate;
import java.time.format.DateTimeParseException;
import java.time.format.TextStyle;
import java.time.temporal.ChronoUnit;
import java.util.Locale;

public class SolCountdown {
    static String weekday(LocalDate d) {
        return d.getDayOfWeek().getDisplayName(TextStyle.FULL, Locale.ENGLISH);
    }

    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("usage: java SolCountdown FROM TO"
                    + "   (dates as YYYY-MM-DD)");
            System.exit(1);
        }
        LocalDate start;
        LocalDate end;
        try {
            start = LocalDate.parse(args[0]);
            end = LocalDate.parse(args[1]);
        } catch (DateTimeParseException e) {
            System.out.println("Not a date: " + e.getParsedString());
            System.exit(1);
            return;
        }
        long days = ChronoUnit.DAYS.between(start, end);
        long weeks = Math.abs(days) / 7;          // pembagian lantai ...
        long rest = Math.abs(days) % 7;           // ... dan sisa
        String word = days >= 0 ? "after" : "before";
        System.out.println(end + " (" + weekday(end) + ") is " + Math.abs(days)
                + " days " + word + " " + start + " (" + weekday(start) + "),");
        String unit = rest == 1 ? "day" : "days";
        System.out.println("that is " + weeks + " weeks and " + rest + " "
                + unit + ".");
    }
}
```

Menjalankan `python3 sol_countdown.py 2026-07-17 2027-01-16` dan `java SolCountdown 2026-07-17 2027-01-16` memberikan jawaban yang sama:

```
2027-01-16 (Saturday) is 183 days after 2026-07-17 (Friday),
that is 26 weeks and 1 day.
```

PP Nomor 33 Tahun 2026 mulai berlaku 183 hari setelah peluncuran Profitina. Dengan urutan tanggal dibalik, `2026-09-24 2026-07-17`, kedua program mencetak `2026-07-17 (Friday) is 69 days before 2026-09-24 (Thursday),` diikuti `that is 9 weeks and 6 days.`

Penyelesaian 1.9. Baris baru muncul di antara “Organization Visibility” (+54) dan “Version 0.14.5-testnet” (+69), dengan selisih +59 dan nama hari `Mon`, karena `rows.sort()` di Python dan `rows.sort(...)` di Java mengurutkan rekaman menurut tanggal sebelum dicetak. Tanpa pengurutan itu, baris tersebut akan dicetak paling akhir, tidak berurutan, dan pembaca yang memindai tabel akan menyimpulkan bahwa gerbang rilis itu datang setelah rilis 24 September.

Penyelesaian 1.10. Di Python, untuk setiap perintah:

```
import subprocess
result = subprocess.run([tool, "--version"], capture_output=True, text=True)
print(tool, (result.stdout or result.stderr).strip())
```

(`javac` memahami `-version`; versi lama mencetaknya ke aliran galat, itulah sebabnya kedua aliran dibaca.) Di Java, `new ProcessBuilder("javac", "-`

`version").redirectErrorStream(true).start()` lalu baca `process.getInputStream()`. Versinya dapat berbeda karena `check_setup` melaporkan interpreter yang *menjalankannya*, yang mungkin dinyalakan melalui jalur lengkap atau melalui pemilih interpreter VS Code, sedangkan terminal menjalankan program mana pun yang *pertama dalam PATH*. Bila keduanya berbeda, VS Code dan terminal memakai dua pemasangan yang berbeda, penyebab yang sering dari keluhan "di terminal jalan, di VS Code tidak".

Penyelesaian 1.11. Jawabannya bersifat pribadi. Contoh untuk seorang mahasiswa: Bab 1 dan 2 secara utuh; Bab 3 dengan lab G1; lalu Bab 4 dan 5 dengan lab G2 dan G3; kemudian Bab 9 dengan G4 dan Bab 11 dengan G7; sisanya berurutan. Pertanyaan yang dapat dibawa: "Bagaimana sebuah platform dapat tahu bahwa sebuah unggahan berguna tanpa membaca isi hati pengunggahnya?" (Bab 7).

Penyelesaian 1.12. Misalkan kuota harus direset pada pukul 00.00 WIB. Saat 00.00 WIB tanggal 25 September sama dengan pukul 17.00 UTC tanggal 24 September. Program yang menanyakan "tanggal hari ini" dalam UTC akan terus menjawab 24 September sampai pukul 00.00 UTC, yaitu pukul 07.00 WIB tanggal 25 September. Jadi pengguna di Surabaya yang menghabiskan kuotanya pukul 23.30 WIB tanggal 24 September akan mendapati kuotanya *tidak* diperbarui pada tengah malam, dan baru diperbarui pukul tujuh pagi. Perbaikannya adalah menghitung tanggal dalam zona `Asia/Jakarta`, dengan `zoneinfo.ZoneInfo("Asia/Jakarta")` di Python atau `ZoneId.of("Asia/Jakarta")` di Java.

Daftar Pustaka

[1] Profitina, *Profitina Whitepaper — Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[2] Profitina, "Kebijakan Privasi" dan "Syarat dan Ketentuan" (v1.1, berlaku 6 Agustus 2026). [Online]. Available: https://app.profitina.com/legal/privacy

[3] ANTARA, "Survei APJII: Penetrasi internet di Indonesia 2026 capai 81,7 persen," 21 Mei 2026. [Online]. Available: https://www.antaranews.com/berita/5576225/survei-apjii-penetrasi-internet-di-indonesia-2026-capai-817-persen

[4] Microsoft, "Visual Studio Code 1.139," *Visual Studio Code Release Notes*, Sep. 23, 2026. [Online]. Available: https://code.visualstudio.com/updates/v1_139

[5] Inside.java, "JDK 27 is now available," Sep. 15, 2026. [Online]. Available: https://inside.java/2026/09/15/jdk-27-available/

[6] Oracle, "Oracle Java SE Support Roadmap." [Online]. Available: https://www.oracle.com/java/technologies/java-se-support-roadmap.html (diakses Sep. 2026).

[7] Eclipse Foundation, "Eclipse Temurin," *Adoptium*. [Online]. Available: https://adoptium.net (diakses Sep. 2026).

[8] Python Software Foundation, "Python 3.14.7," Aug. 5, 2026. [Online]. Available: https://www.python.org/downloads/latest/python3.14/

[9] Microsoft, "Getting Started with Python in VS Code," *Visual Studio Code Documentation*. [Online]. Available: https://code.visualstudio.com/docs/python/python-tutorial (diakses Sep. 2026).

[10] Microsoft, "Getting Started with Java in VS Code," *Visual Studio Code Documentation*. [Online]. Available: https://code.visualstudio.com/docs/java/java-tutorial (diakses Sep. 2026).

Bab 2

Dari Pencarian ke Jawaban: Bangkitnya GEO

Di akhir bab ini Anda akan mampu menjelaskan, kepada pemilik toko maupun kepada insinyur, mengapa *ditemukan* di internet kini tidak lagi sama dengan *diperingkat*: bagaimana mesin pencari menyusun daftar tautannya, bagaimana mesin jawaban AI sebaliknya membaca sumber-sumber lalu menulis satu jawaban, dan apa kata pengukuran terbaru tahun 2025 dan 2026 tentang besarnya pergeseran itu. Anda akan tahu dari mana istilah GEO, *Generative Engine Optimization*, berasal, apa yang sesungguhnya diukur makalah riset yang memperkenalkannya, dan bagaimana Profitina mendefinisikan disiplin ini bagi penggunanya. Anda akan memahami tiga berkas dan blok kecil yang dapat dibaca mesin, yang menjadi titik awal setiap upaya GEO: data terstruktur schema.org dalam JSON-LD, `robots.txt`, dan `llms.txt` yang lebih baru, termasuk apa yang dapat dan tidak dapat dilakukan masing-masing. Dan Anda akan membangun lab G6, sebuah generator yang mengubah berkas kecil berisi fakta usaha Profitina Laundry menjadi ketiganya, ditambah pemeriksa yang membacanya kembali dan membuktikan keabsahannya, dalam Python dan Java.

2.1 Posisi Kita Sejauh Ini

Bab 1 menuturkan kisah Profitina dan menyiapkan meja kerja: VS Code, Python, dan JDK, yang diuji dengan dua program kecil. Bab itu juga mengajukan sebuah klaim yang belum dijelaskannya. Profitina, katanya, mengukur seberapa siap sebuah usaha untuk *ditemukan dan dikutip oleh mesin jawaban AI*. Mengapa sebuah usaha memerlukannya? Selama dua puluh lima tahun, usaha yang ingin ditemukan di internet menggarap posisinya dalam daftar hasil Google, dan satu profesi utuh, optimasi mesin pencari, tumbuh untuk membantunya. Apa yang berubah?

Bab ini menjawab pertanyaan itu dalam lima langkah. Subbab 2.2 mengingatkan kembali cara kerja daftar tautan dan mengapa daftar itu membentuk web. Subbab 2.3 menjelaskan dengan kata-kata sederhana apa yang dilakukan mesin jawaban secara berbeda. Subbab 2.4 memberi angka pada perubahan itu, semuanya dari 2025 dan 2026 dan semuanya dengan sumbernya. Subbab 2.5 kembali ke makalah riset yang menamai disiplin baru ini, dan Subbab 2.6 memberikan definisi Profitina sendiri. Kemudian Subbab 2.7 memperkenalkan lapisan web yang dapat dibaca mesin, dan Subbab 2.8 membangunnya untuk Profitina Laundry.

2.2 Dua Puluh Lima Tahun Daftar Tautan

2.2.1 Merayapi, mengindeks, memeringkat

Mesin pencari web mengerjakan tiga tugas, dan ada baiknya ketiganya dipisahkan.

Ia **merayapi** (*crawl*). Program yang disebut *crawler* atau *bot* mengambil sebuah halaman, menemukan tautan di dalamnya, lalu menambahkan halaman-halaman tertaut ke antrean halaman yang akan diambil berikutnya. Berawal dari beberapa situs terkenal dan mengikuti tautan, crawler pada akhirnya mengunjungi sebagian besar web publik. Sebelum mengambil

halaman dari sebuah situs, crawler yang santun membaca berkas `robots.txt` situs itu, yang menyatakan apa yang diizinkan pemiliknya; Subbab 2.7.3 menjelaskan caranya.

Ia **mengindeks**. Untuk setiap halaman yang diambilnya, mesin mencatat kata apa muncul di mana, seperti indeks di bagian belakang buku yang mencatat di halaman mana setiap istilah muncul. *Indeks terbalik* (*inverted index*) memetakan setiap kata ke daftar halaman yang memuatnya, sehingga mesin dapat menjawab "halaman mana yang memuat *laundry* dan *Rungkut*?" dalam sepersekian detik tanpa membaca ulang web.

Ia **memeringkat**. Banyak halaman memuat kata-kata sebuah kueri; mesin harus memutuskan mana yang ditampilkan lebih dahulu. Jawaban awal Google, PageRank, memperlakukan tautan dari halaman A ke halaman B sebagai suara A untuk B, dan suara dari halaman penting sebagai lebih bernilai [1]. Pemeringkatan modern menggabungkan ratusan sinyal, tetapi bentuk keluarannya tetap sama selama seperempat abad: sebuah **daftar tautan** yang berurutan, sepuluh per halaman, masing-masing dengan judul dan cuplikan teks.

2.2.2 Manusia yang memilih

Daftar tautan memiliki satu sifat yang membentuk seluruh web komersial: *manusia yang memilih*. Mesin mengusulkan urutan, tetapi manusialah yang membaca judul dan cuplikan, memutuskan mana yang tampak menjanjikan, mengeklik satu atau beberapa, lalu membaca halaman-halaman itu sendiri. Situs web mendapat kunjungan; usaha di dalamnya mendapat kesempatan untuk meyakinkan.

Optimasi mesin pencari (*search engine optimization*, SEO) adalah disiplin untuk memperbaiki posisi halaman dalam daftar itu. Alat-alatnya sudah dikenal: halaman yang dimuat dengan cepat dan dapat dirayapi, judul dan subjudul yang memuat kata-kata yang dicari orang, konten yang layak ditautkan oleh situs lain. SEO tidak akan hilang, dan Profitina tidak pernah mengatakan demikian. Namun satuan keberhasilannya, *posisi dalam daftar*, mengandaikan adanya daftar dan adanya orang yang akan melihatnya.

2.3 Datangnya Mesin Jawaban

2.3.1 Satu jawaban, bukan sepuluh tautan

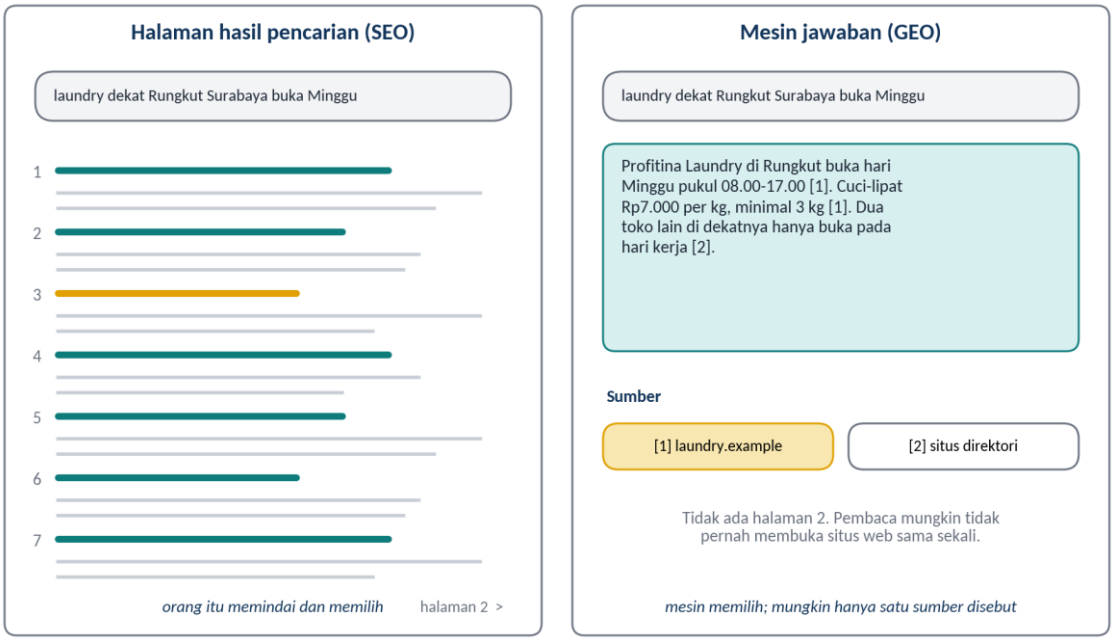
Mesin jawaban (*answer engine*) menerima pertanyaan yang sejenis, tetapi tidak mengembalikan daftar untuk dipilah orang. Ia mengembalikan sebuah *jawaban*: beberapa kalimat atau paragraf teks yang lancar, ditulis untuk pertanyaan ini, sering kali dengan rujukan bernomor kecil yang menunjuk ke sumber yang dipakainya. ChatGPT dengan pencarian, AI Overviews dan AI Mode milik Google, Gemini, Microsoft Copilot, dan Perplexity adalah contoh yang paling dikenal pada 2026.

> **DEFINISI.** *Mesin generatif* (*generative engine*), atau *mesin jawaban*, adalah sistem yang menggabungkan langkah pencarian konvensional, yang menemukan kembali dokumen yang relevan dengan sebuah kueri, dengan model bahasa besar yang membaca dokumen-dokumen itu dan membangkitkan satu tanggapan yang berpijak padanya, biasanya dengan mengutip sebagian sumbernya di dalam teks [2].

Di dalamnya, kebanyakan mesin jawaban mengikuti pola yang dikenal sebagai *retrieval-augmented generation* (pembangkitan yang diperkaya temu kembali) [3]. Pertanyaan dipakai untuk *menemukan kembali* segelintir cuplikan yang relevan, baik dari indeks web milik mesin itu sendiri maupun dari indeks pencarian mitranya. Cuplikan itu diletakkan di hadapan model

bahasa bersama pertanyaannya. Model lalu *membangkitkan* jawaban, dan dalam prosesnya ia dapat mengutip sebuah cuplikan, menyatakan ulang fakta darinya, atau mengabaikannya sama sekali. Model tidak sedang mencari ketika menulis; ia meringkas apa yang diserahkan langkah temu kembali kepadanya, ditambah apa yang dipelajarinya selama pelatihan.

Gambar 2.1 menjajarkan kedua pengalaman itu untuk pertanyaan yang mungkin diajukan seorang pelanggan di Surabaya.



Gambar 2.1 Pertanyaan yang sama di halaman hasil pencarian (kiri) dan di mesin jawaban (kanan). Di kiri, orang memindai daftar berperingkat lalu memilih; di kanan, mesin sudah memilih, dan hanya menyebut satu atau dua sumber.

2.3.2 Apa yang berubah bagi sebuah usaha

Lihatlah sisi kanan **Gambar 2.1** dari sudut pandang pemilik binatu. Tiga hal telah berubah.

Mesin yang memilih. Dalam sebuah daftar, posisi ketiga masih mendapat sejumlah klik. Dalam sebuah jawaban, usaha itu *ada di dalam kalimat* atau tidak ada sama sekali. Situs pemasaran Profitina merumuskannya demikian: GEO “competes for a sentence inside the answer itself — where there is no page two, and often only one source gets named”, yakni bersaing memperebutkan sebuah kalimat di dalam jawaban itu sendiri, tempat tidak ada halaman kedua dan sering kali hanya satu sumber yang disebut [4].

Pembaca mungkin tidak pernah berkunjung. Jika jawaban sudah menyebut bahwa binatu itu buka hari Minggu pukul 08.00 dan memasang tarif Rp7.000 per kilogram, pelanggan sudah mendapatkan yang diinginkannya tanpa membuka situs web mana pun. Kunjungan, yang hendak dimenangkan SEO, mungkin tidak pernah terjadi. *Dikutip dengan benar* menjadi tujuan.

Mesin harus dapat mengekstraksi. Model bahasa hanya dapat menyatakan ulang fakta yang ditemukan langkah temu kembalinya dan dapat dibacanya. Harga yang tersembunyi di foto menu, jam buka yang hanya disebut dalam video promosi, atau alamat yang terpecah ke dalam tiga kotak hiasan bisa jadi tidak terlihat olehnya. Halaman yang menyatakan faktanya dengan lugas, di bagian atas, dalam kalimat yang dapat diangkat mesin, memiliki keunggulan yang tidak banyak berhubungan dengan posisinya dalam daftar mana pun.

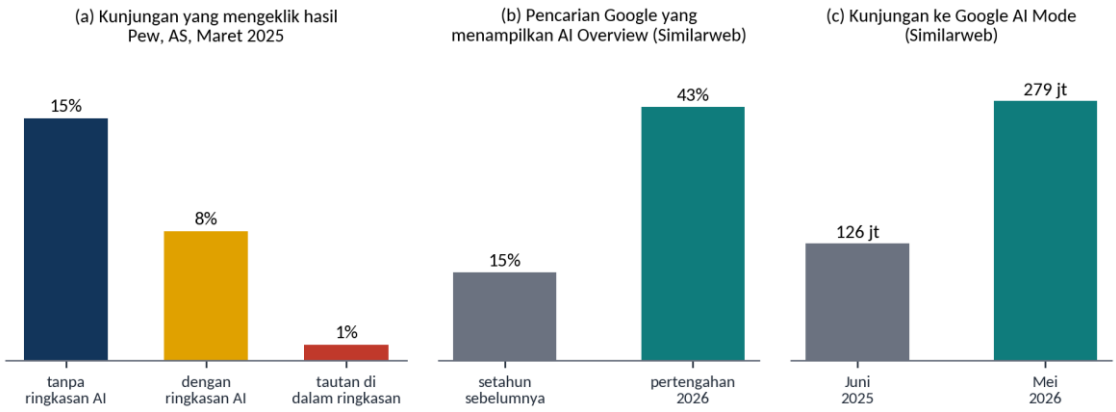
> **WAWASAN KUNCI:** Dalam daftar tautan, satuan keberhasilan adalah *posisi*. Dalam sebuah jawaban, satuannya adalah *sitasi*: menjadi sumber yang fakta, kutipan, atau rekomendasinya diulang oleh mesin. GEO adalah disiplin untuk meraih sitasi; SEO tetap menjadi disiplin untuk meraih posisi. Keduanya saling melengkapi.

2.4 Angka-Angka, 2025–2026

Sebuah pergeseran perilaku hanya menarik bila besar. Subbab ini menghimpun pengukuran paling berguna yang tersedia pada September 2026, masing-masing dengan tanggal dan sumbernya, dan menjelaskan cara membacanya. **Tabel 2.1** mengumpulkannya di satu tempat; **Gambar 2.2** menggambar tiga di antaranya yang paling mudah dibandingkan.

Tabel 2.1 Pengukuran pergeseran dari daftar ke jawaban, 2025–2026, dengan tanggal dan sumber masing-masing. "jt" berarti juta.

Pengukuran	Nilai	Kapan	Sumber
Kunjungan yang mengeklik hasil tradisional, ketika ringkasan AI muncul / tidak muncul	8% / 15%	Maret 2025 (AS, 900 orang dewasa, 68.879 pencarian)	Pew Research Center [5]
Kunjungan yang mengeklik tautan *di dalam* ringkasan AI	1%	Maret 2025	Pew Research Center [5]
Pencarian Google yang menampilkan ringkasan AI	18%	Maret 2025	Pew Research Center [5]
Pencarian Google yang menampilkan AI Overview	dari 15% menjadi 43% dalam setahun	dilaporkan Juli 2026	Similarweb melalui TechCrunch [6]
Kunjungan ke Google AI Mode	126 jt (Juni 2025) → 279 jt (Mei 2026)	2025–2026	Similarweb melalui TechCrunch [6]
Kueri ChatGPT desktop di AS yang jawabannya memuat sitasi	6,8%	Mei 2026	Similarweb melalui TechCrunch [6]
Pengguna aktif mingguan ChatGPT	lebih dari 900 jt	Februari 2026	OpenAI, dilaporkan Search Engine Land [7]
Pengguna aktif bulanan Gemini	950 jt (750 jt pada K4 2025)	K2 2026, dilaporkan 22 Juli 2026	Alphabet, dilaporkan Forbes [8]
Kueri Perplexity per bulan	780 jt, menurut pengungkapan terakhir	Mei 2025	CEO Perplexity, dilaporkan sumber sekunder [9]
Pengguna internet di Indonesia	235,3 jt dari 287,3 jt jiwa (81,72%)	survei 2026, dirilis Mei 2026	APJII, dilaporkan ANTARA [10]



Gambar 2.2 Tiga pengukuran dari **Tabel 2.1**: (a) proporsi kunjungan Google yang mengeklik hasil, dengan dan tanpa ringkasan AI (Pew, Maret 2025); (b) proporsi pencarian Google yang menampilkan AI Overview dengan selang setahun; (c) kunjungan ke Google AI Mode (keduanya Similarweb, dilaporkan Juli 2026).

2.4.1 Lebih sedikit klik ketika jawaban muncul

Pengukuran yang paling cermat berasal dari Pew Research Center, yang pada Maret 2025 merekam penjelajahan nyata 900 orang dewasa di Amerika Serikat yang bersedia ikut serta, dan menganalisis 68.879 pencarian Google yang mereka lakukan [5]. Pada kunjungan ketika Google menampilkan ringkasan AI, orang mengeklik hasil pencarian tradisional pada 8% kunjungan; pada kunjungan tanpa ringkasan, pada 15% kunjungan. Mereka mengeklik tautan di dalam ringkasan AI itu sendiri hanya pada 1% kunjungan. Dan mereka lebih mungkin mengakhiri sesi penjelajahan sama sekali setelah halaman dengan ringkasan AI (26% kunjungan semacam itu) dibandingkan setelah halaman tanpa ringkasan (16%).

Seberapa besar penurunan dari 15% ke 8%? Cara yang berguna untuk membandingkan dua laju adalah *perubahan relatif*: jika sebuah laju bergerak dari a ke b, perubahan relatifnya adalah

$$\Delta_{\text{rel}} = \frac{b - a}{a} \tag{2.1}$$

Untuk angka Pew, $a = 15$ dan $b = 8$, sehingga menurut Persamaan (2.1) $\Delta_{\text{rel}} = (8 - 15)/15 = -7/15 \approx -0.47$: ketika ringkasan muncul, peluang klik pada hasil tradisional turun sekitar 47 persen, hampir separuhnya. Bedakan dengan jelas antara ini dan perubahan *mutlak*: lajunya turun 7 *poin persentase*, dari 15 ke 8, yang merupakan penurunan 47 *persen* dari laju semula. Surat kabar sering mencampuradukkan keduanya; buku ini tidak.

2.4.2 Lebih banyak jawaban, lebih sering

Kelompok angka kedua mengukur seberapa *sering* jawaban muncul. Menurut data Similarweb yang dilaporkan TechCrunch pada Juli 2026, AI Overviews Google meningkat dari muncul pada 15% pencarian menjadi 43% dalam satu tahun, dan kunjungan ke AI Mode Google yang terpisah meningkat lebih dari dua kali lipat, dari 126 juta pada Juni 2025 menjadi 279 juta pada Mei 2026 [6]. Hitungan Pew sendiri setahun sebelumnya, 18% pencarian dengan ringkasan AI pada Maret 2025 [5], sejalan dengan ujung bawah tren itu, walaupun kedua studi mengukur sampel yang berbeda dengan cara yang berbeda, sehingga angkanya tidak boleh digabung menjadi satu kurva.

Laporan Similarweb yang sama memuat angka yang sangat penting bagi GEO: per Mei 2026, hanya 6,8% kueri ChatGPT desktop di Amerika Serikat yang menghasilkan jawaban dengan sitasi [6]. Sebagian besar jawaban tidak menyebut sumber sama sekali. Ketika sebuah jawaban mengutip, menjadi salah satu dari sedikit sumber yang disebut menjadi sangat berharga, dan

laporan itu mencatat bahwa jumlah jawaban yang memuat sitasi telah naik lebih dari lima kali lipat dalam setahun.

2.4.3 Skala mesin-mesin itu

Kelompok ketiga mengukur besarnya khalayak. OpenAI mengumumkan pada Februari 2026 bahwa ChatGPT memiliki lebih dari 900 juta pengguna aktif mingguan [7]. Alphabet melaporkan pada Juli 2026 bahwa aplikasi Gemini-nya memiliki 950 juta pengguna aktif bulanan, naik dari 750 juta dua kuartal sebelumnya [8]. Angka spesifik terakhir Perplexity untuk seluruh perusahaan adalah 780 juta kueri pada Mei 2025, ketika itu tumbuh lebih dari 20% per bulan [9]; tidak ada angka sebanding yang diterbitkan hingga September 2026, sehingga buku ini mengutipnya sebagai "pengungkapan terakhir". Pengguna aktif mingguan dan bulanan adalah ukuran yang berbeda, dan satu orang dapat memakai beberapa mesin, sehingga angka-angka ini tidak dapat dijumlahkan. Yang ditunjukkannya adalah bahwa setiap mesin jawaban besar menjangkau khalayak yang ordenya sama dengan jejaring sosial terbesar.

2.4.4 Indonesia

Bagi usaha di Indonesia, khalayak yang relevan besar dan masih tumbuh. Survei 2026 Asosiasi Penyelenggara Jasa Internet Indonesia (APJII), yang dirilis Mei 2026, mencatat sekitar 235,3 juta pengguna internet dari populasi sekitar 287,3 juta: penetrasi 81,72%, naik dari 80,66% pada 2025 [10]. Dengan Persamaan (2.1), kenaikan dari 80,66 ke 81,72 adalah perubahan relatif sebesar $(81.72 - 80.66)/80.66 \approx 0.013$, sekitar 1,3 persen dalam setahun. Pasar ini tidak lagi tumbuh cepat dalam hal *jumlah orang*; pertumbuhan kini terjadi pada apa yang dilakukan orang-orang itu secara daring, dan bertanya kepada AI alih-alih memindai daftar adalah salah satu kegiatan yang paling cepat tumbuh.

2.4.5 Perlakukan ramalan dengan hati-hati

> **PERINGATAN: RAMALAN BUKAN PENGUKURAN.** Pada Februari 2024 lembaga riset Gartner meramalkan bahwa "by 2026, traditional search engine volume will drop 25%", yakni volume mesin pencari tradisional akan turun 25% pada 2026 karena chatbot AI dan agen virtual mengambil alih [11]. Sekarang sudah 2026, dan pengukuran di atas menuturkan kisah yang lebih rumit: jawaban muncul jauh lebih sering *di dalam* mesin pencari, dan klik pada tautan turun ketika itu terjadi, tetapi "volume pencarian turun seperempat" bukanlah yang diukur oleh studi mana pun dalam **Tabel 2.1**. Ulasan pada 2026 berpendapat bahwa ramalan itu tidak terwujud sebagaimana dinyatakan. Pelajarannya bagi pembaca, dan bagi platform yang menjual kepada usaha, sama dengan yang diterapkan Profitina pada dirinya sendiri: kutiplah pengukuran bersama tanggal dan metodenya, beri label ramalan sebagai ramalan, dan jangan pernah membangun janji di atas ramalan pihak lain.

2.5 Apa Itu GEO: Asal-Usul Akademiknya

2.5.1 Makalah yang menamainya

Istilah *Generative Engine Optimization* diperkenalkan oleh tim riset dari Princeton University, Georgia Tech, Allen Institute for AI, dan IIT Delhi: Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, dan Ameet Deshpande. Makalah mereka, "GEO: Generative Engine Optimization", pertama kali muncul sebagai pracetak pada 16 November 2023 dan diterbitkan di KDD 2024, konferensi ACM tentang penemuan pengetahuan dan penggalian data [2].

Makalah itu mengajukan pertanyaan yang dahulu tidak dapat diajukan pemilik situs kepada mesin pencari dengan cara yang sama: jika mesin generatif menulis satu jawaban dari beberapa sumber, apa yang membuat satu sumber lebih *terlihat* dalam jawaban itu dibandingkan sumber lain, dan dapatkah pemilik sumber memperbaikinya?

2.5.2 Mengukur visibilitas di dalam jawaban

Untuk menjawabnya, tim itu lebih dahulu harus mendefinisikan apa arti "terlihat" ketika tidak ada daftar. Ukuran utama mereka melihat kalimat-kalimat dalam jawaban yang dibangkitkan yang mengutip sebuah sumber, menghitung banyaknya kata, dan memberi bobot lebih pada kalimat yang muncul lebih awal, karena pembaca lebih memperhatikan bagian awal jawaban. Dituliskan secara lengkap, untuk sumber c_i dan tanggapan yang dibangkitkan r yang tersusun atas kalimat-kalimat S:

$$\mathrm{Imp}_{\mathrm{pwc}}(c_i, r) = \frac{\sum_{s \in S_{c_i}} |s| \cdot e^{-\mathrm{pos}(s)/|S|}}{\sum_{s \in S} |s|} \tag{2.2}$$

Bacalah Persamaan (2.2) dari dalam ke luar. S_{c_i} adalah himpunan kalimat tanggapan yang mengutip sumber c_i. $|s|$ adalah banyaknya kata dalam kalimat s, sehingga pembilangnya menjumlahkan kata-kata yang dicurahkan untuk sumber itu. $\mathrm{pos}(s)$ adalah posisi kalimat s dalam tanggapan (pertama, kedua, …), dan $|S|$ adalah banyaknya kalimat, sehingga $\mathrm{pos}(s)/|S|$ bergerak dari hampir 0 di awal sampai hampir 1 di akhir. Karena itu faktor $e^{-\mathrm{pos}(s)/|S|}$ mendekati 1 untuk kalimat awal dan turun hingga sekitar $e^{-1} \approx 0.37$ untuk kalimat terakhir: penyebutan di awal dihitung hampir penuh, penyebutan di akhir sekitar sepertiganya. Penyebutnya adalah banyaknya kata dalam seluruh tanggapan, sehingga hasilnya adalah *proporsi* jawaban yang menjadi milik sumber itu, disesuaikan dengan posisi. Makalah itu menyebutnya *position-adjusted word count* (hitungan kata yang disesuaikan posisi) dan melengkapinya dengan skor *subjective impression* (kesan subjektif) yang dinilai oleh model bahasa menurut tujuh dimensi, seperti relevansi, pengaruh, dan kemungkinan pembaca akan mengeklik [2].

Contoh kecil membuat Persamaan (2.2) menjadi konkret. Misalkan sebuah jawaban memiliki $|S| = 4$ kalimat masing-masing 10 kata, total 40 kata, dan halaman binatu itu dikutip hanya oleh kalimat pertama, yang posisinya kita beri nomor $\mathrm{pos} = 0$ (konvensi penomoran posisi yang persis dalam makalah tidak mengubah gagasannya). Maka pembilangnya $10 \cdot e^{0} = 10$ dan ukurannya $10/40 = 0.25$. Jika sebaliknya binatu itu hanya dikutip oleh kalimat terakhir, $\mathrm{pos} = 3$, pembilangnya $10 \cdot e^{-3/4} \approx 10 \cdot 0.47 = 4.7$ dan ukurannya sekitar 0.12: sepuluh kata yang sama, bernilai separuhnya karena datang paling akhir.

2.5.3 Sebuah tolok ukur dan sembilan metode

Tim itu membangun **GEO-bench**, tolok ukur berisi 10.000 kueri yang diambil dari sembilan himpunan data yang sudah ada dan mencakup 25 ranah, dengan pembagian latih, validasi, dan uji sebanyak 8.000, 1.000, dan 1.000 kueri [2]. Untuk setiap kueri mereka mengambil sumber-sumber yang dikembalikan langkah pencarian, menulis ulang *satu* sumber dengan sebuah metode optimasi, membangkitkan ulang jawabannya, lalu mengukur apakah visibilitas sumber itu naik atau turun dibandingkan versi yang tidak diubah.

Sembilan metode diuji: nada yang lebih *Authoritative* (persuasif), *Statistics Addition* (penambahan statistik), *Keyword Stuffing* (penjejalan kata kunci), *Cite Sources* (mengutip sumber), *Quotation Addition* (penambahan kutipan), bahasa *Easy-to-Understand* (mudah dipahami), *Fluency*

Optimization (optimasi kelancaran), *Unique Words* (kata-kata unik), dan *Technical Terms* (istilah teknis). **Tabel 2.2** merangkum hasil utama pada hitungan kata yang disesuaikan posisi.

Tabel 2.2 Hasil terpilih makalah GEO pada hitungan kata yang disesuaikan posisi: tiga metode terbaik dan penjejalan kata kunci dibandingkan dengan dasar yang tidak diubah [2].

Metode	Apa yang diubah pada sumber	Hasil pada Imp_{pwc}
Tanpa optimasi (dasar)	tidak ada	19,3
Quotation Addition	menambahkan kutipan relevan dari sumber tepercaya	27,2 (sekitar +41%)
Statistics Addition	mengganti klaim samar dengan angka	25,2 (sekitar +32%)
Cite Sources	menambahkan rujukan ke sumber tepercaya	24,6 (sekitar +28%)
Keyword Stuffing	mengulang kata-kata kueri, trik SEO lama	17,7 (lebih buruk daripada dasar)

Keuntungan relatif di kolom terakhir berasal dari Persamaan (2.1): untuk Quotation Addition, $(27.2 - 19.3)/19.3 = 7.9/19.3 \approx 0.41$. Makalah itu merangkum temuannya sebagai peningkatan visibilitas "up to 40%", hingga 40%, dalam tanggapan mesin generatif, dan melaporkan kenaikan hingga 37% dalam uji di Perplexity, sebuah mesin komersial yang sudah beroperasi [2].

Dua pelajaran dari **Tabel 2.2** telah membentuk GEO praktis sejak itu. Metode yang berhasil menambahkan **substansi yang dapat diverifikasi**: kutipan, angka, rujukan. Metode yang gagal adalah yang mencoba mengakali mesin: mengulang kata kunci, yang dahulu membantu halaman naik peringkat, justru membuatnya *kurang* terlihat oleh model yang sedang berusaha menulis jawaban yang berguna. Bagi usaha yang jujur, ini kabar baik. Hal terbaik yang dapat dilakukan sebuah binatu untuk visibilitasnya dalam jawaban adalah menyatakan harga, jam buka, dan kebijakannya yang sebenarnya dengan tepat, dengan angka.

> **PERINGATAN: BACALAH HASIL MAKALAH SEBAGAIMANA MAKALAH MENYATAKANNYA.** GEO-bench mengukur perubahan pada satu sumber di antara beberapa sumber, pada mesin dan model tahun 2023–2024, dengan metrik makalah itu sendiri. Persentasenya adalah bukti bahwa substansi membantu dan penjejalan merugikan, bukan jaminan kenaikan tertentu pada mesin mana pun hari ini. Mesin-mesin terus mengubah temu kembali dan modelnya.

2.6 Definisi GEO Menurut Profitina

2.6.1 Dalam satu kalimat, dan dalam sebuah perbandingan

Profitina menjelaskan disiplin ini kepada penggunanya dalam dua kalimat yang layak dihafal [4]:

> **GEO, DALAM KATA-KATA PROFITINA.** "SEO optimizes your ranking on search results (a list of links). GEO optimizes so your content is understood, extracted, and cited directly inside AI answers. They complement each other." Artinya: SEO mengoptimalkan peringkat Anda di hasil pencarian (sebuah daftar tautan). GEO mengoptimalkan agar konten Anda dipahami, diekstraksi, dan dikutip langsung di dalam jawaban AI. Keduanya saling melengkapi.

Tiga kata kerja di tengahnya dipilih dengan cermat. Konten harus **dipahami**: mesin harus tahu halaman itu tentang apa dan usaha mana yang digambarkannya, dan untuk itulah data terstruktur ada. Konten harus **diekstraksi**: faktanya harus berada dalam teks yang dapat diangkat mesin, bukan dalam gambar atau skrip. Dan konten harus **dikutip**: dari semua sumber yang mengatakan hal serupa, sumber inilah yang semestinya disebut mesin. **Tabel 2.3** menjajarkan kedua disiplin itu.

Tabel 2.3 SEO dan GEO dibandingkan. Fondasinya sama; yang berbeda adalah siapa yang memilih sumber dan apa yang dihitung sebagai keberhasilan.

	SEO	GEO
Mesin mengembalikan	daftar tautan berperingkat	satu jawaban yang dibangkitkan, kadang dengan sitasi
Siapa yang memilih sumber	manusia, dengan mengeklik	mesin, sambil menulis
Satuan keberhasilan	posisi dalam daftar	sitasi atau penyebutan yang benar di dalam jawaban
Keberhasilan tampak sebagai	kunjungan ke situs	fakta usaha diulang dengan benar, entah ada yang berkunjung atau tidak
Yang paling membantu	dapat dirayapi, judul dan subjudul yang relevan, tautan dari situs lain	fondasi yang sama, ditambah teks *answer-first* yang dapat diekstraksi, fakta dan angka yang tepat, data terstruktur, dan akses crawler bagi bot AI
Yang merugikan	halaman lambat, tak dapat dirayapi, dan tipis	hal yang sama, ditambah fakta yang terkunci dalam gambar, bahasa pemasaran yang samar, penjejalan kata kunci

2.6.2 Apa yang diukur Profitina, dan apa yang tidak

AI Visibility Engine milik Profitina memperkirakan kesiapan sebuah situs untuk dikutip dari sinyal yang dapat diamatinya langsung di web publik: seberapa mudah kontennya diekstraksi, data terstruktur schema.org apa yang dipublikasikan situs itu, dan seberapa kokoh fondasi teknisnya [12]. Mesin itu menggabungkan beberapa sinyal menjadi skor kesiapan dari 0 sampai 100, memperlihatkan metodenya secara terbuka, dan mendaftar perbaikan yang diprioritaskan. Ia juga menyatakan dengan lugas apa skor itu: "Visibility results are methodological estimates, not guarantees of actual third-party AI engine output", yakni hasil visibilitas adalah perkiraan metodologis, bukan jaminan atas keluaran mesin AI pihak ketiga yang sebenarnya [12].

Yang sama pentingnya adalah apa yang *tidak* dilakukan platform ini. Profitina tidak mengukur visibilitas sebuah usaha dengan bertanya kepada ChatGPT, Gemini, atau Perplexity tentang usaha itu. Bab 1 sudah memberikan alasannya: kemandirian (pengukuran yang bergantung pada layanan pihak ketiga akan rusak ketika layanan itu mengubah ketentuannya), privasi (nama merek dan pesaing akan terkirim ke penyedia luar), dan aturan internal yang melarang API pihak ketiga berbayar. Mesin jawaban adalah *sasaran* yang kesiapannya diperkirakan, bukan komponen yang menjadi sandaran platform. Riset pada Subbab 2.5 menjelaskan mengapa pilihan ini masuk akal: faktor-faktor yang membuat sebuah sumber terlihat, yaitu substansi yang dapat diekstraksi, struktur yang jelas, dan fakta yang tepat, adalah sifat *sumber* itu, dan sifat itu dapat diamati pada sumbernya.

2.6.3 Siklusnya: Pindai, Optimasi, Pantau

Profitina menata pekerjaannya sebagai sebuah siklus dengan tiga langkah yang sama dalam kedua bahasanya: **Pindai · Optimasi · Pantau** dalam bahasa Indonesia, **Scan · Optimize · Monitor** dalam bahasa Inggris, disingkat POP dan SOM. *Pindai* memperkirakan kesiapan dan menemukan kesenjangan; *optimasi* mengubah kesenjangan menjadi perbaikan, seperti teks *answer-first*, bagian FAQ, data terstruktur, dan akses crawler; *pantau* memindai ulang menurut jadwal dan memperlihatkan trennya. Bab 3 menelusuri siklus itu layar demi layar. Lab bab ini membangun potongan terkecil dari langkah *optimasi*: data terstruktur dan berkas-berkas crawler.

2.6.4 Harapan yang jujur

Platform GEO tergoda untuk menjanjikan terlalu banyak: "jadilah jawaban yang diberikan ChatGPT", "peringkat satu di AI". Dokumen-dokumen Profitina menolak bahasa semacam itu karena alasan sederhana: tidak ada pihak di luar sebuah mesin yang mengendalikan apa yang akan dikatakannya, tidak ada papan peringkat tunggal untuk sitasi AI, dan mesin-mesin terus mengubah temu kembali dan modelnya. Yang *dapat* dilakukan sebuah platform adalah menaikkan peluang, secara sah dan terukur, dan memperlihatkan cara kerjanya. Sikap yang sama mengalir di seluruh buku ini. Ketika sebuah klaim adalah perkiraan, ia disebut perkiraan.

2.7 Lapisan yang Dapat Dibaca Mesin: Data Terstruktur, robots.txt, dan llms.txt

Halaman web ditulis untuk manusia, tetapi dibaca oleh program lebih dahulu. Tiga potong teks kecil, yang tidak satu pun terlihat di halaman itu sendiri, memberi tahu program-program itu tentang apa halaman tersebut dan apa yang boleh mereka lakukan dengannya. Bersama-sama, ketiganya adalah fondasi setiap pekerjaan GEO, dan tepat itulah yang dibangkitkan lab G6.

2.7.1 Data terstruktur: schema.org dalam JSON-LD

schema.org adalah kosakata bersama untuk menggambarkan berbagai hal di web: usaha, produk, acara, artikel, orang, pertanyaan, dan jawaban. Kosakata ini didirikan oleh mesin-mesin pencari utama dan dipelihara secara terbuka; rilisnya pada saat buku ini disusun adalah versi 30.1, tertanggal 16 September 2026 [13]. Setiap jenis hal adalah sebuah *tipe*, misalnya `LocalBusiness`, dan tipe-tipe membentuk hierarki. `DryCleaningOrLaundry`, misalnya, adalah jenis `LocalBusiness` yang lebih khusus [14]. Setiap tipe memiliki *properti*, seperti `name`, `address`, `telephone`, atau `openingHours`.

JSON-LD (JSON for Linked Data) adalah cara paling umum untuk menuliskan data schema.org ke dalam halaman [15]. Wujudnya JSON biasa, diletakkan di HTML halaman di dalam elemen `<script type="application/ld+json">`, yang tidak ditampilkan peramban tetapi dibaca crawler. Tiga kunci khusus, semuanya diawali `@`, mengubah JSON biasa menjadi data tertaut:

- `@context` menyatakan dari kosakata mana nama-nama itu berasal, di sini `https://schema.org`;
- `@type` menyatakan jenis sebuah objek, seperti `DryCleaningOrLaundry` atau `FAQPage`;
- `@id` memberi objek sebuah nama global, berupa URL, sehingga objek lain dapat merujuknya; `"about": {"@id": "https://laundry.example/#business"}` berarti "FAQ ini tentang usaha yang dinamai oleh pengenal tersebut".

Kunci keempat, `@graph`, menampung daftar beberapa objek dalam satu blok, sehingga sebuah halaman dapat menggambarkan usaha dan FAQ-nya sekaligus dan menautkannya dengan `@id`.

> **DEFINISI.** *Data terstruktur* adalah informasi tentang sebuah halaman yang ditulis dalam kosakata yang dapat dibaca mesin. Dalam *JSON-LD* wujudnya adalah objek JSON yang `@context`-nya menyebut kosakata, `@type`-nya menyebut jenis hal yang digambarkan, dan kunci-kunci lainnya adalah properti tipe tersebut.

Mengapa ini membantu mesin jawaban? Karena ia menghapus tebak-tebakan. Sebuah halaman mungkin berbunyi "Buka setiap hari kecuali Minggu sore, hubungi kami!" dengan huruf hias; JSON-LD-nya berbunyi `"openingHours": ["Mo-Sa 07:00-21:00", "Su 08:00-17:00"]` dan `"telephone": "+62-812-0000-1234"`. FAQ yang ditulis sebagai `FAQPage` dengan objek `Question` dan `Answer` menyerahkan kepada mesin persis jenis pernyataan pendek, mandiri, dan dapat

dikutip yang menurut Subbab 2.5 paling terlihat. Data terstruktur tidak menjamin bahwa sesuatu akan dikutip; ia membuat fakta-faktanya tidak ambigu bagi program mana pun yang menginginkannya.

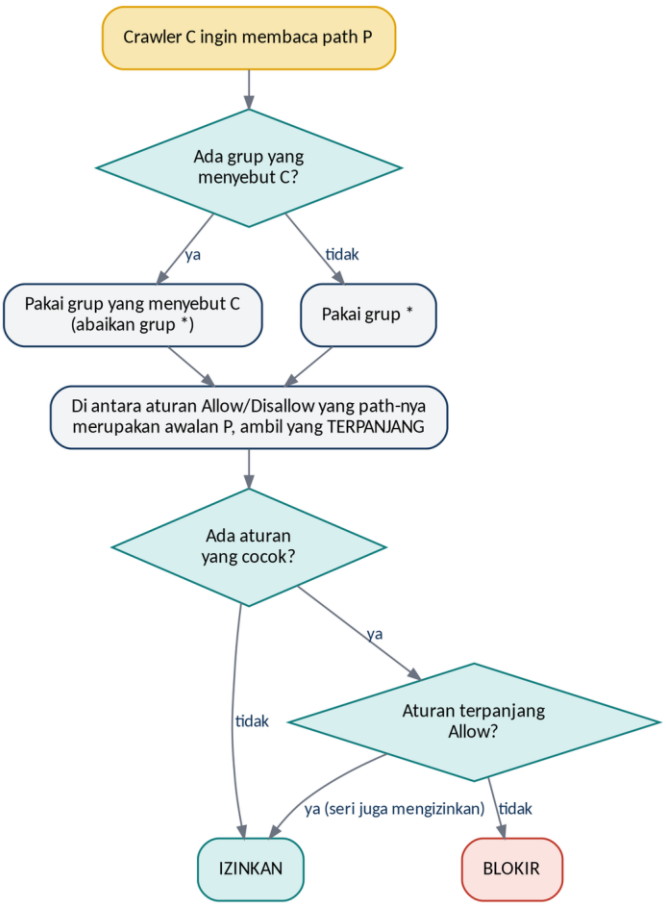
2.7.2 Jagalah kebenaran fakta

Data terstruktur adalah janji tentang halaman, dan janji itu harus cocok dengan halamannya. `priceRange` dalam JSON-LD yang berbeda dari daftar harga di halaman membingungkan setiap pembaca, manusia maupun mesin, dan pedoman mesin pencari sendiri memperlakukan data terstruktur yang tidak sesuai dengan konten yang terlihat sebagai masalah. Karena itu aturan lab G6 adalah bahwa setiap fakta berasal dari satu berkas yang dipelihara pemilik, `business.json`, dan berkas yang sama memasok teks halaman sekaligus data terstrukturnya.

2.7.3 robots.txt: siapa boleh membaca apa

`robots.txt` adalah berkas teks biasa di akar sebuah situs, `https://laundry.example/robots.txt`, yang memberi tahu crawler jalur mana saja yang boleh diambilnya. Ini salah satu konvensi tertua di web, sejak 1994, dan akhirnya dituliskan sebagai standar Internet, *Robots Exclusion Protocol*, RFC 9309, pada September 2022 [16]. Berkas itu berupa daftar **grup**. Setiap grup diawali satu atau lebih baris `User-agent:` yang menamai crawler, diikuti aturan `Allow:` dan `Disallow:`, masing-masing dengan awalan jalur. Grup khusus `User-agent: *` berlaku bagi setiap crawler yang tidak dinamai grup lain. Baris `Sitemap:`, di mana pun dalam berkas, menunjuk ke daftar halaman situs.

RFC 9309 menetapkan dengan tepat bagaimana crawler memutuskan, dan **Gambar 2.3** menggambarkan prosedurnya. Pertama, crawler mencari grup yang `User-agent`-nya menyebut dirinya; bila tidak ada, ia memakai grup `*`. Crawler yang memiliki grupnya sendiri *mengabaikan* grup `*` sepenuhnya, hal yang mengejutkan banyak pemilik situs. Kemudian, di antara aturan-aturan grup itu yang jalurnya merupakan awalan dari jalur yang diinginkannya, aturan yang **terpanjang** menang; jika aturan `Allow` dan `Disallow` sama panjang, `Allow` menang; dan jika tidak ada aturan yang cocok, jalur itu diizinkan.



Gambar 2.3 Cara crawler menerapkan `robots.txt` menurut RFC 9309: pilih grup yang menyebut dirinya (atau grup `*`), lalu biarkan aturan cocok yang terpanjang memutuskan, dengan seri dimenangkan `Allow`.

Dua peringatan patut dicatat di sini. Pertama, `robots.txt` adalah *permintaan*, bukan kunci. Crawler yang berperilaku baik menaatinya; apa pun yang bersifat pribadi harus dilindungi dengan kata sandi dan izin akses, jangan hanya dengan baris `Disallow`, yang justru *mengiklankan* jalur itu kepada siapa saja yang membaca berkasnya. Kedua, berkas itu mencocokkan *awalan jalur*, karakter demi karakter, bukan nama folder: `Disallow: /admin` memblokir `/admin/`, `/admin.html`, dan `/administrator`, dan `Allow: /admin/help` juga mengizinkan `/admin/helpdesk`. Soal 2.9 menjelajahi kedua jebakan itu.

2.7.4 Crawler-crawler AI

Setiap perusahaan AI besar menjalankan beberapa crawler dengan tugas yang berbeda, dan masing-masing memublikasikan namanya agar pemilik situs dapat memutuskan secara terpisah. **Tabel 2.4** mendaftar crawler yang dinamai lab G6, dengan uraian dari perusahaan masing-masing.

Tabel 2.4 Nama-nama crawler yang dipakai dalam `robots.txt` lab G6, dengan uraian tugas crawler dari perusahaan masing-masing.

Perusahaan	Crawler (User-agent)	Apa yang dikerjakannya, menurut perusahaan
OpenAI	`OAI-SearchBot`	Menemukan situs untuk ditampilkan dalam jawaban pencarian ChatGPT; situs yang memblokirnya tidak ditampilkan di sana [17]
OpenAI	`GPTBot`	Merayapi konten yang dapat dipakai untuk melatih model fondasi OpenAI [17]
OpenAI	`ChatGPT-User`	Mengunjungi halaman ketika permintaan pengguna ChatGPT memerlukannya; bukan perayapan otomatis [17]
Anthropic	`ClaudeBot`	Mengumpulkan konten web untuk melatih dan meningkatkan model Claude [18]
Anthropic	`Claude-SearchBot`	Merayapi untuk meningkatkan mutu hasil pencarian Claude [18]
Anthropic	`Claude-User`	Mengambil halaman ketika pengguna Claude mengajukan pertanyaan yang memerlukannya [18]
Perplexity	`PerplexityBot`	Mengindeks situs untuk dimunculkan dalam jawaban Perplexity [19]
Google	`Google-Extended`	Bukan crawler tersendiri melainkan token kendali: apakah Google boleh memakai konten untuk melatih model Gemini dan untuk *grounding* di aplikasi Gemini; tidak memengaruhi Google Search [20]
Google, Microsoft	`Googlebot`, `Bingbot`	Crawler mesin pencari klasik, yang indeksnya juga memasok fitur AI seperti AI Overviews dan Copilot

Tabel itu menunjukkan mengapa satu sakelar "izinkan AI" atau "blokir AI" terlalu kasar. Binatu yang ingin muncul dalam jawaban pencarian ChatGPT harus mengizinkan `OAI-SearchBot`; apakah ia juga mengizinkan `GPTBot`, yang mengumpulkan data pelatihan, adalah keputusan terpisah. Banyak penerbit memblokir crawler pelatihan dan mengizinkan crawler pencarian; usaha kecil yang ingin dikenal luas biasanya mengizinkan keduanya. Saran Profitina sendiri menempatkan akses crawler *pertama* di antara perbaikan teknis, karena mesin tidak dapat mengutip halaman yang tidak boleh dibacanya [12]. Bing milik Microsoft layak disebut khusus dalam saran itu: indeksnya menjadi pijakan Copilot, dan mendaftarkan situs beserta sitemap-nya di Bing Webmaster Tools adalah salah satu langkah bernilai tinggi yang paling murah bagi usaha kecil.

2.7.5 llms.txt: peta ringkas bagi model bahasa

`llms.txt` adalah yang terbaru dari ketiganya dan yang paling belum mapan. Berkas ini diusulkan pada 3 September 2024 oleh Jeremy Howard dari Answer.AI sebagai berkas di akar situs, `/llms.txt`, yang ditulis dalam Markdown, yang memberi model bahasa gambaran ringkas dan terkurasi tentang situs itu: judul H1 berisi nama situs, ringkasan satu baris dalam *block quote*, beberapa paragraf rincian penting, lalu bagian-bagian berjudul `##` yang berisi daftar tautan, masing-masing ditulis `- [judul](url): catatan` [21]. Bagian bernama `## Optional` menandai tautan yang boleh dilewati model ketika ruangnya terbatas.

Adakah yang membacanya? Jawaban jujurnya pada September 2026: belum banyak. Adopsinya tumbuh cepat. Pemindaian pada 18 September 2026 atas daftar Tranco untuk domain teratas menemukan `llms.txt` atau `llms-full.txt` pada 9,3% dari 1.000 domain teratas dan 8,3% dari 10.000 teratas [22], dan satu pelacak menghitung 8,8 kali lebih banyak situs dengan berkas itu pada Mei 2026 dibandingkan setahun sebelumnya [23]. Namun pemakaiannya jauh tertinggal. Analisis log server dari 137.000 domain menemukan bahwa 97% berkas `llms.txt` tidak menerima permintaan sama sekali pada Mei 2026, dan bahwa sebagian besar permintaan yang datang berasal dari alat audit SEO, bukan dari layanan AI [23]. Tim yang menjalankan pemindaian

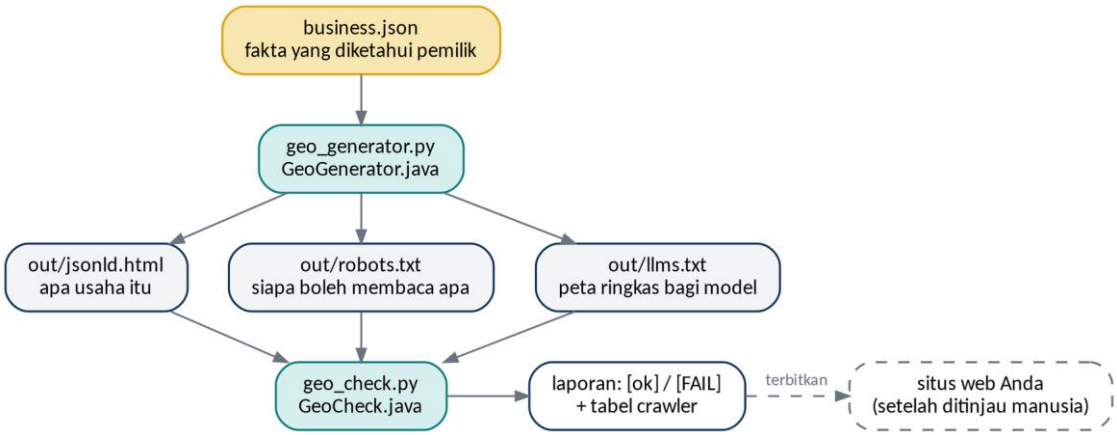
itu menambahkan bahwa memublikasikan berkas tersebut “does not establish search-ranking improvements, AI citations, or training permissions”, yakni tidak memastikan kenaikan peringkat pencarian, sitasi AI, maupun izin pelatihan [22].

> **WAWASAN KUNCI:** `llms.txt` adalah kebersihan yang murah, bukan tuas. Profitina memublikasikannya di situs pemasarannya sendiri karena biayanya kecil dan isinya menggambarkan situs dengan jujur, dan lab G6 membangkitkannya dengan alasan yang sama. Tidak satu pun mengklaim bahwa berkas itu menyebabkan sitasi. Berkas yang *benar-benar* menentukan apakah sebuah mesin boleh membaca halaman adalah `robots.txt`.

2.8 Lab G6: Generator JSON-LD, robots.txt, dan llms.txt

2.8.1 Apa yang akan kita bangun

Lab G6 mengambil fakta-fakta yang diketahui pemilik usaha lalu menghasilkan tiga potong teks yang dapat dibaca mesin dari Subbab 2.7, kemudian memeriksanya dengan membacanya kembali. **Gambar 2.4** memperlihatkan seluruh alurnya.



Gambar 2.4 Lab G6. Satu berkas fakta memasok generator yang menulis tiga berkas; pemeriksa terpisah membacanya kembali dan melapor; seorang manusia meninjau hasilnya sebelum apa pun dipublikasikan.

Rancangannya mengikuti tiga aturan, yang masing-masing sudah dimotivasi bab ini.

1. **Satu sumber kebenaran.** Setiap fakta tinggal di `business.json`. Tidak ada yang diketik dua kali, sehingga data terstruktur, berkas crawler, dan ringkasan untuk model tidak mungkin saling bertentangan.
2. **Bangkitkan, lalu verifikasi secara independen.** Pemeriksa tidak memercayai generator. Ia membaca berkas-berkas yang tertulis dari disk, menguraikannya dengan pengurai sungguhan, dan menerapkan aturan RFC 9309 serta usulan `llms.txt`. Karena itu kutu (*bug*) pada generator muncul sebagai pemeriksaan yang gagal, bukan sebagai situs web yang rusak diam-diam.
3. **Manusia yang memublikasikan.** Program-program ini menulis berkas ke folder bernama `out`; program tidak pernah mengunggah apa pun. Seorang manusia meninjaunya dan menyalinnya ke situs. `robots.txt` bahkan menyatakannya pada baris keduanya.

Kedua versi, Python dan Java, hanya memakai pustaka standar bahasanya, membaca `business.json` yang sama, dan menulis berkas yang identik sampai ke tiap byte.

2.8.2 Berkas fakta

Kode 2.1 adalah berkas fakta Profitina Laundry. Alamat, telepon, dan alamat web-nya adalah milik usaha fiktif yang dipakai di seluruh Profitina Course Library; `laundry.example` memakai

domain puncak `.example`, yang dicadangkan untuk dokumentasi dan tidak akan pernah menjadi milik situs sungguhan. Teksnya berbahasa Inggris, bahasa yang juga dipakai keluaran program.

Kode 2.1 `business.json`: fakta-fakta yang dipelihara pemilik Profitina Laundry, satu-satunya sumber bagi ketiga berkas yang dibangkitkan.

```
{
  "name": "Profitina Laundry",
  "type": "DryCleaningOrLaundry",
  "description": "Wash-and-fold, ironing and dry cleaning in Surabaya.",
  "url": "https://laundry.example",
  "telephone": "+62-812-0000-1234",
  "email": "hello@laundry.example",
  "priceRange": "Rp7,000 - Rp75,000",
  "address": {
    "street": "Jl. Penjaringan Sari No. 65, Rungkut",
    "city": "Surabaya",
    "region": "Jawa Timur",
    "postalCode": "60297",
    "country": "ID"
  },
  "openingHours": ["Mo-Sa 07:00-21:00", "Su 08:00-17:00"],
  "faq": [
    {"q": "How much does wash-and-fold cost?",
     "a": "Rp7,000 per kilogram, with a minimum of 3 kg per order."},
    {"q": "How long does a normal order take?",
     "a": "Two days. Express orders are ready the same day for 50% more."},
    {"q": "Can I pay with QRIS?",
     "a": "Yes. Every outlet accepts cash and QRIS."},
    {"q": "How do I check the status of my order?",
     "a": "Enter your bill number and phone number on the status page."}
  ],
  "pages": [
    {"title": "Services and prices", "path": "/services",
     "note": "prices of every service"},
    {"title": "Outlets", "path": "/outlets",
     "note": "addresses and opening hours"},
    {"title": "Order status", "path": "/status",
     "note": "find an order by bill number"}
  ],
  "allowCrawlers": ["Googlebot", "Bingbot", "GPTBot", "OAI-SearchBot",
                    "ChatGPT-User", "ClaudeBot", "Claude-SearchBot",
                    "Claude-User", "PerplexityBot", "Google-Extended"],
  "blockCrawlers": ["CCBot"],
  "privatePaths": ["/admin/", "/api/"]
}
```

Bidang-bidangnya terbagi dalam empat kelompok. Sepuluh yang pertama menggambarkan usaha itu sendiri. `faq` menampung pasangan pertanyaan dan jawaban, ditulis sebagaimana dianjurkan Subbab 2.5: pendek, mandiri, dengan angka. `pages` mendaftar halaman-halaman penting situs untuk `llms.txt`. Tiga bidang terakhir adalah *kebijakan*: crawler mana yang diizinkan pemilik, mana yang diblokirnya (di sini `CCBot`, crawler Common Crawl, sebagai contoh pilihan pemilik, bukan anjuran), dan jalur mana yang bersifat pribadi.

2.8.3 Generator dalam Python

Kode 2.2 membaca fakta, memeriksanya, lalu menulis ketiga berkas.

Kode 2.2 `geo_generator.py`: membaca `business.json`, memvalidasinya, lalu menulis `jsonld.html`, `robots.txt`, dan `llms.txt` ke sebuah folder.

```
# geo_generator.py - fakta usaha masuk; JSON-LD, robots.txt, llms.txt keluar
import json
import os
import sys

REQUIRED = ["name", "type", "description", "url", "telephone", "address", "faq"]


def load_facts(filename):
    with open(filename, encoding="utf-8") as f:
        facts = json.load(f)
    for field in REQUIRED:
        if not facts.get(field):
            raise ValueError(f"{filename}: field '{field}' is missing or empty")
    if not facts["url"].startswith("https://"):
        raise ValueError(f"{filename}: url must start with https://")
    facts["url"] = facts["url"].rstrip("/")      # satu ejaan, tanpa / di akhir
    return facts


def build_jsonld(facts):
    home = facts["url"] + "/"
    a = facts["address"]
    business = {
        "@type": facts["type"],
        "@id": home + "#business",
        "name": facts["name"],
        "description": facts["description"],
        "url": home,
        "telephone": facts["telephone"],
        "email": facts.get("email", ""),
        "priceRange": facts.get("priceRange", ""),
        "address": {
            "@type": "PostalAddress",
            "streetAddress": a["street"],
            "addressLocality": a["city"],
            "addressRegion": a["region"],
            "postalCode": a["postalCode"],
            "addressCountry": a["country"],
        },
        "openingHours": facts.get("openingHours", []),
    }
    questions = []
    for pair in facts["faq"]:
        questions.append({
            "@type": "Question",
```

Kode 2.2 (Lanjutan) *geo_generator.py: membaca business.json, memvalidasinya, lalu menulis jsonld.html, robots.txt, dan llms.txt ke sebuah folder.*

```
            "name": pair["q"],
            "acceptedAnswer": {"@type": "Answer", "text": pair["a"]},
        })
    faq_page = {
        "@type": "FAQPage",
        "@id": home + "#faq",
        "about": {"@id": home + "#business"},
        "mainEntity": questions,
    }
    return {"@context": "https://schema.org", "@graph": [business, faq_page]}


def jsonld_script(doc):
    text = json.dumps(doc, indent=2, ensure_ascii=False)
    text = text.replace("</", "<\\/")      # "</script>" tak dapat menutupnya dini
    return ('<script type="application/ld+json">\n' + text + "\n</script>\n")


def build_robots(facts):
    private = facts.get("privatePaths", [])
    lines = [f"# robots.txt for {facts['name']}",
             "# Generated from business.json; review before publishing.", ""]
    allowed = facts.get("allowCrawlers", [])
    if allowed:
        lines += [f"User-agent: {bot}" for bot in allowed]
        lines += [f"Disallow: {p}" for p in private] + ["Allow: /", ""]
    for bot in facts.get("blockCrawlers", []):
        lines += [f"User-agent: {bot}", "Disallow: /", ""]
    lines += ["User-agent: *"]
    lines += [f"Disallow: {p}" for p in private] + ["Allow: /", ""]
    lines += [f"Sitemap: {facts['url']}/sitemap.xml"]
    return "\n".join(lines) + "\n"


def build_llms(facts):
    a = facts["address"]
    lines = [f"# {facts['name']}", "", f"> {facts['description']}", "",
             f"Address: {a['street']}, {a['city']}, {a['region']} "
             f"{a['postalCode']}, {a['country']}.",
             f"Telephone: {facts['telephone']}. "
             f"Opening hours: {'; '.join(facts.get('openingHours', []))}.",
             "", "## Pages", ""]
    for page in facts.get("pages", []):
        lines.append(f"- [{page['title']}]({facts['url']}{page['path']}):"
                     f" {page['note']}")
    lines += ["", "## Optional", "",
```

Kode 2.2 (Lanjutan) *geo_generator.py: membaca business.json, memvalidasinya, lalu menulis jsonld.html, robots.txt, dan llms.txt ke sebuah folder.*

```
            f"- [FAQ]({facts['url']}/#faq):"
            f" {len(facts['faq'])} questions and answers"]
    return "\n".join(lines) + "\n"


def write(folder, name, text):
    path = os.path.join(folder, name)
    with open(path, "w", encoding="utf-8", newline="\n") as f:
        f.write(text)
    lines = text.count("\n")
    size = len(text.encode("utf-8"))
    print(f"Wrote {path:<18} {lines:>3} lines {size:>5} bytes")


def main():
    source = sys.argv[1] if len(sys.argv) > 1 else "business.json"
    folder = sys.argv[2] if len(sys.argv) > 2 else "out"
    try:
        facts = load_facts(source)
    except (OSError, ValueError) as e:   # JSONDecodeError juga ValueError
        print("Cannot generate:", e)
        sys.exit(1)
    print(f"Read {source}: {facts['name']}, {len(facts['faq'])} FAQ pairs,"
          f" {len(facts.get('pages', []))} pages")
    os.makedirs(folder, exist_ok=True)
    write(folder, "jsonld.html", jsonld_script(build_jsonld(facts)))
    write(folder, "robots.txt", build_robots(facts))
    write(folder, "llms.txt", build_llms(facts))


if __name__ == "__main__":
    main()
```

Telusurilah fungsi demi fungsi.

- `load_facts` membaca berkas JSON dengan `json.load` dan menolak untuk melanjutkan jika bidang yang wajib tidak ada atau kosong, atau jika alamat web tidak diawali `https://`. Generator yang diam-diam menghasilkan data terstruktur tanpa URL lebih buruk daripada tidak ada generator sama sekali. Fungsi ini juga menormalkan alamat dengan membuang garis miring di akhir, sehingga kode berikutnya dapat menambahkan `/` tepat satu kali.
- `build_jsonld` membangun dua kamus Python, usaha dan halaman FAQ, lalu membungkusnya dengan `@context` dan `@graph`. Kamus Python menjaga urutan penyisipan kunci, sehingga JSON keluar dalam urutan yang mudah dibaca dan dapat diramalkan. Properti `about` pada FAQ menunjuk ke `@id` usaha, menautkan kedua objek sebagaimana diuraikan Subbab 2.7.1.
- `jsonld_script` mengubah kamus menjadi teks dengan `json.dumps(doc, indent=2, ensure_ascii=False)`: indentasi dua spasi agar mudah dibaca, dan karakter non-ASCII dibiarkan apa adanya, tidak di-*escape*. Kemudian fungsi ini mengganti setiap `</` dengan `<\/`. Dalam JSON, `\/` hanyalah cara lain menulis `/`, sehingga datanya tidak berubah; tetapi dalam HTML, teks `</script>` di dalam sebuah string akan menutup elemen skrip terlalu dini dan merusak halaman. Satu baris ini adalah kebiasaan keamanan yang kecil tetapi nyata.
- `build_robots` menulis satu grup yang menamai setiap crawler yang diizinkan, dengan jalur pribadi dilarang dan sisanya diizinkan; lalu satu grup untuk setiap crawler yang diblokir dengan `Disallow: /`; lalu grup `*` dengan aturan yang sama seperti crawler yang diizinkan; dan akhirnya baris `Sitemap:`. Perhatikan urutan di dalam setiap grup: baris `Disallow:` yang spesifik datang *sebelum* `Allow: /`. Menurut RFC 9309 urutan tidak berpengaruh, tetapi sebagian pembaca lama

menerapkan aturan cocok yang pertama, dan urutan ini memberikan jawaban yang sama menurut kedua cara baca (Subbab 2.8.7).

- `build_llms` mengikuti struktur usulan itu: judul H1, ringkasan dalam *block quote*, baris-baris rincian, bagian `## Pages` berisi tautan, dan bagian `## Optional`.
- `write` menyimpan setiap berkas sebagai UTF-8 dengan `newline="\n"`, agar berkasnya identik juga di Windows, tempat Python sebaliknya akan menulis akhir baris `\r\n`, lalu mencetak banyaknya baris dan byte.

Jalankan di folder `Chapter02/python` dengan `python3 geo_generator.py`:

```
Read business.json: Profitina Laundry, 4 FAQ pairs, 3 pages
Wrote out/jsonld.html     70 lines  2043 bytes
Wrote out/robots.txt      26 lines   513 bytes
Wrote out/llms.txt        16 lines   560 bytes
```

Di Windows jalurnya dicetak sebagai `out\jsonld.html`, karena `os.path.join` memakai pemisah milik sistem.

2.8.4 Tiga berkas yang dibangkitkan

Blok JSON-LD, siap ditempelkan ke `<head>` halaman beranda:

```
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@graph": [
    {
      "@type": "DryCleaningOrLaundry",
      "@id": "https://laundry.example/#business",
      "name": "Profitina Laundry",
      "description": "Wash-and-fold, ironing and dry cleaning in Surabaya.",
      "url": "https://laundry.example/",
      "telephone": "+62-812-0000-1234",
      "email": "hello@laundry.example",
      "priceRange": "Rp7,000 - Rp75,000",
      "address": {
        "@type": "PostalAddress",
        "streetAddress": "Jl. Penjaringan Sari No. 65, Rungkut",
        "addressLocality": "Surabaya",
        "addressRegion": "Jawa Timur",
        "postalCode": "60297",
        "addressCountry": "ID"
      },
      "openingHours": [
        "Mo-Sa 07:00-21:00",
        "Su 08:00-17:00"
      ]
    },
    {
      "@type": "FAQPage",
      "@id": "https://laundry.example/#faq",
      "about": {
        "@id": "https://laundry.example/#business"
      },
      "mainEntity": [
        {
          "@type": "Question",
          "name": "How much does wash-and-fold cost?",
          "acceptedAnswer": {
            "@type": "Answer",
            "text": "Rp7,000 per kilogram, with a minimum of 3 kg per order."
          }
        },
        {
          "@type": "Question",
          "name": "How long does a normal order take?",
          "acceptedAnswer": {
            "@type": "Answer",
            "text": "Two days. Express orders are ready the same day for 50% more."
          }
        },
        {
          "@type": "Question",
          "name": "Can I pay with QRIS?",
```

```
          "acceptedAnswer": {
            "@type": "Answer",
            "text": "Yes. Every outlet accepts cash and QRIS."
          }
        },
        {
          "@type": "Question",
          "name": "How do I check the status of my order?",
          "acceptedAnswer": {
            "@type": "Answer",
            "text": "Enter your bill number and phone number on the status page."
          }
        }
      ]
    }
  ]
}
</script>
```

Tujuh puluh baris lebih banyak daripada yang mau diketik orang dengan tangan, dan setiap barisnya bersifat mekanis, dan itulah sebabnya generator layak ditulis. Perhatikan kedua nilai `@id` dan tautan `about` di antara keduanya, objek `PostalAddress` yang bersarang di dalam usaha, dan bahwa jawaban-jawaban FAQ adalah kalimat utuh yang memuat angka.

`robots.txt`-nya:

```
# robots.txt for Profitina Laundry
# Generated from business.json; review before publishing.

User-agent: Googlebot
User-agent: Bingbot
User-agent: GPTBot
User-agent: OAI-SearchBot
User-agent: ChatGPT-User
User-agent: ClaudeBot
User-agent: Claude-SearchBot
User-agent: Claude-User
User-agent: PerplexityBot
User-agent: Google-Extended
Disallow: /admin/
Disallow: /api/
Allow: /

User-agent: CCBot
Disallow: /

User-agent: *
Disallow: /admin/
Disallow: /api/
Allow: /

Sitemap: https://laundry.example/sitemap.xml
```

Dan `llms.txt`-nya:

```
# Profitina Laundry

> Wash-and-fold, ironing and dry cleaning in Surabaya.

Address: Jl. Penjaringan Sari No. 65, Rungkut, Surabaya, Jawa Timur 60297, ID.
Telephone: +62-812-0000-1234. Opening hours: Mo-Sa 07:00-21:00; Su 08:00-17:00.

## Pages

- [Services and prices](https://laundry.example/services): prices of every service
- [Outlets](https://laundry.example/outlets): addresses and opening hours
- [Order status](https://laundry.example/status): find an order by bill number

## Optional

- [FAQ](https://laundry.example/#faq): 4 questions and answers
```

2.8.5 Generator dalam Java

Pustaka standar Java tidak memiliki pengurai maupun pencetak JSON, sehingga versi Java dimulai dengan sebuah pengurai kecil. **Kode 2.3** panjangnya sekitar 230 baris: sebuah *pengurai turun rekursif* (*recursive-descent parser*), yang membaca sebuah nilai dengan melihat karakter pertamanya lalu memanggil metode untuk jenis nilai itu (`object()`, `array()`, `string()`, dan seterusnya), yang pada gilirannya memanggil `value()` untuk nilai-nilai yang dikandungnya; dan sebuah pencetak yang menghasilkan keluaran rapi dengan indentasi dua spasi persis seperti `json.dumps(..., indent=2)` di Python.

Kode 2.3 `MiniJson.java`: pengurai dan pencetak rapi JSON yang minimal untuk lab ini, dengan posisi galat dilaporkan sebagai baris dan kolom.

```java
// MiniJson.java - JSON secukupnya untuk lab ini: urai teks, tulis kembali
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

public class MiniJson {
    /** Dilempar untuk teks yang bukan JSON; baris dan kolom dihitung dari 1. */
    public static class JsonError extends RuntimeException {
        private static final long serialVersionUID = 1L;
        final int line;
        final int column;

        JsonError(String message, String text, int pos) {
            super(message);
            int nl = text.lastIndexOf('\n', pos - 1);
            int count = 1;
            for (int i = 0; i < pos; i++) {
                if (text.charAt(i) == '\n') {
                    count++;
                }
            }
            line = count;
            column = pos - nl;                  // nl bernilai -1 di baris pertama
        }
    }

    private final String s;
    private int pos = 0;

    private MiniJson(String text) {
        s = text;
    }

    /** Objek menjadi LinkedHashMap (urutan terjaga), larik ArrayList. */
    public static Object parse(String text) {
        MiniJson p = new MiniJson(text);
        Object value = p.value();
        p.skipSpace();
        if (p.pos != text.length()) {
            throw new JsonError("Extra data", text, p.pos);
        }
        return value;
    }
```

Kode 2.3 (Lanjutan) *MiniJson.java: pengurai dan pencetak rapi JSON yang minimal untuk lab ini, dengan posisi galat dilaporkan sebagai baris dan kolom.*

```
    private void skipSpace() {
        while (pos < s.length() && " \t\r\n".indexOf(s.charAt(pos)) >= 0) {
            pos++;
        }
    }

    private Object value() {
        skipSpace();
        if (pos >= s.length()) {
            throw new JsonError("Expecting value", s, pos);
        }
        char c = s.charAt(pos);
        if (c == '{') {
            return object();
        } else if (c == '[') {
            return array();
        } else if (c == '"') {
            return string();
        } else if (s.startsWith("true", pos)) {
            pos += 4;
            return Boolean.TRUE;
        } else if (s.startsWith("false", pos)) {
            pos += 5;
            return Boolean.FALSE;
        } else if (s.startsWith("null", pos)) {
            pos += 4;
            return null;
        } else if (c == '-' || Character.isDigit(c)) {
            int start = pos;
            while (pos < s.length()
                    && "+-0123456789.eE".indexOf(s.charAt(pos)) >= 0) {
                pos++;
            }
            return Double.parseDouble(s.substring(start, pos));
        }
        throw new JsonError("Expecting value", s, pos);
    }
```

Kode 2.3 (Lanjutan) *MiniJson.java: pengurai dan pencetak rapi JSON yang minimal untuk lab ini, dengan posisi galat dilaporkan sebagai baris dan kolom.*

```
    private Map<String, Object> object() {
        Map<String, Object> map = new LinkedHashMap<>();
        pos++;                                      // tanda '{'
        skipSpace();
        if (pos < s.length() && s.charAt(pos) == '}') {
            pos++;
            return map;
        }
        while (true) {
            skipSpace();
            if (pos >= s.length() || s.charAt(pos) != '"') {
                throw new JsonError("Expecting property name", s, pos);
            }
            String key = string();
            skipSpace();
            if (pos >= s.length() || s.charAt(pos) != ':') {
                throw new JsonError("Expecting ':' delimiter", s, pos);
            }
            pos++;
            map.put(key, value());
            skipSpace();
            if (pos < s.length() && s.charAt(pos) == ',') {
                pos++;
            } else if (pos < s.length() && s.charAt(pos) == '}') {
                pos++;
                return map;
            } else {
                throw new JsonError("Expecting ',' delimiter", s, pos);
            }
        }
    }

    private List<Object> array() {
        List<Object> list = new ArrayList<>();
        pos++;                                      // tanda '['
        skipSpace();
        if (pos < s.length() && s.charAt(pos) == ']') {
            pos++;
            return list;
        }
        while (true) {
            list.add(value());
            skipSpace();
            if (pos < s.length() && s.charAt(pos) == ',') {
                pos++;
            } else if (pos < s.length() && s.charAt(pos) == ']') {
```

Kode 2.3 (Lanjutan) *MiniJson.java: pengurai dan pencetak rapi JSON yang minimal untuk lab ini, dengan posisi galat dilaporkan sebagai baris dan kolom.*

```
                pos++;
                return list;
            } else {
                throw new JsonError("Expecting ',' delimiter", s, pos);
            }
        }
    }

    private String string() {
        StringBuilder out = new StringBuilder();
        int start = pos;
        pos++;                                  // tanda kutip pembuka
        while (pos < s.length()) {
            char c = s.charAt(pos++);
            if (c == '"') {
                return out.toString();
            }
            if (c != '\\') {
                out.append(c);
                continue;
            }
            char e = s.charAt(pos++);
            switch (e) {
                case 'n' -> out.append('\n');
                case 't' -> out.append('\t');
                case 'r' -> out.append('\r');
                case 'b' -> out.append('\b');
                case 'f' -> out.append('\f');
                case 'u' -> {
                    String hex = s.substring(pos, pos + 4);
                    out.append((char) Integer.parseInt(hex, 16));
                    pos += 4;
                }
                default -> out.append(e);       // \" \\ dan \/
            }
        }
        throw new JsonError("Unterminated string", s, start);
    }

    /** Cetak rapi, indentasi dua spasi, seperti json.dumps(indent=2) Python. */
    public static String write(Object value) {
        StringBuilder out = new StringBuilder();
        write(value, 0, out);
        return out.toString();
    }
```

Kode 2.3 (Lanjutan) *MiniJson.java: pengurai dan pencetak rapi JSON yang minimal untuk lab ini, dengan posisi galat dilaporkan sebagai baris dan kolom.*

```
    private static void write(Object v, int depth, StringBuilder out) {
        String pad = "  ".repeat(depth + 1);
        String end = "  ".repeat(depth);
        if (v instanceof Map<?, ?> map) {
            if (map.isEmpty()) {
                out.append("{}");
                return;
            }
            out.append("{\n");
            int i = 0;
            for (Map.Entry<?, ?> e : map.entrySet()) {
                out.append(pad).append(quote(e.getKey().toString())).append(": ");
                write(e.getValue(), depth + 1, out);
                out.append(++i < map.size() ? ",\n" : "\n");
            }
            out.append(end).append('}');
        } else if (v instanceof List<?> list) {
            if (list.isEmpty()) {
                out.append("[]");
                return;
            }
            out.append("[\n");
            for (int i = 0; i < list.size(); i++) {
                out.append(pad);
                write(list.get(i), depth + 1, out);
                out.append(i + 1 < list.size() ? ",\n" : "\n");
            }
            out.append(end).append(']');
        } else if (v instanceof String str) {
            out.append(quote(str));
        } else {
            out.append(v);                       // bilangan, true, false, null
        }
    }

    static String quote(String str) {
        StringBuilder out = new StringBuilder("\"");
        for (char c : str.toCharArray()) {
            switch (c) {
                case '"' -> out.append("\\\"");
                case '\\' -> out.append("\\\\");
                case '\n' -> out.append("\\n");
                case '\t' -> out.append("\\t");
                case '\r' -> out.append("\\r");
                case '\b' -> out.append("\\b");
                case '\f' -> out.append("\\f");
```

Kode 2.3 (Lanjutan) *MiniJson.java: pengurai dan pencetak rapi JSON yang minimal untuk lab ini, dengan posisi galat dilaporkan sebagai baris dan kolom.*

```
                default -> {
                    if (c < 0x20) {
                        out.append(String.format("\\u%04x", (int) c));
                    } else {
                        out.append(c);
                    }
                }
            }
        }
        return out.append('"').toString();
    }
}
```

Tiga pilihan rancangan dalam **Kode 2.3** layak diberi catatan. Objek diurai menjadi `LinkedHashMap`, yang, tidak seperti `HashMap`, mengingat urutan penyisipan kunci; tanpanya JSON yang ditulis akan keluar dalam urutan yang tak dapat diramalkan dan tidak akan sama dengan milik Python. Ketika pengurai menjumpai sesuatu yang bukan JSON, ia melempar `JsonError` yang mencatat *baris dan kolom* masalahnya, dihitung dengan cara yang sama seperti Python, sehingga pemeriksa pada Subbab 2.8.6 dapat melaporkan galat secara identik dalam kedua bahasa. Dan bilangan dibaca sebagai `double`, yang sudah cukup untuk lab ini, yang datanya tidak memuat bilangan sama sekali; pustaka serbaguna harus lebih baik, misalnya menulis `3` alih-alih `3.0`. Untuk proyek sungguhan, pustaka seperti Jackson atau Gson adalah pilihan yang tepat; di sini, menulis pengurai sekali saja adalah pelajaran tentang cara kerja setiap pustaka JSON di dalamnya.

Kode 2.4 adalah generatornya sendiri, padanan baris demi baris dari **Kode 2.2**.

Kode 2.4 `GeoGenerator.java`: generator yang sama dalam Java, memakai `MiniJson` dan peta berurutan.

```
// GeoGenerator.java - fakta usaha masuk; JSON-LD, robots.txt, llms.txt keluar
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

public class GeoGenerator {
    static final List<String> REQUIRED = List.of("name", "type", "description",
            "url", "telephone", "address", "faq");

    @SuppressWarnings("unchecked")
    static Map<String, Object> map(Object o) {
        return (Map<String, Object>) o;
    }

    @SuppressWarnings("unchecked")
    static List<Object> list(Object o) {
        return o == null ? List.of() : (List<Object>) o;
    }

    static String str(Map<String, Object> m, String key) {
        Object v = m.get(key);
        return v == null ? "" : v.toString();
    }

    static boolean empty(Object v) {            // padanan "not v" Python
        return v == null || v.equals("") || (v instanceof List<?> l && l.isEmpty())
                || (v instanceof Map<?, ?> m && m.isEmpty());
    }

    static Map<String, Object> loadFacts(String filename) throws IOException {
        String text = Files.readString(Path.of(filename), StandardCharsets.UTF_8);
        Map<String, Object> facts = map(MiniJson.parse(text));
        for (String field : REQUIRED) {
            if (empty(facts.get(field))) {
                throw new IllegalArgumentException(filename + ": field '" + field
                        + "' is missing or empty");
            }
        }
        String url = str(facts, "url");
        if (!url.startsWith("https://")) {
            throw new IllegalArgumentException(filename
```

Kode 2.4 (Lanjutan) *GeoGenerator.java: generator yang sama dalam Java, memakai MiniJson dan peta berurutan.*

```
                    + ": url must start with https://");
        }
        while (url.endsWith("/")) {              // satu ejaan, tanpa / di akhir
            url = url.substring(0, url.length() - 1);
        }
        facts.put("url", url);
        return facts;
    }

    static Map<String, Object> node(Object... keyValues) {
        Map<String, Object> m = new LinkedHashMap<>();   // menjaga urutan sisipan
        for (int i = 0; i < keyValues.length; i += 2) {
            m.put((String) keyValues[i], keyValues[i + 1]);
        }
        return m;
    }

    static Map<String, Object> buildJsonLd(Map<String, Object> facts) {
        String home = str(facts, "url") + "/";
        Map<String, Object> a = map(facts.get("address"));
        Map<String, Object> business = node(
                "@type", str(facts, "type"),
                "@id", home + "#business",
                "name", str(facts, "name"),
                "description", str(facts, "description"),
                "url", home,
                "telephone", str(facts, "telephone"),
                "email", str(facts, "email"),
                "priceRange", str(facts, "priceRange"),
                "address", node(
                        "@type", "PostalAddress",
                        "streetAddress", str(a, "street"),
                        "addressLocality", str(a, "city"),
                        "addressRegion", str(a, "region"),
                        "postalCode", str(a, "postalCode"),
                        "addressCountry", str(a, "country")),
                "openingHours", list(facts.get("openingHours")));
        List<Object> questions = new ArrayList<>();
        for (Object o : list(facts.get("faq"))) {
            Map<String, Object> pair = map(o);
            questions.add(node(
                    "@type", "Question",
                    "name", str(pair, "q"),
                    "acceptedAnswer",
                    node("@type", "Answer", "text", str(pair, "a"))));
        }
```

Kode 2.4 (Lanjutan) *GeoGenerator.java: generator yang sama dalam Java, memakai MiniJson dan peta berurutan.*

```
        Map<String, Object> faqPage = node(
                "@type", "FAQPage",
                "@id", home + "#faq",
                "about", node("@id", home + "#business"),
                "mainEntity", questions);
        return node("@context", "https://schema.org",
                "@graph", List.of(business, faqPage));
    }

    static String jsonLdScript(Map<String, Object> doc) {
        String text = MiniJson.write(doc);
        text = text.replace("</", "<\\/");   // "</script>" tak bisa menutup dini
        return "<script type=\"application/ld+json\">\n" + text + "\n</script>\n";
    }

    static String buildRobots(Map<String, Object> facts) {
        List<Object> priv = list(facts.get("privatePaths"));
        List<String> lines = new ArrayList<>(List.of(
                "# robots.txt for " + str(facts, "name"),
                "# Generated from business.json; review before publishing.", ""));
        List<Object> allowed = list(facts.get("allowCrawlers"));
        if (!allowed.isEmpty()) {
            allowed.forEach(bot -> lines.add("User-agent: " + bot));
            priv.forEach(p -> lines.add("Disallow: " + p));
            lines.addAll(List.of("Allow: /", ""));
        }
        for (Object bot : list(facts.get("blockCrawlers"))) {
            lines.addAll(List.of("User-agent: " + bot, "Disallow: /", ""));
        }
        lines.add("User-agent: *");
        priv.forEach(p -> lines.add("Disallow: " + p));
        lines.addAll(List.of("Allow: /", ""));
        lines.add("Sitemap: " + str(facts, "url") + "/sitemap.xml");
        return String.join("\n", lines) + "\n";
    }
```

Kode 2.4 (Lanjutan) *GeoGenerator.java: generator yang sama dalam Java, memakai MiniJson dan peta berurutan.*

```
    static String buildLlms(Map<String, Object> facts) {
        Map<String, Object> a = map(facts.get("address"));
        List<String> hours = new ArrayList<>();
        list(facts.get("openingHours")).forEach(h -> hours.add(h.toString()));
        List<String> lines = new ArrayList<>(List.of(
                "# " + str(facts, "name"), "", "> " + str(facts, "description"), "",
                "Address: " + str(a, "street") + ", " + str(a, "city") + ", "
                        + str(a, "region") + " " + str(a, "postalCode") + ", "
                        + str(a, "country") + ".",
                "Telephone: " + str(facts, "telephone") + ". Opening hours: "
                        + String.join("; ", hours) + ".",
                "", "## Pages", ""));
        for (Object o : list(facts.get("pages"))) {
            Map<String, Object> page = map(o);
            lines.add("- [" + str(page, "title") + "](" + str(facts, "url")
                    + str(page, "path") + "): " + str(page, "note"));
        }
        lines.addAll(List.of("", "## Optional", "",
                "- [FAQ](" + str(facts, "url") + "/#faq): "
                        + list(facts.get("faq")).size()
                        + " questions and answers"));
        return String.join("\n", lines) + "\n";
    }

    static void write(String folder, String name, String text) throws IOException {
        Path path = Path.of(folder, name);
        Files.writeString(path, text, StandardCharsets.UTF_8);
        long lines = text.chars().filter(c -> c == '\n').count();
        int size = text.getBytes(StandardCharsets.UTF_8).length;
        System.out.printf("Wrote %-18s %3d lines %5d bytes%n",
                folder + "/" + name, lines, size);
    }

    public static void main(String[] args) throws IOException {
        String source = args.length > 0 ? args[0] : "business.json";
        String folder = args.length > 1 ? args[1] : "out";
        Map<String, Object> facts;
        try {
            facts = loadFacts(source);
        } catch (IOException | RuntimeException e) {  // JsonError juga termasuk
            System.out.println("Cannot generate: " + e.getMessage());
            System.exit(1);
            return;
        }
        System.out.println("Read " + source + ": " + str(facts, "name") + ", "
                + list(facts.get("faq")).size() + " FAQ pairs, "
```

Kode 2.4 (Lanjutan) *GeoGenerator.java: generator yang sama dalam Java, memakai MiniJson dan peta berurutan.*

```
                + list(facts.get("pages")).size() + " pages");
        Files.createDirectories(Path.of(folder));
        write(folder, "jsonld.html", jsonLdScript(buildJsonLd(facts)));
        write(folder, "robots.txt", buildRobots(facts));
        write(folder, "llms.txt", buildLlms(facts));
    }
}
```

Metode-metode pembantu kecil di bagian atas menyerap perbedaan antarbahasa. `map` dan `list` mengubah (*cast*) hasil `Object` dari pengurai ke tipe yang kita ketahui; `str` mengembalikan string kosong untuk kunci yang tidak ada, seperti `facts.get(key, "")` di Python; `empty` meniru aturan Python bahwa `None`, `""`, daftar kosong, dan kamus kosong semuanya dianggap “tidak ada”. Metode

`node(Object... keyValues)` menerima argumennya berpasangan, kunci lalu nilai, dan membangun peta berurutan, yang membuat `buildJsonLd` hampir semudah dibaca seperti literal kamus Python. `Files.writeString` menulis UTF-8 dengan karakter `\n` persis seperti yang dimasukkan program, di setiap sistem operasi.

Kompilasi dan jalankan di folder `Chapter02/java`:

```
javac *.java
java GeoGenerator
```

```
Read business.json: Profitina Laundry, 4 FAQ pairs, 3 pages
Wrote out/jsonld.html     70 lines  2043 bytes
Wrote out/robots.txt      26 lines   513 bytes
Wrote out/llms.txt        16 lines   560 bytes
```

Ketiga berkasnya identik sampai ke tiap byte dengan milik versi Python; di macOS atau Linux, `cmp java/out/robots.txt python/out/robots.txt` tidak mencetak apa pun, yang berarti "tidak ada perbedaan".

2.8.6 Membacanya kembali: pemeriksa

Generator yang berkata "selesai" hanya membuktikan bahwa ia tidak macet. **Kode 2.5** membaca ketiga berkas dari disk dan memeriksanya menurut aturan Subbab 2.7, dengan mencetak `[ok]` atau `[FAIL]` untuk setiap pemeriksaan.

Kode 2.5 `geo_check.py`: mengurai JSON-LD, menerapkan `robots.txt` pada sederet crawler dan jalur, dan memeriksa struktur `llms.txt`.

```
# geo_check.py - baca kembali berkas hasil dan periksa
import json
import os
import re
import sys
import urllib.robotparser

BUSINESS_TYPES = {"LocalBusiness", "DryCleaningOrLaundry", "Store", "Restaurant"}
PATHS = ["/", "/services", "/admin/", "/api/orders"]
LINK = re.compile(r"^- \[[^\]]+\]\(https://[^)\s]+\)(: .+)?$")

results = []                                    # pasangan (lulus, pesan)


def check(passed, message):
    results.append((passed, message))
    print(("  [ok]   " if passed else "  [FAIL] ") + message)


def read(folder, name):
    with open(os.path.join(folder, name), encoding="utf-8") as f:
        return f.read()


def check_jsonld(text):
    m = re.search(r'<script type="application/ld\+json">(.*?)</script>',
                  text, re.DOTALL)
    check(m is not None, "jsonld.html contains one JSON-LD script block")
    if m is None:
        return
    try:
        doc = json.loads(m.group(1))          # "<\/" = "/" yang sah dalam JSON
    except json.JSONDecodeError as e:
        check(False, f"JSON-LD is not valid JSON (line {e.lineno},"
                     f" column {e.colno})")
        return
    check(True, "JSON-LD parses as JSON")
    check(doc.get("@context") == "https://schema.org",
          "@context is https://schema.org")
    nodes = doc.get("@graph", [])
    ids = {n.get("@id") for n in nodes}
    shops = [n for n in nodes if n.get("@type") in BUSINESS_TYPES]
    check(len(shops) == 1, f"one business node ({len(shops)} found)")
    for shop in shops:
        missing = [k for k in ("name", "url", "address", "telephone")
                   if not shop.get(k)]
```

Kode 2.5 (Lanjutan) *geo_check.py: mengurai JSON-LD, menerapkan robots.txt pada sederet crawler dan jalur, dan memeriksa struktur llms.txt.*

```
        check(not missing, f"{shop['@type']} has name, url, address, telephone"
              + (f" (missing: {', '.join(missing)})" if missing else ""))
    faqs = [n for n in nodes if n.get("@type") == "FAQPage"]
    check(len(faqs) == 1, f"one FAQPage node ({len(faqs)} found)")
    for faq in faqs:
        qs = faq.get("mainEntity", [])
        good = [q for q in qs if q.get("@type") == "Question" and q.get("name")
                and q.get("acceptedAnswer", {}).get("text")]
        check(len(qs) > 0 and len(good) == len(qs),
              f"FAQPage has {len(qs)} questions, {len(good)} with an answer")
        target = faq.get("about", {}).get("@id")
        check(target in ids, "FAQPage.about points to a node in the graph")


def check_robots(text):
    rp = urllib.robotparser.RobotFileParser()
    rp.parse(text.splitlines())
    bots = [line.split(":", 1)[1].strip() for line in text.splitlines()
            if line.lower().startswith("user-agent:")]
    bots = [b for b in bots if b != "*"] + ["SomeNewBot"]
    maps = rp.site_maps() or []
    check(len(maps) == 1, f"robots.txt names one sitemap ({len(maps)} found)")
    print()
    print((f"    {'Crawler':<18}" + "".join(f"{p:<13}" for p in PATHS)).rstrip())
    for bot in bots:
        row = ["allow" if rp.can_fetch(bot, p) else "block" for p in PATHS]
        print((f"    {bot:<18}" + "".join(f"{r:<13}" for r in row)).rstrip())
    print()
    readers = [b for b in bots if rp.can_fetch(b, "/")]
    check(len(readers) >= 1, f"{len(readers)} of {len(bots)} crawlers may read /")
    leaks = [b for b in bots if rp.can_fetch(b, "/admin/")]
    check(not leaks, "no crawler may read /admin/")


def check_llms(text):
    lines = text.splitlines()
    check(len(lines) > 0 and lines[0].startswith("# "),
          "llms.txt starts with an H1 title")
    check(any(line.startswith("> ") for line in lines),
          "llms.txt has a one-line summary (> ...)")
    items = [line for line in lines if line.startswith("- ")]
    bad = [line for line in items if not LINK.match(line)]
    check(len(items) > 0 and not bad,
          f"{len(items)} link lines, {len(bad)} malformed")
```

Kode 2.5 (Lanjutan) *geo_check.py: mengurai JSON-LD, menerapkan robots.txt pada sederet crawler dan jalur, dan memeriksa struktur llms.txt.*

```
def main():
    folder = sys.argv[1] if len(sys.argv) > 1 else "out"
    try:
        files = {n: read(folder, n) for n in ("jsonld.html", "robots.txt",
                                              "llms.txt")}
    except OSError as e:
        print("Cannot check:", e)
        sys.exit(1)
    print(f"Checking {folder}/jsonld.html")
    check_jsonld(files["jsonld.html"])
    print(f"Checking {folder}/robots.txt")
    check_robots(files["robots.txt"])
    print(f"Checking {folder}/llms.txt")
    check_llms(files["llms.txt"])
    passed = sum(1 for ok, _ in results if ok)
    print(f"\n{passed} of {len(results)} checks passed.")
    sys.exit(0 if passed == len(results) else 1)


if __name__ == "__main__":
    main()
```

`check_jsonld` menemukan blok skrip dengan ekspresi reguler. Pola `(.*?)` bersifat *non-greedy* (tidak rakus): ia mencocokkan sesedikit mungkin, sehingga berhenti di `</script>` yang pertama. Dengan bendera `re.DOTALL`, `.` juga cocok dengan pergantian baris. Blok itu lalu diurai dengan `json.loads`; jika gagal, eksepsinya membawa `lineno` dan `colno`, dan pemeriksaan melaporkannya. Lalu datang pemeriksaan struktur: `@context` yang benar, tepat satu simpul usaha dengan empat properti pentingnya, tepat satu `FAQPage` yang setiap pertanyaannya memiliki jawaban, dan tautan `about` yang menunjuk ke `@id` yang benar-benar ada dalam graf.

`check_robots` menyerahkan berkas itu kepada `urllib.robotparser` milik Python sendiri [24] dan menanyakan, untuk setiap crawler yang dinamai dalam berkas ditambah `SomeNewBot` yang tak dikenal, apakah masing-masing dari empat jalur boleh diambil. Tabel yang dicetaknya adalah keluaran paling berguna dari lab ini: dalam sekali lihat, pemilik tahu apa yang akan dilakukan setiap crawler. `check_llms` memeriksa tiga aturan struktur usulan itu, dengan ekspresi reguler untuk baris tautan: `- [`, judul tanpa `]`, `](https://`, alamat tanpa spasi atau `)`, `)`, dan secara opsional `:` diikuti catatan.

Jalankan setelah generator dengan `python3 geo_check.py`:

```
Checking out/jsonld.html
  [ok]   jsonld.html contains one JSON-LD script block
  [ok]   JSON-LD parses as JSON
  [ok]   @context is https://schema.org
  [ok]   one business node (1 found)
  [ok]   DryCleaningOrLaundry has name, url, address, telephone
  [ok]   one FAQPage node (1 found)
  [ok]   FAQPage has 4 questions, 4 with an answer
  [ok]   FAQPage.about points to a node in the graph
Checking out/robots.txt
  [ok]   robots.txt names one sitemap (1 found)

    Crawler           /           /services    /admin/      /api/orders
    Googlebot         allow       allow        block        block
    Bingbot           allow       allow        block        block
    GPTBot            allow       allow        block        block
    OAI-SearchBot     allow       allow        block        block
    ChatGPT-User      allow       allow        block        block
    ClaudeBot         allow       allow        block        block
    Claude-SearchBot  allow       allow        block        block
    Claude-User       allow       allow        block        block
    PerplexityBot     allow       allow        block        block
    Google-Extended   allow       allow        block        block
    CCBot             block       block        block        block
    SomeNewBot        allow       allow        block        block

  [ok]   11 of 12 crawlers may read /
  [ok]   no crawler may read /admin/
Checking out/llms.txt
  [ok]   llms.txt starts with an H1 title
  [ok]   llms.txt has a one-line summary (> ...)
  [ok]   4 link lines, 0 malformed

14 of 14 checks passed.
```

Bacalah tabel crawler baris demi baris dengan berpedoman pada **Gambar 2.3**. Sepuluh crawler yang dinamai menemukan grupnya sendiri dan boleh membaca semuanya kecuali jalur pribadi. `CCBot` menemukan grupnya sendiri, yang melarang `/`, awalan dari setiap jalur, sehingga ia diblokir di mana-mana. `SomeNewBot` tidak dinamai di mana pun, sehingga ia kembali ke grup `*`, yang aturannya sama dengan aturan crawler yang diizinkan. Program ini juga menetapkan *kode keluar*, 0 bila setiap pemeriksaan lulus dan 1 bila tidak, sehingga program lain, atau gerbang rilis seperti di Bab 9, dapat menjalankannya dan bertindak berdasarkan hasilnya tanpa membaca teksnya.

Pemeriksa Java memerlukan pembaca `robots.txt`-nya sendiri, dan menulisnya adalah cara paling jelas untuk memahami RFC 9309. **Kode 2.6** mengurai berkas menjadi grup-grup dan menerapkan keputusan **Gambar 2.3** dalam `canFetch`.

Kode 2.6 `Robots.java`: pembaca `robots.txt` kecil yang menerapkan pemilihan grup dan aturan cocok terpanjang RFC 9309 (tanpa *wildcard* `*` dan `$`).

```java
// Robots.java - pembaca robots.txt kecil menurut RFC 9309 (tanpa pola * atau $)
import java.util.ArrayList;
import java.util.List;

public class Robots {
    record Rule(boolean allow, String path) { }

    static class Group {
        final List<String> agents = new ArrayList<>();
        final List<Rule> rules = new ArrayList<>();
    }

    final List<Group> groups = new ArrayList<>();
    final List<String> sitemaps = new ArrayList<>();

    Robots(String text) {
        Group current = null;
        boolean lastWasAgent = false;
        for (String raw : text.split("\n")) {
            String line = raw.replaceAll("#.*", "").strip();  // buang komentar
            int colon = line.indexOf(':');
            if (colon < 0) {
                continue;
            }
            String key = line.substring(0, colon).strip().toLowerCase();
            String value = line.substring(colon + 1).strip();
            switch (key) {
                case "user-agent" -> {
                    if (current == null || !lastWasAgent) {  // grup baru dimulai
                        current = new Group();
                        groups.add(current);
                    }
                    current.agents.add(value);
                    lastWasAgent = true;
                }
                case "allow", "disallow" -> {
                    if (current != null && !value.isEmpty()) {
                        current.rules.add(new Rule(key.equals("allow"), value));
                    }
                    lastWasAgent = false;
                }
                case "sitemap" -> sitemaps.add(value);
                default -> lastWasAgent = false;
            }
        }
    }
```

Kode 2.6 (Lanjutan) *Robots.java: pembaca robots.txt kecil yang menerapkan pemilihan grup dan aturan cocok terpanjang RFC 9309 (tanpa wildcard * dan $).*

```
    // Grup yang menyebut crawler ini; bila tak ada, grup "*".
    List<Group> groupsFor(String agent) {
        List<Group> named = new ArrayList<>();
        List<Group> star = new ArrayList<>();
        for (Group g : groups) {
            for (String a : g.agents) {
                if (a.equalsIgnoreCase(agent)) {
                    named.add(g);
                } else if (a.equals("*")) {
                    star.add(g);
                }
            }
        }
        return named.isEmpty() ? star : named;
    }

    // Aturan cocok terpanjang menang; bila seri, allow menang; tanpa aturan: allow.
    boolean canFetch(String agent, String path) {
        Rule best = null;
        for (Group g : groupsFor(agent)) {
            for (Rule r : g.rules) {
                if (!path.startsWith(r.path())) {
                    continue;
                }
                int len = r.path().length();
                if (best == null || len > best.path().length()
                        || (len == best.path().length() && r.allow())) {
                    best = r;
                }
            }
        }
        return best == null || best.allow();
    }
}
```

Konstruktornya membaca baris demi baris. Ia membuang komentar dengan ekspresi reguler `#.*`, memecah setiap baris pada titik dua pertamanya, dan mengubah kuncinya menjadi huruf kecil, karena RFC 9309 membuat nama bidang tidak peka huruf besar-kecil. Variabel `lastWasAgent` menerapkan aturan pengelompokan: baris `User-agent` yang berurutan termasuk dalam satu grup, dan baris `User-agent` yang datang setelah sebuah aturan memulai grup baru. `groupsFor` memilih grup-grup yang menyebut crawler itu, tanpa memedulikan huruf besar-kecil, atau bila tidak ada, grup-grup `*`. `canFetch` menyimpan aturan terbaik sejauh ini dan menggantinya ketika ada yang lebih panjang yang cocok, atau `Allow` yang sama panjang. Satu-satunya penyederhanaan dinyatakan di baris pertama berkas: *wildcard* `*` dan `$` di dalam jalur, yang juga didefinisikan RFC 9309, tidak didukung, karena berkas-berkas lab G6 tidak memakainya.

Kode 2.7 adalah pemeriksanya sendiri, padanan Java dari **Kode 2.5**.

Kode 2.7 `GeoCheck.java`: pemeriksaan yang sama dalam Java, memakai `MiniJson` dan `Robots`.

```
// GeoCheck.java - baca kembali berkas hasil dan periksa
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class GeoCheck {
    static final Set<String> BUSINESS_TYPES =
            Set.of("LocalBusiness", "DryCleaningOrLaundry", "Store", "Restaurant");
    static final List<String> PATHS =
            List.of("/", "/services", "/admin/", "/api/orders");
    static final Pattern LINK =
            Pattern.compile("^- \\[[^\\]]+\\]\\(https://[^)\\s]+\\)(: .+)?$");
    static final Pattern SCRIPT = Pattern.compile(
            "<script type=\"application/ld\\+json\">(.*?)</script>",
            Pattern.DOTALL);

    static int passed = 0;
    static int total = 0;

    static void check(boolean ok, String message) {
        total++;
        if (ok) {
            passed++;
        }
        System.out.println((ok ? "  [ok]   " : "  [FAIL] ") + message);
    }

    static String text(Object o) {              // "" untuk nilai yang tidak ada
        return o instanceof String s ? s : "";
    }

    @SuppressWarnings("unchecked")
    static Map<String, Object> map(Object o) {
        return o instanceof Map<?, ?> ? (Map<String, Object>) o : Map.of();
    }
```

Kode 2.7 (Lanjutan) *GeoCheck.java: pemeriksaan yang sama dalam Java, memakai MiniJson dan Robots.*

```
    @SuppressWarnings("unchecked")
    static List<Object> list(Object o) {
        return o instanceof List<?> ? (List<Object>) o : List.of();
    }

    static void checkJsonLd(String html) {
        Matcher m = SCRIPT.matcher(html);
        boolean found = m.find();
        check(found, "jsonld.html contains one JSON-LD script block");
        if (!found) {
            return;
        }
        Map<String, Object> doc;
        try {
            doc = map(MiniJson.parse(m.group(1)));  // "<\/" adalah "/" yang sah
        } catch (MiniJson.JsonError e) {
            check(false, "JSON-LD is not valid JSON (line " + e.line
                    + ", column " + e.column + ")");
            return;
        }
        check(true, "JSON-LD parses as JSON");
        check("https://schema.org".equals(doc.get("@context")),
                "@context is https://schema.org");
        List<Object> nodes = list(doc.get("@graph"));
        Set<String> ids = new HashSet<>();
        List<Map<String, Object>> shops = new ArrayList<>();
        List<Map<String, Object>> faqs = new ArrayList<>();
        for (Object o : nodes) {
            Map<String, Object> n = map(o);
            ids.add(text(n.get("@id")));
            if (BUSINESS_TYPES.contains(text(n.get("@type")))) {
                shops.add(n);
            } else if ("FAQPage".equals(n.get("@type"))) {
                faqs.add(n);
            }
        }
        check(shops.size() == 1, "one business node (" + shops.size() + " found)");
        for (Map<String, Object> shop : shops) {
            List<String> missing = new ArrayList<>();
            for (String k : List.of("name", "url", "address", "telephone")) {
                Object v = shop.get(k);
                if (v == null || v.equals("") || (v instanceof Map<?, ?> mm
                        && mm.isEmpty())) {
                    missing.add(k);
                }
            }
```

Kode 2.7 (Lanjutan) *GeoCheck.java: pemeriksaan yang sama dalam Java, memakai MiniJson dan Robots.*

```
            check(missing.isEmpty(), shop.get("@type")
                    + " has name, url, address, telephone"
                    + (missing.isEmpty() ? ""
                       : " (missing: " + String.join(", ", missing) + ")"));
        }
        check(faqs.size() == 1, "one FAQPage node (" + faqs.size() + " found)");
        for (Map<String, Object> faq : faqs) {
            List<Object> qs = list(faq.get("mainEntity"));
            int good = 0;
            for (Object o : qs) {
                Map<String, Object> q = map(o);
                String answer = text(map(q.get("acceptedAnswer")).get("text"));
                if ("Question".equals(q.get("@type"))
                        && !text(q.get("name")).isEmpty() && !answer.isEmpty()) {
                    good++;
                }
            }
            check(!qs.isEmpty() && good == qs.size(),
                    "FAQPage has " + qs.size() + " questions, " + good
                    + " with an answer");
            String target = text(map(faq.get("about")).get("@id"));
            check(ids.contains(target),
                    "FAQPage.about points to a node in the graph");
        }
    }

    static String row(String first, List<String> cells) {
        StringBuilder sb = new StringBuilder(String.format("    %-18s", first));
        cells.forEach(c -> sb.append(String.format("%-13s", c)));
        return sb.toString().stripTrailing();
    }

    static void checkRobots(String txt) {
        Robots rp = new Robots(txt);
        List<String> bots = new ArrayList<>();
        for (Robots.Group g : rp.groups) {
            g.agents.stream().filter(a -> !a.equals("*")).forEach(bots::add);
        }
        bots.add("SomeNewBot");
        check(rp.sitemaps.size() == 1,
                "robots.txt names one sitemap (" + rp.sitemaps.size() + " found)");
        System.out.println();
        System.out.println(row("Crawler", PATHS));
        for (String bot : bots) {
            List<String> cells = new ArrayList<>();
            PATHS.forEach(p -> cells.add(rp.canFetch(bot, p) ? "allow" : "block"));
```

Kode 2.7 (Lanjutan) *GeoCheck.java: pemeriksaan yang sama dalam Java, memakai MiniJson dan Robots.*

```
            System.out.println(row(bot, cells));
        }
        System.out.println();
        long readers = bots.stream().filter(b -> rp.canFetch(b, "/")).count();
        check(readers >= 1,
                readers + " of " + bots.size() + " crawlers may read /");
        boolean leak = bots.stream().anyMatch(b -> rp.canFetch(b, "/admin/"));
        check(!leak, "no crawler may read /admin/");
    }

    static void checkLlms(String txt) {
        List<String> lines = txt.lines().toList();
        check(!lines.isEmpty() && lines.get(0).startsWith("# "),
                "llms.txt starts with an H1 title");
        check(lines.stream().anyMatch(l -> l.startsWith("> ")),
                "llms.txt has a one-line summary (> ...)");
        List<String> items =
                lines.stream().filter(l -> l.startsWith("- ")).toList();
        long bad = items.stream().filter(l -> !LINK.matcher(l).matches()).count();
        check(!items.isEmpty() && bad == 0,
                items.size() + " link lines, " + bad + " malformed");
    }

    static String read(String folder, String name) throws IOException {
        return Files.readString(Path.of(folder, name), StandardCharsets.UTF_8);
    }

    public static void main(String[] args) {
        String folder = args.length > 0 ? args[0] : "out";
        String html;
        String robots;
        String llms;
        try {
            html = read(folder, "jsonld.html");
            robots = read(folder, "robots.txt");
            llms = read(folder, "llms.txt");
        } catch (IOException e) {
            System.out.println("Cannot check: " + e);
            System.exit(1);
            return;
        }
        System.out.println("Checking " + folder + "/jsonld.html");
        checkJsonLd(html);
        System.out.println("Checking " + folder + "/robots.txt");
        checkRobots(robots);
        System.out.println("Checking " + folder + "/llms.txt");
```

Kode 2.7 (Lanjutan) *GeoCheck.java: pemeriksaan yang sama dalam Java, memakai MiniJson dan Robots.*

```
        checkLlms(llms);
        System.out.println("\n" + passed + " of " + total + " checks passed.");
        System.exit(passed == total ? 0 : 1);
    }
}
```

Jalankan `java GeoCheck` setelah `java GeoGenerator`, dan program ini mencetak 34 baris yang persis sama dengan pemeriksa Python, termasuk tabel crawler dan baris penutup `14 of 14 checks passed.`

2.8.7 Ketika ada yang salah

Pemeriksa membuktikan gunanya pada masukan yang buruk. Folder kode memuat dua kasus yang sengaja dirusak.

Yang pertama adalah `broken.json`, salinan berkas fakta yang alamatnya diawali `http://`, bukan `https://`. Generator menolaknya, dalam kedua bahasa, dan keluar dengan kode 1:

```
Cannot generate: broken.json: url must start with https://
```

Yang kedua adalah folder `damaged` berisi tiga berkas yang ditulis tangan dengan jenis kerusakan yang lazim di situs sungguhan: JSON-LD dengan koma di belakang properti terakhir (sah dalam JavaScript, tidak sah dalam JSON), `robots.txt` yang melarang segalanya bagi semua crawler, dan `llms.txt` tanpa penanda judul maupun ringkasan. Menjalankan `python3 geo_check.py damaged` atau `java GeoCheck damaged` mencetak:

```
Checking damaged/jsonld.html
  [ok]   jsonld.html contains one JSON-LD script block
  [FAIL] JSON-LD is not valid JSON (line 5, column 67)
Checking damaged/robots.txt
  [FAIL] robots.txt names one sitemap (0 found)

    Crawler           /           /services    /admin/      /api/orders
    SomeNewBot        block       block        block        block

  [FAIL] 0 of 1 crawlers may read /
  [ok]   no crawler may read /admin/
Checking damaged/llms.txt
  [FAIL] llms.txt starts with an H1 title
  [FAIL] llms.txt has a one-line summary (> ...)
  [FAIL] 1 link lines, 1 malformed

2 of 8 checks passed.
```

Baris 5, kolom 67 dari blok skrip adalah `}` tepat setelah koma yang berlebih: pengurai mengharapkan nama properti berikutnya tetapi menjumpai akhir objek. (Teks blok itu dimulai tepat setelah tag pembuka, pada baris tag itu sendiri, sehingga nomor barisnya sama dengan nomor baris berkas.) Setiap kegagalan lainnya menyebut aturan yang dilanggarnya. Pemilik situs yang membaca laporan ini tahu persis apa yang harus diperbaiki, dan dalam urutan apa: JSON-LD sama sekali tidak dapat dibaca, dan tidak ada crawler yang boleh membaca situs itu.

Masih ada satu hal lagi yang dapat dipelajari dari pemeriksa ini, dan hal itu menyangkut alatnya, bukan berkasnya. `urllib.robotparser` pada Python 3.11, yang dipakai di sini, tidak menerapkan aturan cocok terpanjang RFC 9309: ia menerapkan aturan *pertama* dalam grup yang cocok. Perubahan agar modul itu mengikuti RFC 9309 diusulkan untuk CPython pada September 2025 [25], sehingga versi yang lebih baru mungkin berperilaku lain. Untuk berkas yang ditulis generator kita, kedua cara baca itu sepakat, karena generator menaruh baris `Disallow:` yang sempit sebelum `Allow: /` yang luas. Untuk berkas yang menulis `Allow: /` lebih dahulu dan `Disallow: /admin/` sesudahnya, keduanya tidak sepakat: cara baca cocok-pertama membiarkan setiap crawler masuk ke `/admin/`. Soal 2.8 mengukur perbedaan itu dengan program kecil dalam kedua bahasa. Pelajarannya bersifat umum: ketika dua program harus sepakat tentang sebuah standar, ujilah keduanya pada kasus-kasus tempat aturan standar itu benar-benar berlaku.

2.9 Profitina dalam Praktik

Segala sesuatu dalam lab G6 memiliki padanan dalam produk yang berjalan, pada tingkat apa yang dikerjakannya, bukan bagaimana ia memberi skor.

Content Engine dan Technical Assistant. Setelah pemindaian, tab *Optimasi* Profitina mengubah kesenjangan visibilitas menjadi draf siap terbit: teks *answer-first* yang menyatakan jawaban langsung di kalimat-kalimat pertamanya, nama entitas, bagian FAQ, dan JSON-LD schema.org

[12]. Technical Assistant membantu dengan `robots.txt`, terutama mengizinkan crawler AI pada **Tabel 2.4**, dan dengan pengiriman sitemap ke Bing. Draf-draf itu ditulis oleh model bahasa lokal platform, dan sebuah gerbang mutu hanya menyajikan draf AI bila draf itu mengalahkan templat bawaan; setelah revisi September 2026, draf AI menang dalam 5 dari 6 kasus tolok ukur. Ketika tidak ada model yang tersedia, aplikasi menyatakannya alih-alih menyamarkan templat sebagai keluaran AI.

Kesiapan, bukan mengajukan kueri. Pemindaian memperkirakan kesiapan dari sinyal situs itu sendiri: kemudahan ekstraksi, data terstruktur, dan kekokohan teknis. Pemindaian tidak pernah mengirim nama usaha ke ChatGPT, Gemini, atau Perplexity untuk melihat apa yang mereka katakan, karena alasan-alasan pada Subbab 2.6.2. Hasilnya membawa penafian yang sama dengan Subbab 2.6.4: perkiraan, bukan jaminan [12].

Situs pemasaran Profitina sendiri. `profitina.com` mempraktikkan apa yang dianjurkan platform. Situs itu memublikasikan `/llms.txt`, data terstruktur FAQ, dan tanggal `dateModified` pada halaman-halamannya. Bab 10 menelaahnya sebagai studi kasus, termasuk pelajaran yang dipetik tim dengan cara yang pahit: jaringan penghantar konten (CDN) di depan situs pernah menyajikan `robots.txt` "terkelola" yang diam-diam melarang `GPTBot` dan `ClaudeBot`, menimpa berkas yang ditulis tim. Berkas di repositori benar; berkas yang dilihat dunia salah. Aturan yang lahir dari kejadian itu berlaku langsung pada lab G6: *selalu verifikasi URL yang hidup, dengan melewati cache apa pun*, bukan berkas yang Anda kira telah Anda publikasikan.

Siklusnya. Lab G6 adalah potongan terkecil dari *Optimasi*. Lab G1 di Bab 3 membangun pemeriksa kesiapan kecil untuk *Pindai*, yang dapat diarahkan ke berkas-berkas yang ditulis G6, dan lab G4 di Bab 9 menjadwalkan pemindaian ulang untuk *Pantau*. Lab di Bab 10 menambahkan auditor URL hidup tepat untuk masalah cache di atas.

2.10 Kesalahpahaman yang Sering Terjadi

"GEO menggantikan SEO." GEO dibangun di atasnya. Mesin tidak dapat mengutip halaman yang tidak dapat diambil crawler-nya, dan banyak mesin jawaban mengambil dari indeks pencarian klasik. Fondasi pada **Tabel 2.3** dimiliki bersama.

"Kalau saya menambahkan JSON-LD, AI akan merekomendasikan usaha saya." Data terstruktur menghapus ambiguitas; ia tidak menciptakan preferensi. Ia membuat sitasi yang benar menjadi mungkin, bukan pasti. Makalah pada Subbab 2.5 menemukan bahwa *substansi*, seperti angka, kutipan, dan rujukan, menaikkan visibilitas, dan bahwa mengakali teks justru menurunkannya.

"Penjejalan kata kunci masih berhasil, hanya lebih halus." Dalam GEO-bench, penjejalan kata kunci membuat sumber *kurang* terlihat (**Tabel 2.2**). Model yang sedang menulis jawaban tidak memerlukan pengulangan.

"`llms.txt` adalah cara AI menemukan situs saya." Pada Mei 2026, 97% berkas dalam satu sampel besar tidak menerima permintaan sama sekali [23]. Berkas itu adalah kebersihan. Berkas yang menentukan akses adalah `robots.txt`.

"`Disallow` menjaga halaman tetap pribadi." Itu adalah permintaan yang dihormati crawler yang santun, dan ia mencantumkan jalurnya secara terbuka. Halaman pribadi memerlukan autentikasi.

"Memblokir `GPTBot` menghapus situs saya dari ChatGPT." `GPTBot` mengumpulkan data pelatihan; jawaban *pencarian* ChatGPT bersandar pada `OAI-SearchBot` [17]. Keduanya dikendalikan secara terpisah, dan itulah inti **Tabel 2.4**.

"`Google-Extended` mengendalikan apakah saya muncul di AI Overviews Google." Menurut Google, token itu mengendalikan pemakaian untuk melatih model Gemini dan untuk *grounding* di aplikasi Gemini, dan "does not impact a site's inclusion in Google Search", tidak memengaruhi dimuatnya situs di Google Search [20].

"Crawler yang memiliki grupnya sendiri tetap menaati aturan `*` juga." Menurut RFC 9309 crawler itu hanya memakai grup yang menyebut dirinya. Jika grup milik sebuah crawler lupa mencantumkan `Disallow: /admin/`, aturan grup `*` tidak melindungi `/admin/` dari crawler itu.

"Turun 47% dan turun 7 poin itu sama saja." Keduanya menggambarkan data yang sama, 15% menjadi 8%, dalam satuan yang berbeda: perubahan relatif menurut Persamaan (2.1) lawan perubahan mutlak dalam poin persentase. Katakanlah mana yang Anda maksud.

"Profitina menjamin posisi dalam jawaban AI." Tidak ada yang bisa menjaminnya. Platform menyatakan hal itu dalam syarat dan ketentuannya dan dalam setiap hasilnya [12].

2.11 Ringkasan Bab

- **Mesin pencari** merayapi, mengindeks, dan memeringkat, lalu mengembalikan **daftar tautan** yang dipilih oleh *manusia*. **SEO** meraih posisi dalam daftar itu.
- **Mesin jawaban** menemukan kembali sumber-sumber lalu meminta model bahasa **membangkitkan satu jawaban**, kadang dengan sitasi; *mesin* yang memilih. Keberhasilan menjadi **sitasi**, dan pembaca mungkin tidak pernah mengunjungi situs.
- **Angka-angka** pada **Tabel 2.1** dan **Gambar 2.2**: dengan ringkasan AI Google, klik pada hasil tradisional turun dari 15% menjadi 8% kunjungan, perubahan relatif sekitar −47% menurut Persamaan (2.1) (Pew, Maret 2025); AI Overviews naik dari 15% menjadi 43% pencarian dalam setahun, dan kunjungan AI Mode dari 126 jt menjadi 279 jt (Similarweb, 2025–2026); ChatGPT melampaui 900 jt pengguna mingguan, Gemini 950 jt pengguna bulanan. Indonesia memiliki 235,3 jt pengguna internet (81,72%). Ramalan bukan pengukuran.
- **GEO** dinamai oleh Aggarwal dkk. (KDD 2024). Visibilitas di dalam jawaban diukur dengan **hitungan kata yang disesuaikan posisi** pada Persamaan (2.2). Dalam GEO-bench, kutipan, statistik, dan rujukan menaikkan visibilitas sekitar 30–40%; penjejalan kata kunci menurunkannya (**Tabel 2.2**).
- **Definisi Profitina:** SEO mengoptimalkan peringkat dalam daftar; GEO mengoptimalkan agar konten *dipahami, diekstraksi, dan dikutip* di dalam jawaban; keduanya saling melengkapi (**Tabel 2.3**). Profitina **memperkirakan kesiapan** dari sinyal yang dapat diamati dan tidak pernah mengajukan kueri kepada mesin; hasilnya perkiraan, bukan jaminan. Siklusnya adalah **Pindai · Optimasi · Pantau** (Scan · Optimize · Monitor).
- **Lapisan yang dapat dibaca mesin**: **JSON-LD** schema.org (`@context`, `@type`, `@id`, `@graph`) membuat fakta tidak ambigu; `robots.txt` (RFC 9309) menentukan crawler mana boleh membaca jalur mana, melalui pemilihan grup dan cocok terpanjang (**Gambar 2.3**), dengan nama terpisah untuk crawler pelatihan, pencarian, dan yang dipicu pengguna (**Tabel 2.4**); `llms.txt` adalah peta Markdown bagi model, banyak dipublikasikan tetapi jarang dibaca.
- **Lab G6** (**Gambar 2.4**) membangkitkan ketiganya dari satu `business.json` dan memeriksanya secara independen, dalam Python dan Java dengan berkas dan laporan yang identik; manusia yang memublikasikan.

2.12 Soal Latihan

Soal 2.1. Dengan kata-kata Anda sendiri, jelaskan perbedaan antara *merayapi*, *mengindeks*, dan *memeringkat*, dan sebutkan mana dari ketiganya yang masih diperlukan mesin jawaban dan mana yang digantikan atau diubahnya.

Soal 2.2. Dengan memakai Persamaan (2.1), hitunglah perubahan relatif untuk (a) AI Overviews dari 15% menjadi 43% pencarian; (b) kunjungan AI Mode dari 126 jt menjadi 279 jt; (c) pengguna bulanan Gemini dari 750 jt menjadi 950 jt. Untuk (a), berikan juga perubahan mutlaknya dalam poin persentase.

Soal 2.3. Sebuah jawaban memiliki $|S| = 5$ kalimat dengan 12, 8, 10, 6, dan 14 kata, dalam urutan itu, yang diberi nomor posisi 0 sampai 4. Sumber A dikutip oleh kalimat 0 dan 3; sumber B oleh kalimat 2 dan 4. Hitunglah $\mathrm{Imp}_{\mathrm{pwc}}$ pada Persamaan (2.2) untuk kedua sumber. Mana yang lebih terlihat, dan mengapa, padahal B memiliki lebih banyak kata?

Soal 2.4. Situs sebuah toko berbunyi “We are the best laundry in Surabaya! Best service, best price, best laundry Surabaya.” Tulislah ulang menjadi dua atau tiga kalimat yang mengikuti pelajaran **Tabel 2.2**, dengan memakai fakta Profitina Laundry dari **Kode 2.1**.

Soal 2.5. Golongkan setiap pernyataan sebagai *pengukuran* atau *ramalan*, dan sebutkan sumbernya: (a) “43% pencarian menampilkan AI Overview”; (b) “Volume pencarian akan turun 25% pada 2026”; (c) “97% berkas llms.txt tidak menerima permintaan sama sekali pada Mei 2026”.

Soal 2.6. Tambahkan bidang `"sameAs"` ke `business.json`, berisi daftar alamat profil media sosial binatu itu, dan ubahlah `build_jsonld` (Python) atau `buildJsonLd` (Java) agar simpul usaha memuatnya. Di bagian mana dalam simpul itu sebaiknya ia diletakkan, dan apa arti properti itu dalam schema.org?

Soal 2.7. Pemilik memutuskan untuk memblokir crawler pelatihan (`GPTBot`, `ClaudeBot`, `Google-Extended`) tetapi mengizinkan crawler pencarian dan yang dipicu pengguna. Ubahlah `business.json`, jalankan generator dan pemeriksa, lalu perlihatkan tabel crawler yang baru. Apakah `Google-Extended` benar-benar sebuah crawler? Apa yang berubah bila ia diblokir, menurut Google?

Soal 2.8. Untuk grup `User-agent: *` dengan aturan `Allow: /`, `Disallow: /admin/`, dan `Allow: /admin/help`, dalam urutan itu, tentukan dengan tangan, pertama dengan cara baca cocok-pertama lalu dengan cara baca cocok-terpanjang RFC 9309, apakah setiap jalur `/`, `/services`, `/admin/`, `/admin/help`, dan `/admin/helpdesk` diizinkan. Kemudian tulislah program dalam kedua bahasa yang mencetak kedua jawaban berdampingan.

Soal 2.9. `robots.txt` sebuah situs memuat `User-agent: *` diikuti `Disallow: /admin`. Manakah dari `/admin`, `/admin/`, `/admin.html`, `/administrator/`, dan `/Admin/` yang diblokir? Bagaimana cara memblokir tepat folder `/admin/` dan tidak ada yang lain?

Soal 2.10. Buatlah `jsonld.html` rusak yang salah satu pertanyaan FAQ-nya tidak memiliki `acceptedAnswer`, dan yang `about`-nya menunjuk ke `https://laundry.example/#shop`. Tebaklah pemeriksaan mana pada `geo_check.py` yang gagal dan apa yang dicetaknya, lalu jalankan kedua pemeriksa untuk memastikan.

Soal 2.11. Mengapa `jsonld_script` mengganti `</` dengan `<\/`? Susunlah `business.json` yang jawaban FAQ-nya akan merusak halaman tanpa baris itu, dan perlihatkan seperti apa HTML yang dihasilkan.

Soal 2.12. Laporan Similarweb menyatakan hanya 6,8% kueri ChatGPT desktop di AS yang memuat sitasi pada Mei 2026. Jelaskan, dalam satu paragraf pendek untuk pemilik usaha, mengapa hal ini membuat data terstruktur dan fakta yang dapat diekstraksi *semakin* penting, bukan sebaliknya.

2.13 Penyelesaian Soal Latihan

Penyelesaian 2.1. *Merayapi* adalah mengambil halaman dengan mengikuti tautan; *mengindeks* adalah mencatat kata dan fakta apa muncul di halaman mana agar dapat ditemukan dengan cepat; *memeringkat* adalah mengurutkan halaman-halaman yang cocok dengan sebuah kueri. Mesin jawaban masih memerlukan perayapan dan pengindeksan, miliknya sendiri atau milik mitra, karena langkah temu kembalinya hanya dapat mengembalikan apa yang sudah diambil dan diindeks. Yang diubahnya adalah pemeringkatan: alih-alih menyajikan daftar berurutan kepada manusia, mesin menyerahkan cuplikan-cuplikan teratas kepada model bahasa, yang *memilih* apa yang dipakai lalu menulis satu jawaban. Pemeringkatan tetap penting, karena hanya cuplikan yang ditemukan kembali yang dapat dikutip, tetapi kata akhir ada pada model.

Penyelesaian 2.2. (a) $(43-15)/15 = 28/15 \approx 1.87$, naik sekitar 187%, dan secara mutlak naik 28 poin persentase. (b) $(279-126)/126 = 153/126 \approx 1.21$, sekitar +121%, lebih dari dua kali lipat. (c) $(950-750)/750 = 200/750 \approx 0.27$, sekitar +27% dalam dua kuartal.

Penyelesaian 2.3. Totalnya $12+8+10+6+14 = 50$ kata dan $|S| = 5$. Bobotnya adalah $e^{-0/5} = 1$, $e^{-1/5} \approx 0.819$, $e^{-2/5} \approx 0.670$, $e^{-3/5} \approx 0.549$, dan $e^{-4/5} \approx 0.449$. Untuk A: $12 \cdot 1 + 6 \cdot 0.549 = 12 + 3.29 = 15.29$, jadi $\mathrm{Imp}_{\mathrm{pwc}}(A) \approx 15.29/50 \approx 0.31$. Untuk B: $10 \cdot 0.670 + 14 \cdot 0.449 = 6.70 + 6.29 = 12.99$, jadi $\mathrm{Imp}_{\mathrm{pwc}}(B) \approx 12.99/50 \approx 0.26$. A lebih terlihat walaupun B memiliki 24 kata dibandingkan 18 kata milik A, karena penyebutan utama A adalah kalimat *pertama*, dengan bobot penuh, sedangkan kata-kata B datang di paruh kedua, tempat faktor eksponensialnya sudah turun di bawah 0,7.

Penyelesaian 2.4. Misalnya: “Profitina Laundry in Rungkut, Surabaya, washes and folds clothes for Rp7,000 per kilogram, with a minimum of 3 kg per order. Normal orders are ready in two days; express orders are ready the same day for 50% more. All three outlets accept cash and QRIS.” Tulisan ulang ini mengganti kata-kata superlatif, yang tidak dapat diverifikasi mesin mana pun, dengan statistik (harga, minimum, waktu), menyebut entitas dan tempatnya di kata-kata pertama, dan memberikan kalimat mandiri yang dapat dikutip satu per satu.

Penyelesaian 2.5. (a) Pengukuran, data Similarweb yang dilaporkan TechCrunch pada Juli 2026 [6]. (b) Ramalan, Gartner, Februari 2024 [11]. (c) Pengukuran, analisis log server 137.000 domain, dilaporkan pada Juli 2026 [23].

Penyelesaian 2.6. Di `business.json` tambahkan, misalnya, `"sameAs": ["https://www.instagram.com/profitinalaundry.example", "https://www.facebook.com/profitinalaundry.example"]`. Di Python, tambahkan baris `"sameAs": facts.get("sameAs", []),` ke kamus `business` setelah `"openingHours"`; di Java, tambahkan `"sameAs", list(facts.get("sameAs"))` sebagai pasangan berikutnya dalam pemanggilan `node(...)` (dengan koma setelah pasangan `openingHours`). Posisi mana pun di dalam simpul usaha adalah JSON-LD yang sah; meletakkannya paling akhir membuat keluaran yang sudah ada di atasnya tidak berubah. Dalam schema.org, `sameAs` memberikan URL halaman lain yang secara tegas mengenali entitas yang sama, yang membantu mesin menghubungkan situs web, profil media sosial, dan daftar di peta sebagai satu usaha.

Penyelesaian 2.7. Tetapkan `"allowCrawlers": ["Googlebot", "Bingbot", "OAI-SearchBot", "ChatGPT-User", "Claude-SearchBot", "Claude-User", "PerplexityBot"]` dan `"blockCrawlers": ["GPTBot", "ClaudeBot", "Google-Extended", "CCBot"]`. Generator lalu menulis satu grup berisi tujuh crawler yang diizinkan dan empat grup pemblokiran satu baris; dalam tabel pemeriksa keempat crawler yang diblokir menampilkan `block` di setiap kolom, dan baris pemeriksaannya menjadi `8 of 12 crawlers may read /` (tujuh yang dinamai ditambah `SomeNewBot`). `Google-Extended` bukan crawler yang mengambil halaman; ia adalah *token produk* yang dihormati crawler biasa Google. Memblokirnya memberi tahu Google agar tidak memakai konten untuk melatih model Gemini atau untuk *grounding* di aplikasi Gemini; menurut Google hal itu tidak memengaruhi dimuatnya atau peringkat situs di Google Search [20].

Penyelesaian 2.8. Dengan tangan, cocok-pertama (aturan pertama yang jalurnya merupakan awalan akan menang): `Allow: /` adalah awalan dari *setiap* jalur, sehingga setiap jalur diizinkan dan aturan-aturan sesudahnya tidak pernah tercapai. Cocok-terpanjang: `/` dan `/services` hanya cocok dengan `Allow: /`, sehingga diizinkan; `/admin/` cocok dengan `Allow: /` (panjang 1) dan `Disallow: /admin/` (panjang 7), dan yang lebih panjang memblokirnya; `/admin/help` juga cocok dengan `Allow: /admin/help` (panjang 11), yang terpanjang, sehingga diizinkan; `/admin/helpdesk` juga diawali `/admin/help`, sehingga ikut diizinkan, pengingat bahwa aturan mencocokkan awalan, bukan nama folder. **Kode 2.8** dan **Kode 2.9** mencetak kedua cara baca.

Kode 2.8 `sol_rule_order.py`: cara baca cocok-pertama dan cocok-terpanjang untuk satu grup `robots.txt`, berdampingan.

```python
# sol_rule_order.py - first-match lawan longest-match pada satu grup robots.txt
RULES = [("allow", "/"), ("disallow", "/admin/"), ("allow", "/admin/help")]
PATHS = ["/", "/services", "/admin/", "/admin/help", "/admin/helpdesk"]


def first_match(path):
    for kind, prefix in RULES:              # urutan berkas yang menentukan
        if path.startswith(prefix):
            return kind
    return "allow"


def longest_match(path):
    best = None
    for kind, prefix in RULES:              # RFC 9309: awalan terpanjang menentukan
        if path.startswith(prefix):
            if (best is None or len(prefix) > len(best[1])
                    or (len(prefix) == len(best[1]) and kind == "allow")):
                best = (kind, prefix)
    return "allow" if best is None else best[0]


def main():
    for kind, prefix in RULES:
        print(f"{kind.capitalize()}: {prefix}")
    print()
    print(f"{'Path':<18}{'first match':<14}longest match")
    for p in PATHS:
        a, b = first_match(p), longest_match(p)
        flag = "  <- differ" if a != b else ""
        print(f"{p:<18}{a:<14}{b:<14}".rstrip() + flag)


if __name__ == "__main__":
    main()
```

Kode 2.9 `SolRuleOrder.java`: perbandingan yang sama dalam Java.

```java
// SolRuleOrder.java - first-match lawan longest-match pada satu grup robots.txt
import java.util.List;

public class SolRuleOrder {
    record Rule(String kind, String prefix) { }

    static final List<Rule> RULES = List.of(new Rule("allow", "/"),
            new Rule("disallow", "/admin/"), new Rule("allow", "/admin/help"));
    static final List<String> PATHS =
            List.of("/", "/services", "/admin/", "/admin/help",
                    "/admin/helpdesk");

    static String firstMatch(String path) {
        for (Rule r : RULES) {                    // urutan berkas yang menentukan
            if (path.startsWith(r.prefix())) {
                return r.kind();
            }
        }
        return "allow";
    }

    static String longestMatch(String path) {
        Rule best = null;
        for (Rule r : RULES) {                    // RFC 9309: awalan terpanjang
            if (!path.startsWith(r.prefix())) {
                continue;
            }
            int len = r.prefix().length();
            if (best == null || len > best.prefix().length()
                    || (len == best.prefix().length()
                        && r.kind().equals("allow"))) {
                best = r;
            }
        }
        return best == null ? "allow" : best.kind();
    }
```

Kode 2.9 (Lanjutan) *SolRuleOrder.java: perbandingan yang sama dalam Java.*

```java
    public static void main(String[] args) {
        for (Rule r : RULES) {
            String k = r.kind();
            System.out.println(Character.toUpperCase(k.charAt(0)) + k.substring(1)
                    + ": " + r.prefix());
        }
        System.out.println();
        System.out.printf("%-18s%-14s%s%n", "Path", "first match", "longest match");
        for (String p : PATHS) {
            String a = firstMatch(p);
            String b = longestMatch(p);
            String flag = a.equals(b) ? "" : "  <- differ";
            System.out.println(String.format("%-18s%-14s%-14s", p, a, b)
                    .stripTrailing() + flag);
        }
    }
}
```

Keduanya mencetak:

```
Allow: /
Disallow: /admin/
Allow: /admin/help

Path              first match   longest match
/                 allow         allow
/services         allow         allow
/admin/           allow         disallow  <- differ
/admin/help       allow         allow
/admin/helpdesk   allow         allow
```

Hanya `/admin/` yang berbeda, dan perbedaannya ke arah yang paling buruk: cara baca cocok-pertama membuka area pribadi. Itulah sebabnya lab G6 menulis aturan yang sempit lebih dahulu.

Penyelesaian 2.9. `Disallow: /admin` memblokir setiap jalur yang *diawali* enam karakter `/admin`: `/admin`, `/admin/`, `/admin.html`, dan `/administrator/`. Aturan itu tidak memblokir `/Admin/`, karena pencocokan jalur peka huruf besar-kecil. Untuk memblokir tepat folder itu, tulislah `Disallow: /admin/`, dengan garis miring di akhir; aturan ini memblokir `/admin/` dan segala sesuatu di bawahnya, tetapi tidak `/admin.html` atau `/administrator/`. (Jalur `/admin` tanpa garis miring lalu tetap diizinkan; jika server mengalihkannya ke `/admin/`, crawler akan berhenti di sana.)

Penyelesaian 2.10. Pemeriksaan `JSON-LD parses as JSON`, `@context`, `one business node`, dan pemeriksaan properti usaha tetap lulus. Pemeriksaan FAQ gagal dan mencetak `[FAIL] FAQPage has 4 questions, 3 with an answer`, karena pertanyaan tanpa `acceptedAnswer` tidak dihitung baik. Pemeriksaan tautan gagal dengan `[FAIL] FAQPage.about points to a node in the graph`, karena `#shop` bukan `@id` simpul mana pun. Kedua pemeriksa mencetak baris yang sama, karena menerapkan aturan yang sama, dan keluar dengan kode 1.

Penyelesaian 2.11. Di dalam elemen HTML `<script>` peramban tidak memahami JSON; ia hanya mencari karakter `</script>` untuk mengetahui di mana elemen itu berakhir. Jawaban FAQ seperti `"Type </script><script>alert(1)</script> to test"` karenanya akan menutup blok JSON-LD di tengah-tengah sebuah string, dan sisanya akan dibaca sebagai HTML, paling ringan halaman rusak dan paling parah skrip yang disusupkan. Dengan penggantian itu teksnya menjadi `"Type <\/script><script>alert(1)<\/script> to test"`: tetap string JSON yang sama bagi pengurai JSON mana pun, karena `\/` berarti `/`, tetapi tidak lagi menjadi tag penutup bagi peramban. Berkas fakta pemilik seharusnya tidak pernah memuat teks semacam itu, tetapi generator harus tetap aman meskipun masukannya tidak.

Penyelesaian 2.12. Ketika sebagian besar jawaban tidak menyebut sumber, segelintir jawaban yang menyebut sumber memilihnya dengan cermat, dan mesin hanya dapat menyebut apa yang dapat dibaca dan dipahaminya. Jika jam buka, harga, dan alamat Anda dinyatakan dengan lugas di halaman dan diulang dalam data terstruktur, mesin yang memutuskan untuk bersikap spesifik dapat bersikap spesifik *tentang Anda*, dan dengan benar. Jika semua itu tersembunyi di gambar atau bertentangan dengan daftar lama, bahkan mesin yang ingin menyebut Anda dapat keliru menyebut faktanya atau memilih pesaing yang faktanya lebih jelas. Dan jawaban yang tidak mengutip apa pun tetap dibangun dari apa yang dibaca mesin: fakta yang benar dan dapat diekstraksi itulah yang membuat jawaban tanpa sitasi tersebut menyatakan hal yang benar tentang usaha Anda.

Daftar Pustaka

[1] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 107–117, 1998.

[2] P. Aggarwal, V. Murahari, T. Rajpurohit, A. Kalyan, K. Narasimhan, and A. Deshpande, "GEO: Generative Engine Optimization," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD '24)*, 2024. arXiv:2311.09735. [Online]. Available: https://arxiv.org/abs/2311.09735

[3] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.

[4] Profitina, "Profitina — GEO platform," marketing site. [Online]. Available: https://profitina.com (diakses 24 Sep. 2026).

[5] A. Chapekis and A. Lieb, "Google users are less likely to click on links when an AI summary appears in the results," Pew Research Center, Jul. 22, 2025. [Online]. Available: https://www.pewresearch.org/short-reads/2025/07/22/google-users-are-less-likely-to-click-on-links-when-an-ai-summary-appears-in-the-results/

[6] S. Perez, "Google's AI search is rapidly becoming the default, new data shows," *TechCrunch*, Jul. 27, 2026 (data Similarweb). [Online]. Available: https://techcrunch.com/2026/07/27/googles-ai-search-is-rapidly-becoming-the-default-new-data-shows/

[7] Search Engine Land, "OpenAI: ChatGPT now has 900 million weekly active users," Feb. 2026. [Online]. Available: https://searchengineland.com/chatgpt-900-million-weekly-active-users-470492

[8] T. Roush, "950 million people now use Gemini each month as Alphabet posts earnings beat," *Forbes*, Jul. 22, 2026. [Online]. Available: https://www.forbes.com/sites/tylerroush/2026/07/22/950-million-people-now-use-gemini-each-month-as-alphabet-posts-earnings-beat/

[9] Perplexity AI Magazine, "Perplexity AI monthly queries," laporan sekunder atas pernyataan A. Srinivas's statement at Bloomberg Tech, Jun. 5, 2025. [Online]. Available: https://perplexityaimagazine.com/perplexity-hub/perplexity-ai-monthly-queries/

[10] ANTARA, "Survei APJII: Penetrasi internet di Indonesia 2026 capai 81,7 persen," May 21, 2026. [Online]. Available: https://www.antaranews.com/berita/5576225/survei-apjii-penetrasi-internet-di-indonesia-2026-capai-817-persen

[11] Gartner, "Gartner predicts search engine volume will drop 25% by 2026, due to AI chatbots and other virtual agents," press release, Feb. 19, 2024. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2024-02-19-gartner-predicts-search-engine-volume-will-drop-25-percent-by-2026-due-to-ai-chatbots-and-other-virtual-agents

[12] Profitina, *Profitina Whitepaper — Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[13] Schema.org, "Schema.org releases: Version 30.1," Sep. 16, 2026. [Online]. Available: https://schema.org/docs/releases.html

[14] Schema.org, "DryCleaningOrLaundry." [Online]. Available: https://schema.org/DryCleaningOrLaundry (diakses Sep. 2026).

[15] W3C, "JSON-LD 1.1: A JSON-based Serialization for Linked Data," W3C Recommendation, Jul. 16, 2020. [Online]. Available: https://www.w3.org/TR/json-ld11/

[16] M. Koster, G. Illyes, H. Zeller, and L. Sassman, "Robots Exclusion Protocol," IETF RFC 9309, Sep. 2022.

[17] OpenAI, "Overview of OpenAI crawlers." [Online]. Available: https://developers.openai.com/api/docs/bots (diakses Sep. 2026).

[18] Anthropic, "Does Anthropic crawl data from the web, and how can site owners block the crawler?" [Online]. Available: https://support.claude.com/en/articles/8896518 (diakses Sep. 2026).

[19] Perplexity, "Perplexity crawlers." [Online]. Available: https://docs.perplexity.ai/docs/resources/perplexity-crawlers (diakses Sep. 2026).

[20] Google, "Google's common crawlers," *Google Search Central*. [Online]. Available: https://developers.google.com/search/docs/crawling-indexing/google-common-crawlers (diakses Sep. 2026).

[21] J. Howard, "/llms.txt—a proposal to provide information to help LLMs use websites," Answer.AI, Sep. 3, 2024. [Online]. Available: https://www.answer.ai/posts/2024-09-03-llmstxt.html

[22] Rankability, "llms.txt adoption," scan of Sep. 18, 2026 (Tranco list of Sep. 17, 2026). [Online]. Available: https://www.rankability.com/data/llms-txt-adoption/

[23] PPC Land, "llms.txt adoption rises 8.8x but 97% of files get zero AI requests," Jul. 2, 2026. [Online]. Available: https://ppc.land/llms-txt-adoption-rises-8-8x-but-97-of-files-get-zero-ai-requests/

[24] Python Software Foundation, "urllib.robotparser — Parser for robots.txt," *The Python Standard Library*. [Online]. Available: https://docs.python.org/3/library/urllib.robotparser.html

[25] Python Software Foundation, "Support RFC 9309 in robotparser," CPython issue #138907, Sep. 15, 2025. [Online]. Available: https://github.com/python/cpython/issues/138907

Bab 3

Apa yang Dikerjakan Profitina: Siklus Visibilitas dan Perjalanan Pengguna

Setelah menyelesaikan bab ini, Anda akan mampu menjelaskan metode kerja Profitina sebagai satu siklus dengan empat kata kerja, *ukur, jelaskan, perbaiki, pantau*, dan memetakan keempatnya ke tiga tab yang benar-benar dilihat pengguna: **Pindai · Optimasi · Pantau** dalam bahasa Indonesia dan **Scan · Optimize · Monitor** dalam bahasa Inggris. Anda akan mengetahui apa yang diperiksa oleh sebuah pindaian dan apa yang sengaja tidak dilakukannya, bagaimana sebuah angka tunggal berubah menjadi daftar perbaikan yang dapat dikerjakan pemilik toko, apa yang dihasilkan Mesin Konten dan Pendamping Teknis, serta mengapa pemantauan mengubah potret sesaat menjadi bukti. Anda akan berkenalan dengan dua laporan yang membawa siklus yang sama dari situs web ke dunia ilmiah, yaitu Laporan Visibilitas Peneliti dan Visibilitas Organisasi, serta dengan alat distribusi bernama *Satu Teks, Banyak Platform*. Sesudah itu Anda akan mengikuti pemilik sebuah usaha kecil fiktif, Profitina Laundry di Surabaya, menempuh seluruh perjalanannya, dari kunjungan pertama hingga saat ia mengekspor atau menghapus datanya sendiri, layar demi layar, dalam *wireframe* (rangka tampilan) yang digambar khusus untuk buku ini. Terakhir, Anda akan membangun lab G1, program unggulan buku ini: pemeriksa kesiapan AI mini yang mengambil sebuah halaman, `robots.txt`, dan `llms.txt`-nya dari peladen lokal yang mungil, lalu memeriksa data terstruktur, struktur judul, banyaknya teks yang dapat diekstraksi, serta tag meta dasar, dan melaporkan delapan pemeriksaan beserta sebuah skor ajar yang transparan. Skor itu milik kita sendiri, dibangun secara terbuka agar setiap aturannya dapat Anda lihat dan ubah; skor itu bukan skor Profitina. Lab ini ditulis dalam Python dan Java, dan kedua versinya mencetak laporan yang persis sama.

3.1 Posisi Kita Sekarang

Bab 2 menjelaskan mengapa *ditemukan* di internet kini tidak lagi sama dengan *berperingkat*. Mesin pencari mengembalikan daftar tautan dan membiarkan manusia memilih; mesin jawaban membaca sumber-sumber lalu menulis satu jawaban, dan di dalam jawaban itu sebuah usaha entah dikutip entah tidak muncul sama sekali. Penelitian menamai disiplin baru ini *Generative Engine Optimization*, GEO, dan mengukur bahwa sumber yang ditulis dengan substansi yang tepat dan mudah dikutip lebih terlihat di dalam jawaban yang dihasilkan [1]. Bab 2 juga memperkenalkan lapisan yang dapat dibaca mesin, titik awal setiap upaya GEO, yaitu data terstruktur schema.org dalam JSON-LD, `robots.txt`, dan `llms.txt`, dan lab G6-nya *membangkitkan* ketiga bagian itu untuk Profitina Laundry.

Bab ini beralih dari disiplin ke produk. Profitina menyebut dirinya platform manajemen dan optimalisasi visibilitas AI (GEO) di Indonesia, dan merumuskan tugasnya dalam satu kalimat: ia mengukur seberapa siap sebuah merek, situs web, peneliti, atau organisasi untuk dikutip oleh mesin jawaban AI seperti ChatGPT, Gemini, dan Perplexity, berdasarkan sinyal nyata yang dapat diamati, lalu membantu memperbaiki apa yang ditemukannya [2]. Kalimat itu memuat seluruh isi bab ini. *Mengukur* adalah pindaian. *Berdasarkan sinyal yang dapat diamati* adalah sebuah keputusan rancangan yang membawa konsekuensi. *Membantu memperbaiki* adalah Mesin Konten

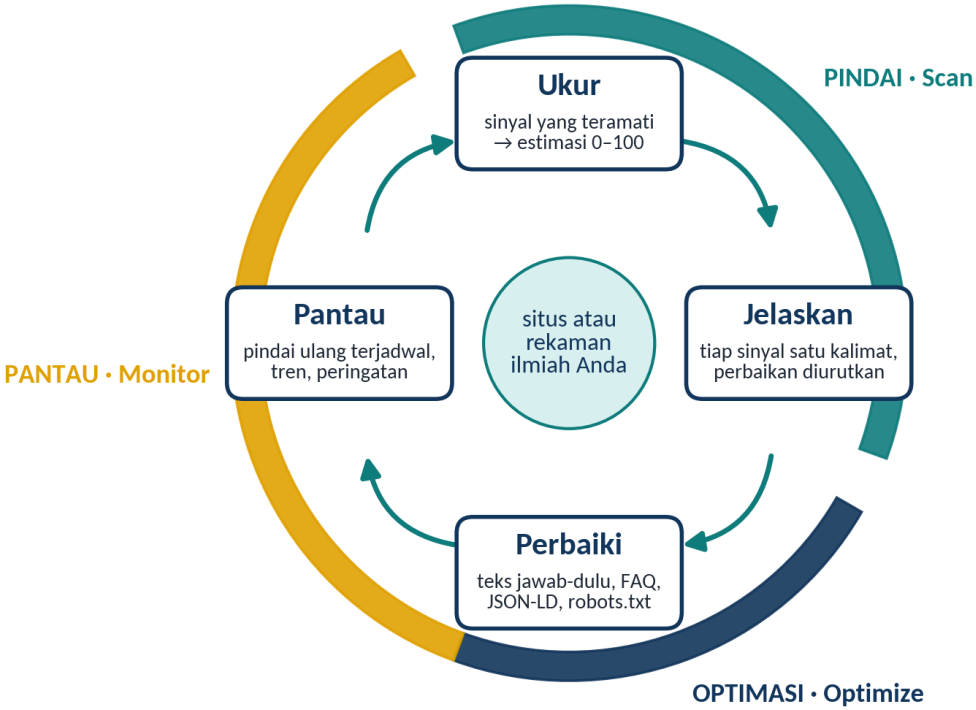
dan Pendamping Teknis. Kata *siap* pun dipilih dengan cermat: Profitina memperkirakan kesiapan, bukan menjanjikan hasil. Syarat dan ketentuannya sendiri menyatakannya terang-terangan: "Visibility results are methodological estimates, not guarantees of actual third-party AI engine output" (hasil visibilitas adalah estimasi metodologis, bukan jaminan atas keluaran nyata mesin AI pihak ketiga) [2], [3].

Kita akan melangkah dalam tiga gerakan. Subbab 3.2 sampai 3.8 menguraikan *apa* yang dikerjakan produk, fitur demi fitur, sesuai urutan pengguna menjumpainya. Subbab 3.9 lalu memperlihatkan fitur-fitur itu *dalam pemakaian*, sebagai perjalanan sembilan langkah dalam kehidupan satu usaha kecil. Subbab 3.10 adalah lab: setelah memahami kegunaan pindaian kesiapan, kita menulis versi kecilnya sendiri agar gagasan-gagasan itu berhenti menjadi abstrak. Ketika buku ini menggambarkan Profitina, yang digambarkan adalah *apa* dan *mengapa*-nya pada tingkat kotak dan panah; ketika ingin memperlihatkan *bagaimana*-nya, buku ini membangun versi ajarnya sendiri, dengan aturan yang kita pilih dan kita umumkan.

3.2 Siklus Visibilitas

3.2.1 Empat kata kerja, tiga tab

Setiap fitur dalam bab ini melayani salah satu dari empat kata kerja, dan keempatnya membentuk siklus. **Gambar 3.1** menggambarkannya.



Gambar 3.1 Siklus visibilitas: ukur, jelaskan, perbaiki, dan pantau, dikelompokkan ke dalam tiga tab aplikasi, Pindai, Optimasi, dan Pantau (Scan, Optimize, Monitor).

Ukur. Amati hal yang hendak dikutip, entah situs web usaha entah rekaman ilmiah, dan kumpulkan sinyal yang dapat diamati secara langsung: bolehkah *crawler* (perayap) AI membaca situs itu, apakah situs itu menerbitkan data terstruktur, apakah kontennya ditulis sedemikian rupa sehingga sebuah program dapat mengambil jawaban yang utuh darinya? Gabungkan sinyal-sinyal itu menjadi estimasi kesiapan pada skala 0 sampai 100.

Jelaskan. Angka saja tidak menolong siapa pun. Ubah setiap sinyal menjadi kalimat yang dipahami orang awam ("dua *crawler* AI diblokir oleh `robots.txt` Anda"), lalu urutkan perbaikan yang dihasilkan sehingga yang paling berguna berada di atas.

Perbaiki. Tutup celah-celahnya. Sebagian perbaikan berupa teks: paragraf pembuka yang menyebut siapa usaha itu dan apa yang dikerjakannya, FAQ berisi pertanyaan yang benar-benar diajukan pelanggan. Sebagian lagi bersifat teknis: blok JSON-LD, `robots.txt` yang mengizinkan *crawler* yang dikehendaki pemilik, dan sitemap yang didaftarkan ke mesin pencari. Profitina menyusun drafnya; pemilik usaha menerbitkannya di situsnya sendiri.

Pantau. Pindai lagi secara terjadwal dan simpan riwayatnya. Satu pindaian adalah sebuah foto; deret mingguan adalah bukti bahwa perbaikan berhasil, atau peringatan bahwa ada sesuatu yang rusak.

Aplikasi mengelompokkan keempat kata kerja itu ke dalam tiga tab, karena *ukur* dan *jelaskan* terjadi pada layar yang sama dan pada saat yang sama: hasil pindaian *adalah* penjelasannya. Ketiga tab itu memiliki nama dalam masing-masing bahasa aplikasi, dan nama-nama itu dipilih agar keduanya membentuk akronimnya sendiri (**Tabel 3.1**).

Tabel 3.1 Empat kata kerja siklus dan tiga tab aplikasi: SOM dalam bahasa Inggris, POP dalam bahasa Indonesia.

Kata kerja siklus	Tab bahasa Inggris	Tab bahasa Indonesia	Yang diperoleh pengguna
Ukur	**S**can	**P**indai	estimasi kesiapan 0–100 dari sinyal yang teramati
Jelaskan	**S**can	**P**indai	setiap sinyal dalam satu kalimat; daftar perbaikan berprioritas
Perbaiki	**O**ptimize	**O**ptimasi	draf jawab-dulu, FAQ, JSON-LD, bantuan `robots.txt` dan sitemap
Pantau	**M**onitor	**P**antau	pindai ulang terjadwal, tren mingguan, peringatan, riwayat

POP dibaca Pindai · Optimasi · Pantau; *SOM* dibaca Scan · Optimize · Monitor. Versi produk yang lebih awal menyebut metode yang sama "metodologi 3P"; siklusnya tidak berubah, hanya namanya. Situs pemasaran menambahkan semboyan tiga kata yang menjadi milik seluruh platform, bukan hanya milik siklus ini, yaitu *Bersama · Berkembang · Berbagi*, dalam bahasa Inggris *Gather · Grow · Give* [2].

3.2.2 Mengapa siklus, bukan garis lurus

Akan lebih sederhana menjual GEO sebagai layanan sekali jadi: pindai, perbaiki, selesai. Siklus ada karena kesiapan itu meluruh, dan cepat atau lambat setiap pemilik situs mengalami tiga penyebabnya.

Pertama, **situs berubah**. Tema baru, pembaruan *plugin*, atau perpindahan ke penyedia hosting lain dapat menghapus data terstruktur atau menulis ulang `robots.txt` tanpa ada yang menyadarinya. Situs pemasaran Profitina sendiri pernah mengalaminya: fitur "*managed* `robots.txt`" pada sebuah jaringan pengiriman konten (CDN) diam-diam melarang `GPTBot` dan `ClaudeBot` di profitina.com, padahal berkas di kode sumber situs itu sendiri sudah benar. Pelajaran yang dicatat timnya singkat saja: selalu periksa URL yang *live*, dengan melewati *cache* [2]. Bab 10 membangun alat audit untuk masalah persis itu; dalam bab ini, kejadian tersebut adalah alasan pemantauan diperlukan.

Kedua, **mesin jawaban berubah**. *Crawler* diganti namanya dan dipecah; perusahaan AI besar kini menerbitkan nama terpisah untuk *crawler* pelatihan, *crawler* pencarian, dan pengambil halaman yang bertindak atas permintaan pengguna, dan daftar itu terus direvisi [4], [5], [6]. `robots.txt` yang ditulis dua tahun lalu mungkin mengizinkan yang satu tetapi tidak yang lain.

Ketiga, **pemilik usaha belajar**. Pindaian pertama sebuah usaha kecil biasanya menemukan celah-celah yang jelas. Setelah celah itu tertutup, pindaian berikutnya menemukan celah yang lebih halus, misalnya paragraf pembuka yang tidak pernah menyebut nama usaha, atau harga yang hanya tertulis di dalam gambar. Perbaikan datang secara bertahap.

Siklus juga merupakan bentuk yang jujur bagi produk yang tidak dapat menjanjikan hasil. Tidak seorang pun di luar sebuah mesin jawaban yang mengendalikan apa yang akan dikatakannya, sehingga yang dapat ditawarkan platform adalah cara yang disiplin dan berulang untuk memperbesar peluang, sekaligus memeriksa bahwa peluang yang sudah membesar itu tetap besar.

3.2.3 Apa yang diukur, dan apa yang tidak

Keputusan rancangan terpenting dalam produk ini justru sebuah penolakan. Profitina **tidak** mengukur visibilitas sebuah usaha dengan bertanya kepada ChatGPT, Gemini, atau Perplexity. Mesin-mesin jawaban itu adalah *sasaran* yang kesiapannya untuk mengutip sebuah usaha diperkirakan; mesin-mesin itu tidak pernah menjadi komponen yang menjadi sandaran Profitina [2].

Ada tiga alasan, dan masing-masing sudah cukup dengan sendirinya. *Kemandirian*: pengukuran yang memanggil layanan pihak ketiga akan rusak ketika layanan itu mengubah ketentuan, harga, atau keluarannya, dan jawaban sebuah mesin atas pertanyaan yang sama berubah-ubah dari hari ke hari. *Privasi*: bertanya kepada penyedia luar dengan menyebut nama usaha, pesaingnya, dan pertanyaan pelanggannya berarti mengirimkan persis data yang lebih suka dijaga oleh usaha kecil. *Biaya dan prinsip*: aturan internal platform melarang ketergantungan pada API pihak ketiga yang berbayar. Penelitian di Bab 2 menjadikan pilihan ini masuk akal, bukan sekadar praktis. Sifat-sifat yang membuat sebuah sumber terlihat di dalam jawaban, yaitu substansi yang dapat diekstraksi, struktur yang jelas, fakta yang tepat, dan akses terbuka bagi *crawler*, adalah sifat-sifat *sumber* itu sendiri, dan dapat diamati langsung pada sumbernya [1].

Penolakan yang sama membentuk AI lokalnya. Di mana pun Profitina memakai model bahasa, untuk menyusun draf konten, menggolongkan teks pendek, atau memilah umpan balik, model itu berjalan di peladennya sendiri, yaitu model keluarga Qwen yang dilayani Ollama, sehingga data pengguna tidak pernah meninggalkan mesin untuk pemrosesan AI [2]. Bab 6 menjelaskan dua jalur AI lokal tersebut.

3.3 Ukur: Apa yang Diperiksa Sebuah Pindaian

3.3.1 Masukan

Sebuah pindaian berangkat dari masukan yang sangat sedikit. Pemilik usaha memasukkan nama dan domainnya; seorang kreator memasukkan nama dan akun media sosialnya. Dari situ, AI Visibility Engine **mengisi otomatis** (*auto-fill*) sisanya: kategori usaha yang paling mungkin, lokasinya, dan beberapa pertanyaan realistis yang mungkin diajukan pelanggan kepada mesin jawaban, seperti “laundry kiloan Surabaya” atau “dry cleaning dekat Rungkut” [2]. Pemilik

memeriksa nilai-nilai yang terisi dan boleh mengubahnya. Prinsip rancangannya: orang yang paling awam secara teknis pun harus dapat memulai, dan mesinlah yang mengetik.

Pengisian otomatis memunculkan pertanyaan yang dihadapi setiap pengklasifikasi: apa yang harus dilakukan ketika ragu? Banyak sistem menebak diam-diam lalu menyajikan tebakan itu seolah-olah fakta. Deteksi kategori Profitina berbuat sebaliknya. Ketika tidak dapat memutuskan, ia **berkata "belum yakin"** dan meminta pengguna memilih dari daftar pendek [2]. Perilaku kecil itu adalah salah satu prinsip seluruh platform dalam ukuran mini: "saya belum tahu" yang jujur lebih berharga daripada kekeliruan yang penuh percaya diri, karena kategori yang salah akan diam-diam membelokkan setiap langkah pindaian berikutnya.

3.3.2 Tiga keluarga sinyal

Apa yang diamati sebuah pindaian? Pada tingkat yang digunakan produk untuk menggambarkan dirinya di depan publik, sinyal-sinyalnya terbagi menjadi tiga keluarga [2]. **Tabel 3.2** mencantumkannya bersama standar terbuka yang memungkinkan masing-masing diamati. Tabel ini menyebut keluarga dan standar, bukan daftar sinyal internal produk, bobotnya, ataupun cara penggabungannya; semua itu milik produk dan bukan pokok bahasan buku ini. Yang penting untuk pemahaman adalah bahwa setiap keluarga terdiri atas hal-hal yang dapat diperiksa oleh siapa saja.

Tabel 3.2 Tiga keluarga sinyal teramati di balik estimasi kesiapan, dan standar publik yang memungkinkan masing-masing diamati.

Keluarga	Pertanyaan yang diajukan	Tempat buktinya (standar terbuka)
Akses	Bolehkah *crawler* mesin jawaban membaca situs itu? Dapatkah mereka menemukan halaman-halamannya?	`robots.txt`, RFC 9309 [7]; sitemap [8]; dokumentasi *crawler* tiap perusahaan [4], [5], [6], [9], [10]
Struktur	Apakah situs menggambarkan dirinya dalam bentuk yang dapat dibaca mesin?	tipe dan properti schema.org [11] yang ditulis sebagai JSON-LD [12]; `llms.txt` [13]
Keterekstrakan	Dapatkah program mengambil jawaban yang utuh dan benar dari teksnya?	judul-judul HTML dan struktur dokumen, *title*, deskripsi meta, bahasa [14]; teks yang terlihat itu sendiri

Akses didahulukan karena tanpa akses tidak ada hal lain yang berarti. Mesin tidak dapat mengutip halaman yang tidak boleh diambil *crawler*-nya, dan tidak dapat mengambil halaman yang tidak pernah ditemukannya. Bab 2 menjelaskan cara kerja grup dan aturan `robots.txt` [7]; sebuah pindaian mengajukan pertanyaan praktisnya untuk setiap *crawler* yang penting.

Struktur membuat fakta tidak bermakna ganda. Harga yang tertulis pada spanduk dekoratif hanyalah tebakan bagi program; `priceRange` di dalam JSON-LD adalah fakta. FAQ yang ditandai sebagai `FAQPage` menyerahkan kepada mesin jawaban persis jenis pernyataan singkat dan utuh yang menurut penelitian paling terlihat [1], [11], [12].

Keterekstrakan adalah mutu teks itu sendiri. Apakah ada satu judul utama yang jelas, dengan bagian-bagian bersarang di bawahnya? Apakah halaman menyatakan *siapa* dan *apa* pada baris-baris pertamanya, atau malah dibuka dengan "Selamat datang!" dan gambar yang bergilir? Apakah ada cukup teks sungguhan, atau halaman itu sebagian besar berisi skrip, gaya, dan gambar? Semua ini adalah keutamaan lama tulisan web yang baik; mesin jawaban menjadikannya akibat yang dapat diukur.

3.3.3 Kemajuan yang dapat dibaca

Pindaian memakan waktu: ia mengambil halaman, mengurainya, menjalankan pemeriksaan, dan untuk beberapa langkah berkonsultasi dengan model lokal. Ikon berputar selama empat puluh detik tidak memberi tahu pengguna apa pun dan mengundangnya menutup tab. Profitina menampilkan **bilah kemajuan terperinci dari 0 sampai 100%** dengan label langkah, misalnya "kueri 3/8", dan perkiraan sisa waktu [2]. Hal kecil, tetapi itulah bedanya pengguna yang memercayai alat dengan pengguna yang curiga alat itu macet.

3.3.4 Angka itu, dan maknanya

Hasilnya adalah estimasi kesiapan pada skala 0 sampai 100, dengan metodologinya ditampilkan terbuka dan daftar perbaikan berprioritas di sampingnya. Tiga sifat angka itu perlu dinyatakan dengan tepat, karena justru sifat-sifat inilah yang sering diabaikan pembaca yang terburu-buru.

Angka itu adalah **estimasi kesiapan**, bukan ukuran jumlah kutipan. Sebuah situs dapat memperoleh skor tinggi dan tetap belum dikutip hari ini, karena pilihan sebuah mesin juga bergantung pada pertanyaannya, pada sumber-sumber pesaing, dan pada perubahan mesin itu sendiri.

Angka itu **relatif terhadap sinyal yang teramati**. Apa pun yang tidak dapat diamati dari luar, misalnya reputasi usaha di mata pelanggannya, berada di luar jangkauannya.

Angka itu terutama **untuk perbandingan dari waktu ke waktu**. Situs yang sama, dipindai sebelum dan sesudah perbaikan dengan metode yang sama, memberi pemilik selisih yang bermakna. Itulah sebabnya produk ini berbentuk siklus.

> **WAWASAN KUNCI:** Skor kesiapan menjawab pertanyaan "seberapa baik situs ini membuat dirinya dapat dibaca, dipahami, dan dikutip oleh mesin?" Skor itu tidak menjawab "apakah ChatGPT akan menyebut saya besok?" Tidak ada alat jujur yang dapat menjawab pertanyaan kedua, dan Profitina menyatakannya pada setiap hasil [2], [3].

3.4 Jelaskan: Dari Sebuah Angka ke Daftar Perbaikan

Angka adalah vonis; daftar adalah rencana. Karena itu, halaman hasil pindaian memakai sebagian besar ruangnya untuk daftar. Setiap celah yang teramati menjadi satu baris dalam bahasa sehari-hari, lengkap dengan bukti yang memunculkannya dan langkah berikutnya. Pemilik binatu tidak perlu tahu apa itu RFC 9309; ia perlu membaca "GPTBot dan ClaudeBot diblokir oleh `robots.txt` Anda; hapus dua baris `Disallow` itu bila Anda ingin dibaca oleh keduanya", dan perlu tahu bahwa hal itu lebih penting daripada atribut `lang` yang hilang.

Dua gagasan mengatur daftar itu, dan keduanya dapat dinyatakan tanpa menguraikan cara produk menyusun peringkat.

Setiap baris dijelaskan. Baris yang hanya berbunyi "data terstruktur: 40%" mengecewakan pemilik. Baris yang berbunyi "blok JSON-LD Anda bukan JSON yang sah sehingga diabaikan program; kesalahannya adalah koma sesudah properti terakhir" memberinya sesuatu untuk dikerjakan. Dalam lab di Subbab 3.10, setiap pemeriksaan mengembalikan tepat tiga hal karena alasan ini: poinnya, *bukti*-nya, dan *saran*-nya.

Urutkan menurut akibatnya. Sebagian celah menghalangi segala sesuatu di belakangnya: *crawler* yang dilarang tidak dapat membaca FAQ yang sempurna sekalipun. Celah lain hanyalah penghalusan. Daftar yang berguna menaruh penghalang di urutan pertama. Cara sederhana untuk

memikirkannya, yang merupakan cara kita sendiri dan bukan cara produk, adalah mengajukan dua pertanyaan untuk setiap perbaikan, “apa yang dibukanya?” dan “berapa besar kerjanya?”, lalu mendahulukan perbaikan yang membuka banyak dengan kerja sedikit. Menghapus dua baris dari `robots.txt` bernilai tinggi pada keduanya.

Halaman hasil juga memuat tautan kecil yang mudah terlewat tetapi penting bagi platform: **kirim umpan balik tentang pindaian ini**. Pengguna yang merasa hasilnya keliru dapat menyampaikannya. Bab 7 menelusuri umpan balik itu melalui alur lima langkah, dan di dalamnya manusia, bukan algoritma, yang memutuskan apa yang berubah pada rilis berikutnya [2].

3.5 Perbaiki: Mesin Konten dan Pendamping Teknis

3.5.1 Apa yang disusun Mesin Konten

Tab *Optimasi* mengubah celah menjadi bahan yang siap terbit. **Mesin Konten** (*Content Engine*) di dalamnya menyusun draf GEO: teks *jawab-dulu* (*answer-first*) yang dalam dua sampai empat kalimat pertamanya menyatakan siapa usaha itu, apa yang dikerjakannya, dan di mana, didukung fakta yang spesifik dan dapat diperiksa; sekumpulan entitas dan poin kunci yang ditulis agar mudah diekstraksi; FAQ yang dirumuskan sebagai pertanyaan yang benar-benar diketik pelanggan; serta JSON-LD schema.org yang sesuai [2]. Pemilik meninjau, membetulkan, dan menerbitkannya di situsnya sendiri. Profitina tidak pernah menerbitkan atas nama pengguna.

Apa arti *jawab-dulu* dalam praktik? Bandingkan dua pembuka untuk binatu yang sama.

Welcome! Clean clothes, happy life. We are the best and the fastest. Your satisfaction is our priority.

Profitina Laundry is a laundry in Rungkut, Surabaya, that washes, irons and dry-cleans clothes for households and small businesses. Wash-and-fold costs Rp7,000 per kilogram with a minimum of 3 kg, a normal order is ready in two days, and every outlet accepts cash and QRIS.

(Kedua kutipan dibiarkan dalam bahasa Inggris karena keduanya adalah teks halaman contoh lab di Subbab 3.10.) Pembuka pertama (“Selamat datang! Pakaian bersih, hidup bahagia. Kami yang terbaik dan tercepat.”) tidak mengatakan apa pun yang dapat dikutip program untuk menjawab pertanyaan apa pun. Pembuka kedua menjawab sedikitnya lima: apa itu Profitina Laundry, di mana letaknya, apa yang dikerjakannya, berapa biayanya, dan berapa lama pengerjaannya. Setiap faktanya dapat diperiksa, dan tidak satu pun berupa klaim “ter-”. Lab di Subbab 3.10 memuat satu pemeriksaan, yang kita namai *answer-first*, yang mengajukan versi sempit pertanyaan ini: apakah nama usaha muncul di antara enam puluh kata pertama konten utama?

3.5.2 Draf AI hanya bila lebih baik

Mesin Konten berjalan pada model lokal platform. Model dapat menulis teks yang lancar tetapi lebih buruk daripada templat sederhana yang benar: lebih kabur, lebih panjang, atau keliru dengan penuh percaya diri tentang suatu fakta. Karena itu Profitina menempatkan **gerbang mutu** (*quality gate*) di depan setiap draf AI. Draf dibandingkan dengan keluaran templat bawaan untuk masukan yang sama, dan hanya disajikan bila memang lebih baik; pada tolok ukur setelah revisi September 2026, draf AI menang dalam 5 dari 6 kasus [2]. Cara gerbang itu menilai draf adalah milik produk dan tidak diuraikan di sini. Prinsipnyalah yang penting, dan prinsip itu dapat disalin ke sistem mana pun: *pakailah model hanya ketika ia mengalahkan cara yang sederhana.*

Aturan yang berkaitan mengatur kegagalan. Ketika model tidak tersedia, karena prosesnya sedang dimulai ulang atau mesinnya sibuk, aplikasi **mengatakannya**, alih-alih diam-diam menyajikan templat seolah-olah draf AI [2]. Pengguna yang diberi tahu "model tidak tersedia, ini templatnya, dan ditandai sebagai templat" dapat memutuskan langkahnya sendiri. Pengguna yang dikelabui tidak bisa.

3.5.3 Pendamping Teknis

Separuh kedua tab *Optimasi* menangani lapisan yang dapat dibaca mesin. **Pendamping Teknis** (*Technical Assistant*) menghasilkan blok JSON-LD untuk usaha itu, membantu pemilik menulis `robots.txt` yang mengizinkan *crawler* pilihannya (daftarnya biasanya mencakup `GPTBot`, `OAI-SearchBot`, `ClaudeBot`, `PerplexityBot`, `Google-Extended`, dan *crawler* mesin pencari klasik), serta memberi panduan untuk mengirimkan sitemap ke Bing Webmaster Tools [2]. Bing mendapat baris tersendiri karena alasan praktis: dalam panduan Profitina, indeks Bing adalah tempat Copilot milik Microsoft dan fitur pencarian ChatGPT mengambil banyak sumbernya, sehingga mendaftar di sana, dan mengizinkan `Bingbot` masuk, adalah salah satu langkah termurah yang bernilai tinggi bagi usaha kecil [2], [10].

Lab G6 di Bab 2 membangkitkan tiga artefak yang sama dari sebuah berkas fakta. Pendamping Teknis mengerjakan tugas yang sama secara interaktif, untuk usaha yang belum pernah mendengar satu pun dari ketiganya.

3.6 Pantau: Riwayat, Jadwal, dan Peringatan

3.6.1 Tren mingguan

Tab *Pantau* menyimpan setiap pindaian sebuah merek dan menggambar **tren skor mingguan** sebagai grafik; riwayatnya dapat diekspor sebagai CSV [2]. Riwayat pindaian **hanya milik pemiliknya**: tidak ada orang lain, termasuk pengguna lain, yang melihat situs mana yang dipantau sebuah usaha atau berapa skornya. **Penjadwal** memindai ulang secara otomatis dengan irama yang bergantung pada paket: mingguan pada paket Free, harian pada Pro. Bila pindaian ulang menemukan **perubahan yang berarti**, naik ataupun turun, platform mengeluarkan peringatan, dan bila dikehendaki juga melalui email [2].

3.6.2 Apa arti "berarti"

Misalkan s_t adalah skor sebuah merek pada minggu ke-t, untuk $t = 1,2,3,\ldots$ Perubahan dari satu minggu ke minggu berikutnya adalah

$$\Delta_t = s_t - s_{t-1} \tag{3.1}$$

dan aturan peringatan sederhana, yang kita pilih untuk keperluan pengajaran, berbunyi ketika $\Delta_t \geq \delta$ atau $\Delta_t \leq -\delta$ untuk ambang δ yang dipilih pemilik. Nilai mutlak membuat aturan ini simetris: penurunan sama pentingnya dengan kenaikan, bahkan biasanya lebih penting, karena penurunan sering merupakan tanda pertama bahwa sebuah pembaruan merusak sesuatu. Dengan $\delta = 10$ dan deret $41,44,58,63,61,$ Persamaan (3.1) menghasilkan perubahan $+3, +14, +5, -2$, dan hanya minggu dengan $\Delta_t = +14$ yang memicu peringatan. Seandainya situs itu rusak oleh pembaruan tema pada minggu kelima dan skornya menjadi 45, perubahan -18 juga akan memicu peringatan.

Aturan semacam ini memiliki kelemahan yang harus dihadapi setiap sistem pemantauan: derau. Bila skor naik-turun beberapa poin setiap minggu karena sebab yang tidak dikendalikan siapa pun, δ yang kecil menghasilkan peringatan yang lama-kelamaan diabaikan orang. Ambang yang terlalu besar melewatkan kerusakan yang nyata. Ambang milik produk adalah urusan produk; pelajaran bagi sistem apa pun adalah memilih δ lebih besar daripada variasi mingguan yang biasa, tetapi cukup kecil untuk menangkap `robots.txt` yang rusak, yang biasanya menggeser skor dalam jumlah besar sekaligus. Soal 3.9 membahas sebuah contoh.

3.7 Melampaui Situs Web: Laporan Peneliti dan Organisasi

Siklus ini dibangun untuk usaha, tetapi gagasannya, yaitu ukur yang dapat diamati, jelaskan celahnya, bantu menutupnya, berlaku pula bagi dunia ilmiah. Publikasi seorang dosen hanya akan dikutip mesin jawaban bila publikasi itu dapat *ditemukan*: di bawah satu nama yang konsisten, terhubung ke pengenal persisten, dengan salinan terbuka yang dapat dibaca *crawler*. Sejak akhir Agustus 2026, Profitina membawa siklus ini ke dunia tersebut melalui dua laporan [2].

3.7.1 Laporan Visibilitas Peneliti

Laporan Visibilitas Peneliti (*Researcher Visibility Report*) mengukur seberapa lengkap dan mudah ditemukan sebuah rekaman ilmiah. Sumber datanya adalah **OpenAlex**, katalog terbuka karya ilmiah, peneliti, wadah publikasi, dan institusi yang menyerap metadata Crossref dan bebas dikueri [15]. Untuk rekaman seorang peneliti, laporan ini memeriksa hal-hal yang menentukan apakah karya-karyanya dapat ditemukan dan diatribusikan: apakah karyanya memiliki **DOI**, apakah penelitinya terhubung ke **ORCID iD**, pengenal persisten untuk peneliti [16], bagaimana **afiliasi** dieja di seluruh karyanya, apakah tersedia salinan **akses terbuka**, dan bagaimana karya-karya itu dikutip. Setiap celah disebut secara konkret, lengkap dengan cara menutupnya: “Belum ada ORCID yang tertaut”, atau “Afiliasi Anda ditulis dalam *N* ejaan berbeda” [2].

Laporan ini memiliki satu fitur yang layak diberi nama sendiri: **kontrol negatif permanen** (*negative control*). Nama fiktif selalu menghasilkan “tidak ditemukan”, tidak pernah skor yang tampak masuk akal. Dalam uji produk sendiri, 8 dari 8 subjek fiktif tetap bernilai 0,0 [2]. Mengapa hal ini begitu penting? Karena sistem yang mengembalikan angka untuk *masukan apa pun* tidak dapat dipercaya untuk masukan yang penting. Seandainya nama karangan memperoleh skor 34, setiap skor nyata menjadi patut dicurigai: berapa bagian yang berasal dari rekaman sungguhan, dan berapa bagian dari kebiasaan sistem menghasilkan angka? Kontrol negatif adalah bukti termurah bahwa angka-angka itu berasal dari data.

Laporan ini memproses *karya dan metadatanya*, bukan orangnya: tidak ada alamat, foto, tautan media sosial, ataupun apa pun dari profil Komunitas. Orang yang tidak ditemukan sama sekali tidak diberi skor [2].

3.7.2 Visibilitas Organisasi

Visibilitas Organisasi (*Organization Visibility*) menerapkan data yang sama pada tingkat institusi. Laporan ini sengaja *bukan* papan peringkat orang; ia hanya melaporkan celah agregat yang dapat ditutup oleh institusi. Uji produk pada Institut Teknologi Sepuluh Nopember (ITS) sangat mencerahkan. Dalam 500 karya sampel, afiliasi institusi itu muncul dalam **466 ejaan berbeda**, dan **244 dari 607 peneliti** belum menautkan ORCID [2]. Tidak satu pun dari angka ini berkata apa-apa tentang mutu penelitian siapa pun. Keduanya berkata sesuatu yang tepat tentang mengapa penelitian itu lebih sulit ditemukan dan diatribusikan daripada semestinya, dan

keduanya menyarankan perbaikan institusional yang konkret: satu ejaan baku afiliasi pada setiap artikel yang dikirim, dan gerakan pendaftaran ORCID. Laporan ini mencakup organisasi yang karya-karyanya terindeks di OpenAlex.

3.7.3 Academic Citation Scan

Layanan ketiga, **Academic Citation Scan**, menilai kesiapan kutip sebuah situs dari sinyal yang tersedia di situs itu, dan dapat dibayar dengan PROFIT, koin utilitas platform yang dibahas di Bab 12. Produk ini secara tegas membedakannya dari Scopus dan Google Scholar: layanan ini bukan indeks sitasi dan tidak mengklaim menghitung sitasi [2].

Gagasan Visibilitas Organisasi, yaitu menghitung ejaan berbeda dari satu nama, cukup sederhana untuk diprogram dalam beberapa baris: normalkan setiap ejaan (huruf besar-kecil, spasi, tanda baca), lalu hitung anggota himpunan hasilnya. Lab G9 di Bab 13 mengerjakan persis hal itu terhadap API publik OpenAlex, lengkap dengan kontrol negatifnya sendiri.

3.8 Satu Teks, Banyak Platform

Visibilitas di mesin jawaban hanyalah satu kanal; usaha kecil juga hidup di media sosial, tempat setiap platform memiliki nada, kebiasaan, dan batas panjangnya sendiri. **Satu Teks, Banyak Platform** (*One Text, Many Platforms*) menangani pekerjaan rutin itu. Pengguna menempelkan satu teks sumber dan memilih satu dari enam platform, X, Instagram, LinkedIn, YouTube, TikTok, atau Facebook, lalu menerima versi siap tempel yang sesuai dengan nada platform itu dan **dijamin muat dalam batas karakternya** [2].

Tiga rincian rancangannya patut diperhatikan.

Satu permintaan pada satu waktu, dengan antrean yang jujur. Penyesuaian berjalan pada model lokal, dan mesin menjalankan satu permintaan semacam itu pada satu waktu untuk seluruh sistem. Alih-alih menyembunyikannya, antarmuka menampilkan **posisi pengguna dalam antrean beserta perkiraan waktu tunggunya** [2]. Antrean yang terlihat dapat ditoleransi; antrean yang tersembunyi tampak seperti produk yang rusak.

Jaminan yang dapat ditepati. Fitur ini menjamin batas karakter, yang dapat diperiksa program secara pasti, dan secara tegas *tidak* menjanjikan hasil yang "lebih menarik" atau "viral", yang tidak dapat diperiksa program mana pun [2]. Janjikanlah hanya apa yang dapat Anda verifikasi.

Berbagi tanpa API. Untuk X, tombol *Bagikan ke X* membuka *web intent* X di akun pengguna sendiri dengan teks yang sudah terisi; pengguna sendirilah yang menekan "posting". Peladen Profitina hanya menyusun teksnya dan tidak pernah menghubungi X; tidak ada API X di dalam produk, dan berbagi tidak menghasilkan PROFIT, karena sebuah unggahan tidak dapat diverifikasi [2]. Fitur ini juga dapat dipakai tanpa akun, sehingga menjadi salah satu hal pertama yang dapat dicoba pengunjung.

3.9 Perjalanan Pengguna: Profitina Laundry, Langkah demi Langkah

Fitur-fitur pada Subbab 3.2 sampai 3.8 lebih mudah dipahami ketika dipakai. Subbab ini mengikuti seorang pengguna melalui sembilan langkah. Usahanya adalah **Profitina Laundry**, binatu fiktif yang menjadi studi kasus buku ini: sebuah gerai di Rungkut, Surabaya, dikelola pemiliknya yang akan kita sebut Sari, dengan situs web kecil di `laundry.example`. **Gambar 3.2**

adalah peta perjalanannya; *wireframe* yang menyusul digambar untuk buku ini guna memperlihatkan bentuk setiap layar, dan menyederhanakan antarmuka yang sebenarnya.

Gambar 3.2 Sembilan langkah perjalanan pengguna. Langkah 1 dan 2 tidak memerlukan akun; langkah 4 sampai 6 adalah siklus SOM/POP, dan panah putus-putus menunjukkan bahwa siklus itu berulang.

3.9.1 Langkah 1: Datang

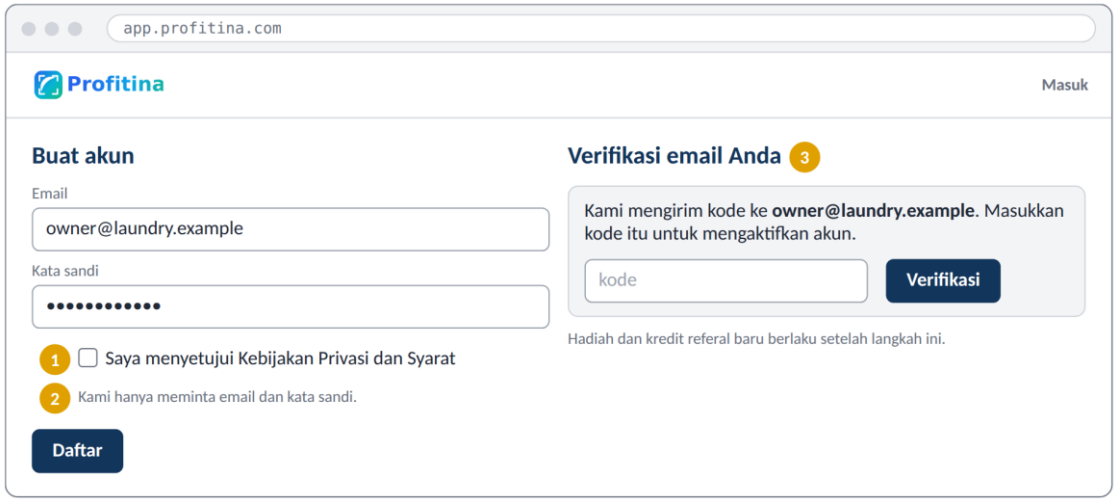
Sari mendengar tentang Profitina dari seorang teman yang berjualan siomay secara daring, lalu mengunjungi profitina.com. Situs pemasaran itu ditulis sesuai nasihat yang diberikannya kepada penggunanya: halaman jawab-dulu, FAQ yang ditandai sebagai data terstruktur, dan `llms.txt` di akar situsnya [2]. Di dalam halaman-halamannya tertanam **widget komunitas** platform, yaitu umpan *real-time* dari Bab 4, yang dapat dibaca pengunjung anonim tetapi tidak dapat diikutinya. Sari membaca beberapa unggahan dari usaha kecil lain dan satu utas tentang `robots.txt`, lalu memutuskan untuk mencoba.

3.9.2 Langkah 2: Mencoba tanpa akun

Dua fitur dapat dipakai tanpa mendaftar. Pengunjung mendapat **kuota pindai gratis** yang kecil, cukup untuk melihat seperti apa sebuah pindaian; dan **Satu Teks, Banyak Platform** terbuka bagi siapa saja [2]. Sari menempelkan pengumuman yang ditulisnya untuk para pelanggan tetap, “Mulai 1 Oktober kami buka pukul 06.30 pada hari kerja”, memilih Instagram, dan menerima keterangan foto yang terdengar sungguh seperti Instagram. Manfaatnya kecil tetapi langsung, dan itu disengaja: sebuah alat semestinya membuktikan nilainya sebelum meminta alamat email.

3.9.3 Langkah 3: Mendaftar

Pendaftaran meminta yang paling minimal: **alamat email dan kata sandi**, tidak lebih. Kotak persetujuan untuk Kebijakan Privasi dan Syarat **tidak pernah tercentang sejak awal**, dan pemberitahuan dwibahasa yang singkat menerangkan apa yang dikumpulkan dan untuk apa (**Gambar 3.3**). Kata sandi hanya disimpan sebagai *hash* satu arah Argon2id [2], [17].

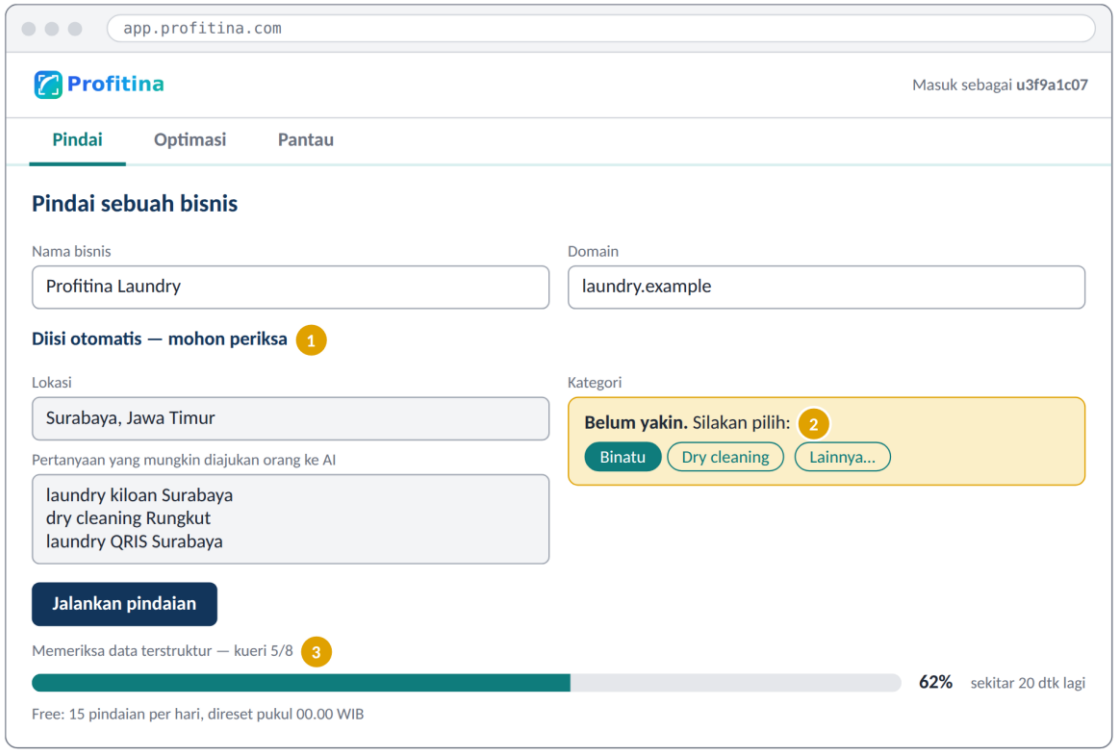


Gambar 3.3 Pendaftaran, sebagai *wireframe*. (1) Kotak persetujuan mula-mula kosong; (2) yang diminta hanya email dan kata sandi; (3) kode yang dikirim melalui email mengaktifkan akun.

Setelah mendaftar, Sari menerima **kode sekali pakai melalui email** dan mengetikkannya. Verifikasi email adalah lapisan pertahanan pertama platform terhadap bot: pendaftaran otomatis yang tidak dapat membaca kotak surat tidak akan melangkah lebih jauh. Verifikasi ini juga melindungi sistem hadiah pada bab-bab berikutnya, karena hadiah dan kredit referal baru berlaku setelah email terverifikasi [2].

3.9.4 Langkah 4: Pindai

Sari membuka tab *Pindai* pada antarmuka berbahasa Indonesianya, lalu mengetikkan nama usaha dan domainnya. **Gambar 3.4** memperlihatkan apa yang terjadi kemudian.



Gambar 3.4 Tab Pindai, sebagai *wireframe*. (1) Lokasi dan pertanyaan pelanggan terisi otomatis untuk diperiksa pemilik; (2) bila kategori belum pasti, sistem berkata “belum yakin” dan bertanya; (3) kemajuan ditampilkan bersama langkahnya dan sisa waktunya.

Lokasi dan tiga pertanyaan pelanggan muncul sudah terisi (penanda 1). Kategorinya tidak: situsnya menyebut mencuci, menyetrika, *dan dry cleaning*, sehingga sistem, yang ragu apakah ini binatu atau jasa *dry cleaning*, berkata **"belum yakin"** dan menawarkan pilihan (penanda 2). Sari mengetuk *Binatu*. Pindaian berjalan dengan bilah kemajuan yang menyebut langkah yang sedang dikerjakan (penanda 3). Paket Free mengizinkan 15 pindaian sehari, dan penghitungnya direset pukul 00.00 WIB, tengah malam menurut zona waktu Jakarta [2].

Hasilnya (**Gambar 3.5**) membuat Sari tersadar, dan berguna.

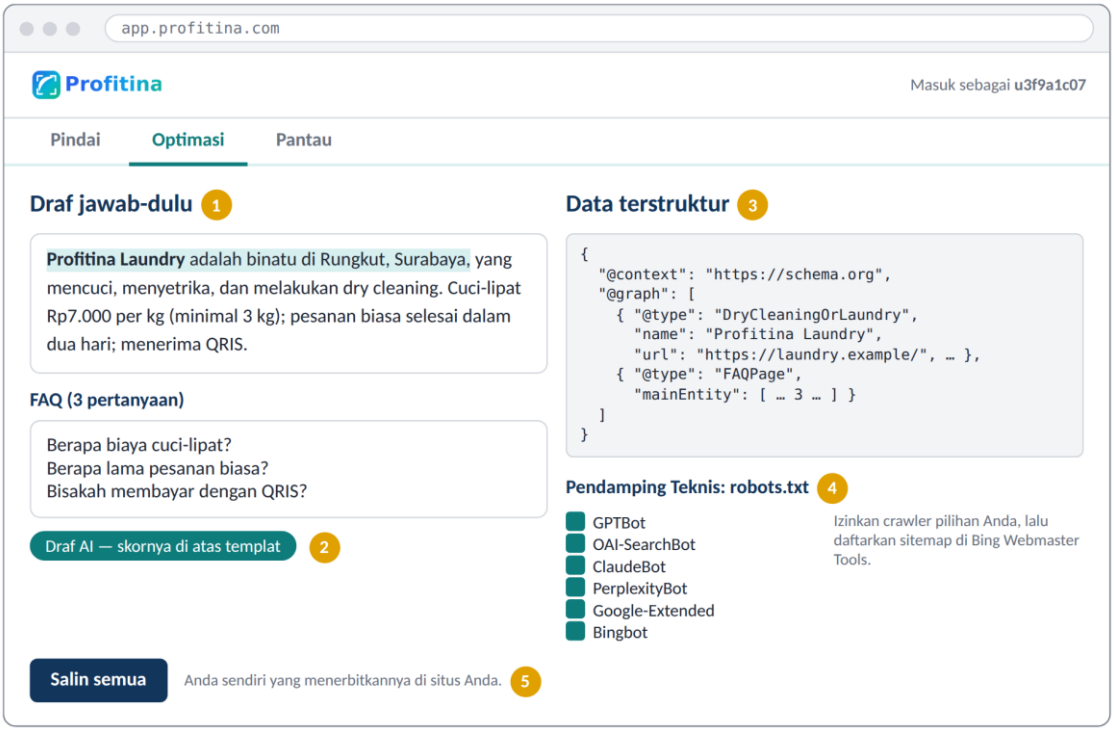
Gambar 3.5 Hasil pindaian, sebagai *wireframe* dengan nilai ilustratif. (1) Estimasi kesiapan; (2) tautan ke metodenya; (3) perbaikan berprioritas; (4) jalan masuk ke Mesin Konten; (5) umpan balik tentang pindaian.

Estimasinya rendah (penanda 1), dan sebuah tautan menjelaskan cara estimasi itu dibuat (penanda 2). Daftar perbaikan (penanda 3) itulah yang sungguh akan ia pakai. Di bagian atas, berprioritas tinggi: dua *crawler* AI diblokir oleh `robots.txt`-nya, dan data terstrukturnya mengandung kesalahan. Berprioritas sedang: halamannya tidak menyebut siapa dirinya pada baris-baris pertama, dan ia belum punya sitemap. Di bagian bawah: judul-judulnya melompat dari `h1` ke `h4`. Setiap baris adalah kalimat yang dapat ia pahami. Dua tautan mengantar ke langkah berikutnya: ke Mesin Konten (penanda 4), dan ke umpan balik tentang pindaian itu sendiri (penanda 5).

Lab di Subbab 3.10 menghadirkan kembali pengalaman ini pada salinan situs lama Sari, dan pemeriksanya menemukan celah-celah yang sama, dengan aturan kita sendiri.

3.9.5 Langkah 5: Optimasi

Tab *Optimasi* mengubah daftar itu menjadi bahan (**Gambar 3.6**).

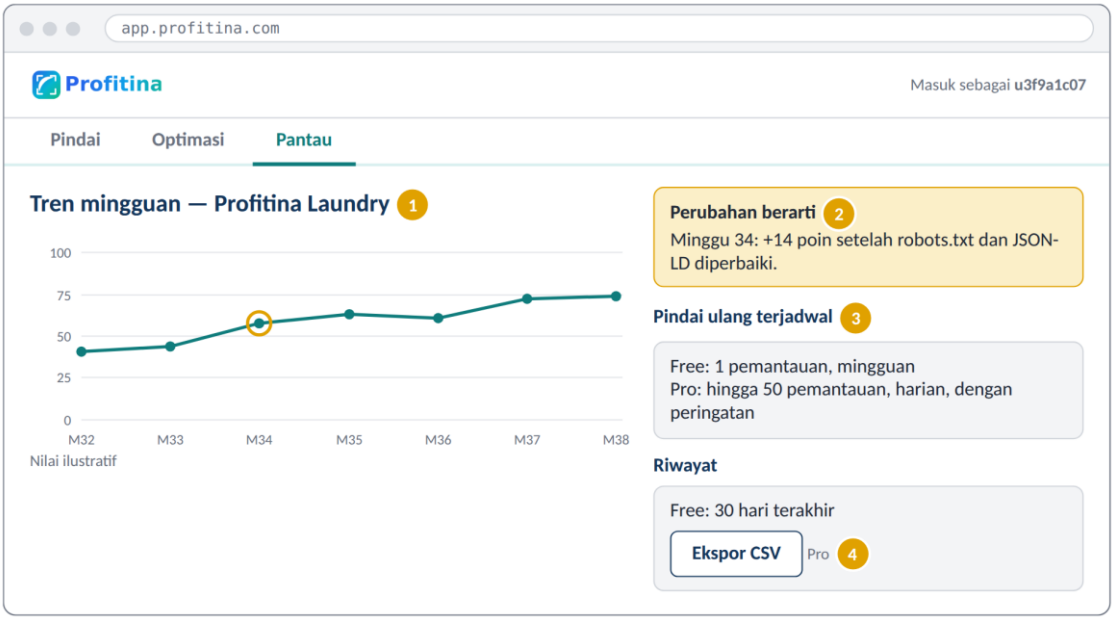


Gambar 3.6 Tab Optimasi, sebagai *wireframe*. (1) Draf jawab-dulu yang langsung menyebut entitasnya; (2) putusan gerbang mutu; (3) JSON-LD yang sesuai; (4) pilihan *crawler* dari Pendamping Teknis; (5) pemilik sendiri yang menerbitkan di situsnya.

Drafnya (penanda 1) dibuka dengan kalimat yang dipuji Subbab 3.5.1: *Profitina Laundry adalah binatu di Rungkut, Surabaya, yang mencuci, menyetrika, dan melakukan dry cleaning.* Sebuah label menyatakan bahwa draf AI ini lolos gerbang mutu (penanda 2). Di sampingnya ada JSON-LD untuk usahanya dan FAQ-nya (penanda 3), serta daftar *crawler* dari Pendamping Teknis, yang dibiarkannya tercentang semua (penanda 4). Ia membetulkan satu harga, menyalin semuanya, lalu meminta pemuda yang dulu membuatkan situsnya untuk menempelkan draf itu ke beranda, mengganti JSON-LD yang rusak, dan mengganti `robots.txt`. Platform tidak menyentuh situsnya; Sari sendiri yang menerbitkan (penanda 5). Sesudah itu ia mendaftarkan sitemap-nya di Bing Webmaster Tools, mengikuti panduan Pendamping Teknis.

3.9.6 Langkah 6: Pantau

Di tab *Pantau*, Sari menyalakan satu pemantauan mingguan yang termasuk dalam paket Free-nya (**Gambar 3.7**).



Gambar 3.7 Tab Pantau, sebagai *wireframe* dengan nilai ilustratif. (1) Tren mingguan; (2) peringatan atas perubahan yang berarti; (3) irama pindai ulang tiap paket; (4) ekspor CSV, fitur Pro.

Selama minggu-minggu berikutnya garis tren menanjak (penanda 1). Minggu ketika perbaikan mulai berlaku menghasilkan peringatan (penanda 2), yaitu kenaikan +14 poin yang dihitung dengan Persamaan (3.1) pada Subbab 3.6.2. Paket Free menyimpan riwayat 30 hari dan memindai ulang setiap minggu; Pro akan memindai ulang setiap hari, mengizinkan hingga 50 pemantauan, dan mengekspor riwayat sebagai CSV (penanda 3 dan 4) [2].

3.9.7 Langkah 7: Berpartisipasi

Sari mengunggah tulisan di komunitas tentang apa yang dipelajarinya ("panel hosting saya ternyata memblokir GPTBot!") dan mengikuti sebuah kanal tentang situs web usaha kecil. Unggahannya mendapat sedikit PROFIT, koin utilitas platform, karena dinilai berguna; hadiahnya memang kecil menurut rancangan, hanya dibayarkan kepada orang yang emailnya terverifikasi, dan memiliki batas harian yang direset pukul 00.00 WIB [2]. Bab 4, 5, dan 7 menguraikan komunitas, langkah-langkah pengamanannya, dan siklus hadiahnya; Bab 12 menguraikan PROFIT itu sendiri.

3.9.8 Langkah 8: Meningkatkan paket, bila ia mau

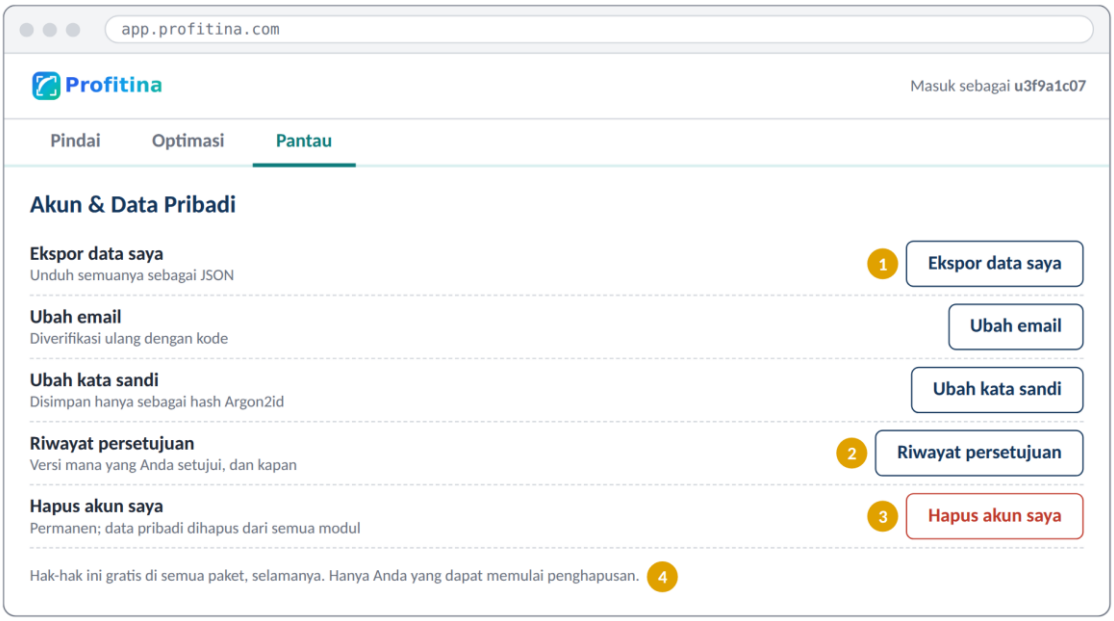
Sari boleh tetap di paket Free selama ia suka. Bila ia menginginkan pemantauan harian, lebih banyak pemantauan, riwayat lengkap, dan ekspor CSV, ia dapat beralih ke **Pro**, yang berharga **Rp50.000 untuk 30 hari**, dibayar dengan QRIS dari dompet digital atau mobile banking Indonesia mana pun, atau dengan menukarkan PROFIT; paket ini dapat diperpanjang otomatis dari saldonya, dengan pengingat sebelum masa berlakunya habis [2]. **Tabel 3.3** membandingkan kedua paket sebagaimana digambarkan kartu paket aplikasi pada September 2026. Hanya ada dua paket dan tanpa iklan, untuk selamanya; paket ketiga yang lebih awal, Agency, dihapus pada Agustus 2026 demi kesederhanaan [2]. Bab 8 menceritakan kisah di balik keputusan-keputusan itu.

Tabel 3.3 Dua paket sebagaimana digambarkan kartu paket aplikasi pada September 2026.

	Free	Pro
Pindaian dan isi otomatis	15 per hari	pindaian dan konten tanpa batas
Mesin Konten, skema, Pendamping Teknis	ya	ya
Pemantauan terjadwal	1 pemantauan, mingguan	50 pemantauan harian, dengan peringatan
Riwayat	30 hari	riwayat dan tren lengkap, ekspor CSV
Masa berlaku dan harga	gratis	30 hari, Rp50.000 via QRIS, atau ditukar dengan PROFIT

3.9.9 Langkah 9: Menggunakan haknya

Langkah terakhir inilah yang paling sering disembunyikan produk lain. Di tab *Pantau*, sebuah panel bernama **Akun & Data Pribadi** memberi Sari setiap hak yang dijamin Undang-Undang Pelindungan Data Pribadi, UU PDP No. 27/2022, dalam bentuk tombol yang dapat ditekannya sendiri (**Gambar 3.8**): mengekspor seluruh datanya sebagai JSON, mengubah email atau kata sandinya, melihat versi kebijakan mana yang disetujuinya dan kapan, serta menghapus akunnya secara permanen [2], [17].

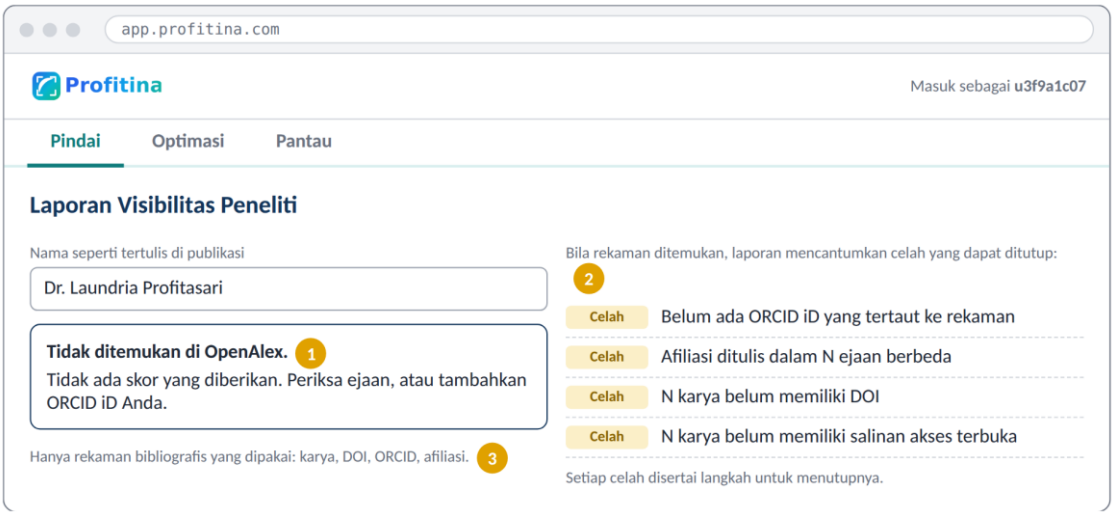


Gambar 3.8 Panel Akun & Data Pribadi, sebagai *wireframe*. (1) Ekspor sebagai JSON; (2) riwayat persetujuan; (3) penghapusan permanen, yang hanya dapat dimulai pengguna; (4) hak-hak ini gratis di semua paket.

Penghapusan membersihkan data pribadinya dari setiap modul dan hanya menyisakan penanda anonim di tempat konten bersama harus tetap utuh bentuknya; hanya Sari yang dapat memulainya, dan admin tidak pernah menghapus "atas namanya". Hak-hak ini, bersama keamanan dan pemulihan akun, tetap **gratis selamanya**, di semua paket [2]. Sejarah produk ini sendiri menyimpan koreksi jujur yang patut diingat: mekanisme penghapusan sudah ada sebelum 19 Agustus 2026, tetapi belum ada tombolnya di antarmuka, sehingga "swalayan" belum benar-benar terwujud. Hal itu baru menjadi kenyataan pada hari panel tersebut dirilis. Bab 11 dikhususkan untuk privasi sejak rancangan.

3.9.10 Perjalanan yang sama bagi seorang peneliti

Perjalanan seorang dosen memiliki bentuk yang sama dengan kata benda yang berbeda. Ia memasukkan namanya sebagaimana tertulis di publikasinya, lalu Laporan Visibilitas Peneliti mencarinya di OpenAlex (**Gambar 3.9**).

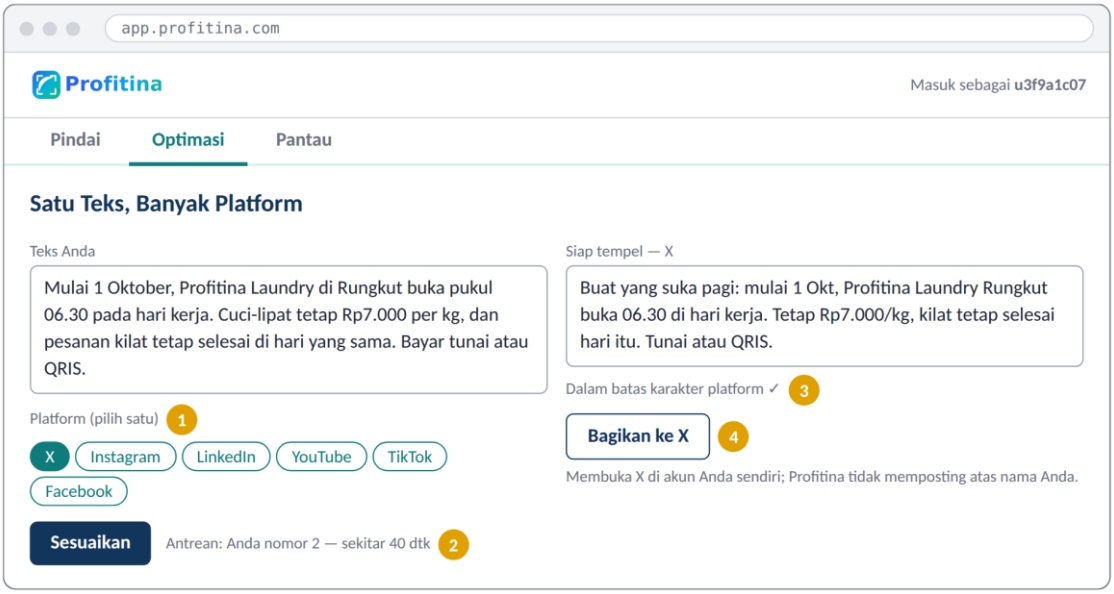


Gambar 3.9 Laporan Visibilitas Peneliti, sebagai *wireframe*. (1) Nama fiktif tidak ditemukan dan tidak diberi skor; (2) untuk rekaman nyata, laporan mencantumkan celah yang dapat ditutup; (3) hanya rekaman bibliografis yang dipakai.

Wireframe itu memperlihatkan kasus yang membuktikan kejujuran metodenya: nama *Dr. Laundria Profitasari*, yang dikarang untuk buku ini, **tidak ditemukan**, dan laporan tidak memberi skor (penanda 1). Untuk rekaman nyata, laporan mencantumkan celah beserta langkah menutupnya (penanda 2): tautkan ORCID iD, seragamkan afiliasi, tambahkan DOI, dan simpan salinan terbuka. Laporan hanya memakai rekaman bibliografis (penanda 3).

3.9.11 Satu Teks, Banyak Platform, dari dalam

Terakhir, **Gambar 3.10** memperlihatkan alat distribusi yang dicoba Sari pada langkah 2, kini dari dalam akunnya.



Gambar 3.10 Satu Teks, Banyak Platform, sebagai *wireframe*. (1) Satu platform dipilih pada satu waktu; (2) posisi antrean dan waktu tunggu ditampilkan; (3) panjang teks diperiksa terhadap batas platform; (4) Bagikan ke X membuka X di akun pengguna sendiri.

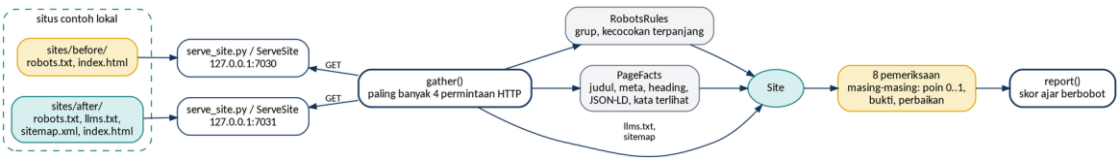
Ia memilih X (penanda 1), menunggu di belakang satu permintaan lain (penanda 2), menerima versi yang muat dalam batas X (penanda 3), lalu menekan *Bagikan ke X*, yang membuka X di akunnya sendiri dengan teks yang siap diposting (penanda 4).

3.10 Lab G1: Pemeriksa Kesiapan AI Mini

3.10.1 Apa yang akan kita bangun, dan milik siapa skornya

Pindaian Sari di Subbab 3.9.4 menemukan lima celah. Dalam lab ini kita menulis program kecil yang menemukan celah-celah sejenis, dengan aturan yang sepenuhnya milik kita dan sepenuhnya terlihat. Inilah latihan unggulan buku ini, karena ia menyentuh hampir semua yang diajarkan buku: HTTP, standar pengecualian robot, penguraian HTML, JSON, rancangan berorientasi objek, sedikit aritmetika, dan banyak kejujuran tentang makna sebuah angka.

Program ini diberi alamat sebuah halaman web. Ia mengambil paling banyak empat hal dari situs halaman itu: `/robots.txt`, halaman itu sendiri, `/llms.txt`, dan, hanya bila `robots.txt` tidak menyebut sitemap, `/sitemap.xml`. Ia mengumpulkan fakta dari semuanya lalu menjalankan delapan pemeriksaan. Setiap pemeriksaan mengembalikan tiga hal: *poin* antara 0 dan 1, *bukti* yang dilihatnya, dan *saran* yang akan menutup celahnya. Laporan mencetak semua itu beserta sebuah **skor ajar** dari 0 sampai 100. **Gambar 3.11** memperlihatkan bagian-bagiannya dan aliran data di antara mereka.



Gambar 3.11 Arsitektur lab G1. Peladen mungil menerbitkan situs contoh di 127.0.0.1; `gather()` mengambil paling banyak empat sumber, `RobotsRules` dan `PageFacts` mengubahnya menjadi `Site`, delapan pemeriksaan menilainya, dan `report()` mencetak hasilnya.

Tabel 3.4 mencantumkan delapan pemeriksaan beserta aturan yang diterapkan masing-masing. Setiap ambang di dalamnya adalah *pilihan ajar*, dibuat agar mudah dipahami dan mudah diubah; tidak satu pun merupakan rekomendasi badan standar, dan tidak satu pun milik Profitina.

Tabel 3.4 Delapan pemeriksaan lab G1 beserta aturannya. Ambang-ambangnya adalah pilihan ajar buku ini.

#	Pemeriksaan	Aturan (poin antara 0 dan 1)	Bukti yang dilaporkan
1	`crawlers`	bagian dari enam *crawler* yang oleh `robots.txt` diizinkan mengambil `/`	keputusan dan aturan penentu untuk setiap *crawler*
2	`sitemap`	1 bila `robots.txt` menyebut sitemap atau `/sitemap.xml` ada	baris sitemap atau status `/sitemap.xml`
3	`llms.txt`	1 bila `/llms.txt` ada dan diawali `# Judul`; 0,5 bila ada tanpa judul	status dan baris judulnya
4	`json-ld`	1 bila setiap blok dapat diurai dan ada simpul dengan `name` dan `url`; 0,5 bila sebagian dapat diurai	blok sah dan tidak sah, `@type` yang ditemukan
5	`headings`	setengah untuk tepat satu `h1`, setengah untuk tidak ada tingkat yang dilompati	jumlah `h1`, judul, dan lompatan tingkat
6	`text`	setengah untuk sedikitnya 120 kata terlihat, setengah untuk rasio teks/HTML sedikitnya 0,20	jumlah kata dan rasionya
7	`answer-1st`	1 bila nama usaha ada di antara 60 kata pertama konten utama	nama yang dipakai dan asalnya
8	`meta`	bagian dari lima hal dasar yang ada: *title* 15+ karakter, deskripsi meta, `lang`, tautan kanonis, `dateModified`	panjang *title* dan jumlahnya

Dengan poin $p_1, \ldots, p_8$ dari kedelapan pemeriksaan dan bobot $w_i \geq 0$ untuk masing-masing, skor ajar adalah rata-rata berbobot, diskalakan ke 100 dan dibulatkan ke bilangan bulat terdekat:

$$S = \left\lfloor 100 \cdot \frac{\sum_{i=1}^{8} w_i \, p_i}{\sum_{i=1}^{8} w_i} + \frac{1}{2} \right\rfloor \tag{3.2}$$

Secara bawaan setiap $w_i = 1$, sehingga S tidak lain adalah rata-rata poin, dikali 100. Pembulatan ke bawah (*floor*) dari "nilai ditambah setengah" membulatkan nilai setengah ke atas, dan rumus ini ditulis lengkap alih-alih diserahkan ke fungsi pustaka karena alasan yang dijelaskan Subbab 3.10.7: `round` milik Python dan `Math.round` milik Java tidak sepakat tentang nilai setengah, padahal kedua versi lab harus mencetak skor yang sama. Pembaca dapat mengubah bobot mana pun dari baris perintah, termasuk memberinya nilai 0 untuk mengabaikan sebuah pemeriksaan, dan laporan selalu mencetak bobot yang dipakainya.

> **PERINGATAN:** Skor ajar pada Persamaan (3.2) adalah alat didaktis. Ia memperlihatkan bagaimana skor kesiapan *mana pun* dibangun, dari fakta yang teramati, aturan, bobot, dan agregasi, sehingga Anda dapat menalar tentangnya. Skor ini bukan skor Profitina, tidak dikalibrasi terhadap kutipan nyata, dan halaman yang meraih 100 menurut delapan aturan ini tetap dapat diabaikan oleh setiap mesin jawaban.

3.10.2 Situs-situs contoh

Lab ini tidak pernah memindai situs pihak ketiga. Memindai peladen orang lain dalam jumlah banyak itu tidak sopan, dan menjalankan latihan buku terhadap web yang *live* akan membuat keluarannya tidak dapat diulang. Sebagai gantinya, folder kode bab ini berisi **situs contoh lokal** yang diterbitkan peladen mungil di komputer Anda sendiri, pada alamat 127.0.0.1, yang tidak

pernah keluar dari komputer Anda. Portanya 7030 sampai 7039, rentang yang dicadangkan buku ini untuk Bab 3.

Folder `sites/before` adalah salinan situs Profitina Laundry sebagaimana ditemukan Sari pada langkah 4. `robots.txt`-nya ditulis oleh panel kendali hosting (**Kode 3.1**).

Kode 3.1 `sites/before/robots.txt`: berkas yang dikelola panel, yang memblokir dua *crawler* AI dan satu *crawler* lain berdasarkan nama, dan mengizinkan semua yang lain.

```
# Managed by the hosting control panel
User-agent: GPTBot
Disallow: /

User-agent: ClaudeBot
Disallow: /

User-agent: CCBot
Disallow: /

User-agent: *
Allow: /
```

Bacalah dengan aturan RFC 9309 dari Bab 2 [7]. `GPTBot` dan `ClaudeBot` masing-masing memiliki grup sendiri, dan di dalam masing-masing grup satu-satunya aturan adalah `Disallow: /`, yang cocok dengan setiap jalur. Setiap *crawler* yang *tidak* disebut jatuh ke grup `*`, yang `Allow: /`-nya mengizinkannya. Jadi `OAI-SearchBot`, `PerplexityBot`, dan `Bingbot` boleh membaca situs itu, sedangkan `GPTBot` dan `ClaudeBot` tidak. Berkas seperti inilah yang mengejutkan tim di balik profitina.com (Subbab 3.2.2): sekilas ia tampak longgar, karena diakhiri dengan `Allow: /`.

Beranda situs itu, **Kode 3.2**, adalah halaman usaha kecil yang khas dari era gambar bergilir.

Kode 3.2 `sites/before/index.html`: beranda sebelum perbaikan. *Title*-nya “Home”, ada dua judul `h1`, JSON-LD-nya berakhir dengan koma, dan nama usaha hanya muncul di *footer*.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Home</title>
<meta name="description" content="Clean clothes, happy life.">
<style>
  body { font-family: sans-serif; margin: 0; background: #fafafa; }
  .hero { background: #0f7c7c; color: white; padding: 48px; }
  .slider img { width: 100%; height: 420px; object-fit: cover; }
  .card { display: inline-block; width: 30%; margin: 1%; }
</style>
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "LocalBusiness",
  "name": "Profitina Laundry",
  "telephone": "+62-812-0000-1234",
}
</script>
<script>
  // image slider and visitor counter
  var slides = ["img/slide1.jpg", "img/slide2.jpg", "img/slide3.jpg"];
  var current = 0;
  function nextSlide() {
    current = (current + 1) % slides.length;
    document.getElementById("slide").src = slides[current];
  }
  setInterval(nextSlide, 4000);
  window.addEventListener("load", function () {
    fetch("/counter").then(function (r) { return r.text(); })
      .then(function (t) { document.getElementById("count").textContent = t; });
  });
</script>
</head>
<body>
<nav><a href="/">Home</a> | <a href="/services">Services</a>
 | <a href="/outlets">Outlets</a> | <a href="/status">Order status</a></nav>
<div class="hero">
  <img src="img/logo.png" alt="logo">
  <h1>Welcome!</h1>
  <p>Clean clothes, happy life.</p>
</div>
<div class="slider"><img id="slide" src="img/slide1.jpg" alt=""></div>
<h1>Our Services</h1>
<h4>Fast</h4>
```

Kode 3.2 (Lanjutan) *sites/before/index.html: beranda sebelum perbaikan. Title-nya "Home", ada dua judul h1, JSON-LD-nya berakhir dengan koma, dan nama usaha hanya muncul di footer.*

```
<p>We are the best and the fastest. Your satisfaction is our priority.</p>
<h4>Clean</h4>
<p>Premium quality at an affordable price. Contact us today!</p>
<h4>Friendly</h4>
<p>Our friendly team is always ready to serve you.</p>
<footer>
  <p>Visitors: <span id="count">0</span></p>
  <p>Copyright 2026 Profitina Laundry. All rights reserved.</p>
</footer>
</body>
</html>
```

Lima masalah bersembunyi di dalamnya. `<title>`-nya “Home”, yang cocok untuk setiap beranda di web sehingga tidak menggambarkan satu pun. Blok JSON-LD mengakhiri properti terakhirnya dengan koma, `"telephone": "+62-812-0000-1234",` lalu `}`; JSON tidak membolehkan hal itu,

sehingga setiap pengurai yang ketat menolak seluruh blok dan data terstrukturnya sama saja dengan tidak ada. Ada dua judul `h1`, "Welcome!" dan "Our Services", dan yang kedua langsung diikuti judul `h4`, melompati dua tingkat. Sebagian besar byte-nya adalah lembar gaya dan skrip untuk penggulir gambar dan penghitung pengunjung. Dan nama usaha tidak muncul di mana pun dalam teks yang terlihat kecuali pada baris hak cipta di *footer*.

Folder `sites/after` berisi situs yang sama setelah langkah 5 Sari. `robots.txt`-nya (**Kode 3.3**) mengizinkan setiap *crawler* membaca halaman publik, menjauhkan dua area privat, memblokir satu *crawler* atas pilihan pemilik, dan menyebut sitemap.

Kode 3.3 `sites/after/robots.txt`: setiap *crawler* boleh membaca halaman publik; `/admin/` dan `/api/` dijauhkan, `CCBot` diblokir atas pilihan pemilik, dan sitemap disebutkan.

```
# robots.txt for Profitina Laundry
User-agent: *
Allow: /
Disallow: /admin/
Disallow: /api/

User-agent: CCBot
Disallow: /

Sitemap: https://laundry.example/sitemap.xml
```

Situs itu juga memiliki `llms.txt` (**Kode 3.4**) dalam format usulan aslinya [13]: judul H1, ringkasan pendek dalam blok kutipan, dan satu bagian berisi tautan beserta catatannya.

Kode 3.4 `sites/after/llms.txt`: judul H1, ringkasan, dan daftar halaman utama situs.

```
# Profitina Laundry

> Wash-and-fold, ironing and dry cleaning in Surabaya, Indonesia.
> Rp7,000 per kilogram; normal orders take two days; QRIS accepted.

## Pages

- [Services and prices](https://laundry.example/services): prices of every service
- [Outlets](https://laundry.example/outlets): addresses and opening hours
- [Order status](https://laundry.example/status): find an order by bill number
```

Berandanya (**Kode 3.5**) kini dibuka dengan paragraf jawab-dulu, memuat satu `h1` dengan bagian-bagian `h2` dan subbagian `h3` di bawahnya, menyatakan `lang="en"`, deskripsi meta, dan alamat kanonis, serta menggambarkan usaha, halaman, dan FAQ-nya dalam satu `@graph` JSON-LD yang sah menggunakan tipe schema.org `DryCleaningOrLaundry`, salah satu jenis `LocalBusiness` [11].

Kode 3.5 `sites/after/index.html`: beranda setelah perbaikan, dengan pembuka jawab-dulu, kerangka judul yang rapi, dan satu graf JSON-LD yang sah.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Profitina Laundry | Wash-and-fold and dry cleaning in Surabaya</title>
<meta name="description" content="Profitina Laundry washes, irons and
  dry-cleans in Surabaya: Rp7,000 per kg, ready in two days, QRIS accepted.">
<link rel="canonical" href="https://laundry.example/">
<link rel="stylesheet" href="/style.css">
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@graph": [
    {
      "@type": "DryCleaningOrLaundry",
      "@id": "https://laundry.example/#shop",
      "name": "Profitina Laundry",
      "url": "https://laundry.example/",
      "telephone": "+62-812-0000-1234",
      "priceRange": "Rp7,000 - Rp75,000",
      "address": {
        "@type": "PostalAddress",
        "streetAddress": "Jl. Penjaringan Sari No. 65, Rungkut",
        "addressLocality": "Surabaya",
        "addressCountry": "ID"
      }
    },
    {
      "@type": "WebPage",
      "url": "https://laundry.example/",
      "about": {"@id": "https://laundry.example/#shop"},
      "dateModified": "2026-09-20"
    },
    {
      "@type": "FAQPage",
      "mainEntity": [
        {"@type": "Question", "name": "How much does wash-and-fold cost?",
         "acceptedAnswer": {"@type": "Answer",
           "text": "Rp7,000 per kilogram, with a minimum of 3 kg per order."}},
        {"@type": "Question", "name": "How long does a normal order take?",
         "acceptedAnswer": {"@type": "Answer",
           "text": "Two days. Express orders are ready the same day."}},
        {"@type": "Question", "name": "Can I pay with QRIS?",
         "acceptedAnswer": {"@type": "Answer",
           "text": "Yes. Every outlet accepts cash and QRIS."}}
      ]
```

Kode 3.5 (Lanjutan) *sites/after/index.html: beranda setelah perbaikan, dengan pembuka jawab-dulu, kerangka judul yang rapi, dan satu graf JSON-LD yang sah.*

```
    }
  ]
}
</script>
</head>
<body>
<nav><a href="/">Home</a> | <a href="/services">Services</a>
 | <a href="/outlets">Outlets</a> | <a href="/status">Order status</a></nav>
<main>
<h1>Profitina Laundry: wash-and-fold and dry cleaning in Surabaya</h1>
<p>Profitina Laundry is a laundry in Rungkut, Surabaya, that washes, irons
and dry-cleans clothes for households and small businesses. Wash-and-fold
costs Rp7,000 per kilogram with a minimum of 3 kg, a normal order is ready
in two days, and express orders are ready the same day. Every outlet
accepts cash and QRIS.</p>
<h2>Services and prices</h2>
<h3>Wash-and-fold</h3>
<p>Rp7,000 per kilogram, minimum 3 kg. Clothes are washed, dried and folded,
and sorted by color before washing.</p>
<h3>Ironing</h3>
<p>Rp5,000 per kilogram, or Rp10,000 per kilogram together with washing.</p>
<h3>Dry cleaning</h3>
<p>From Rp25,000 for a shirt to Rp75,000 for a wedding dress or a suit.</p>
<h2>Frequently asked questions</h2>
<h3>How much does wash-and-fold cost?</h3>
<p>Rp7,000 per kilogram, with a minimum of 3 kg per order.</p>
<h3>How long does a normal order take?</h3>
<p>Two days. Express orders are ready the same day.</p>
<h3>Can I pay with QRIS?</h3>
<p>Yes. Every outlet accepts cash and QRIS.</p>
<h2>Opening hours</h2>
<p>Monday to Saturday 07:00 to 21:00, Sunday 08:00 to 17:00 (WIB).
Last updated on 20 September 2026.</p>
</main>
<footer><p>Profitina Laundry, Jl. Penjaringan Sari No. 65, Rungkut,
Surabaya. Telephone +62-812-0000-1234.</p></footer>
</body>
</html>
```

Perhatikan alamat di dalam berkas-berkas itu: `https://laundry.example/`. Domain `.example` dicadangkan untuk dokumentasi dan contoh sehingga tidak mungkin dimiliki usaha nyata, dan karena itu aman dicetak di buku. Peladen lokal menggantikan alamat publik tersebut: ia menyajikan berkas yang sama dari 127.0.0.1.

3.10.3 Peladen lokal yang mungil

Kedua pustaka standar memuat peladen HTTP kecil yang cukup untuk menerbitkan sebuah folder berkas. **Kode 3.6** memakai `http.server` milik Python [18]. Kelas `SimpleHTTPRequestHandler` sudah tahu cara menjawab `GET /robots.txt` dengan berkas `robots.txt`, dan `GET /` dengan `index.html`; `functools.partial` menetapkan argumen `directory`-nya ke folder kita. Kita menambahkan tiga hal kecil: tipe konten eksplisit dengan `charset=utf-8`, satu baris log per permintaan agar Anda dapat *melihat* setiap permintaan yang dibuat pemeriksa, dan penolakan untuk memakai porta di luar 7030 sampai 7039.

Kode 3.6 `serve_site.py`: peladen berkas statis untuk satu situs contoh, dengan satu baris log per permintaan.

```
# serve_site.py - serve one sample site folder on 127.0.0.1 (Chapter 3 lab)
import functools
import sys
from http.server import SimpleHTTPRequestHandler, ThreadingHTTPServer


class QuietHandler(SimpleHTTPRequestHandler):
    """Serves files from one folder and logs one short line per request."""
    extensions_map = {".html": "text/html; charset=utf-8",
                      ".txt": "text/plain; charset=utf-8",
                      ".xml": "application/xml", "": "application/octet-stream"}
    verbose = True

    def log_request(self, code="-", size="-"):
        if self.verbose:
            status = getattr(code, "value", code)     # HTTPStatus or text
            print(f"  {self.command} {self.path} -> {status}", flush=True)

    def log_message(self, fmt, *args):
        pass                                    # errors are shown by log_request


def make_server(folder, port, verbose=True):
    handler = functools.partial(QuietHandler, directory=folder)
    QuietHandler.verbose = verbose
    return ThreadingHTTPServer(("127.0.0.1", port), handler)


def main():
    if len(sys.argv) != 3:
        print("usage: python serve_site.py <folder> <port 7030-7039>")
        sys.exit(1)
    folder, port = sys.argv[1], int(sys.argv[2])
    if not 7030 <= port <= 7039:
        print("Chapter 3 uses ports 7030 to 7039 only.")
        sys.exit(1)
    server = make_server(folder, port)
    print(f"Serving {folder} at http://127.0.0.1:{port}/ (Ctrl+C stops)")
    try:
        server.serve_forever()
    except KeyboardInterrupt:
        print("Stopped.")
    finally:
        server.server_close()
```

Kode 3.6 (Lanjutan) *serve_site.py: peladen berkas statis untuk satu situs contoh, dengan satu baris log per permintaan.*

```
if __name__ == "__main__":
    main()
```

Pustaka standar Java memiliki peladen kecil sejak Java 6, dalam paket `com.sun.net.httpserver` [19]. Tingkatnya lebih rendah daripada milik Python: kita sendiri yang memetakan jalur permintaan ke berkas (**Kode 3.7**). Dua baris perlu diperhatikan. `root.resolve(path.substring(1)).normalize()` mengubah jalur yang diminta menjadi jalur berkas, lalu `file.startsWith(root)` menolak apa pun yang keluar dari folder, misalnya permintaan `/../../secret.txt`. Peladen yang menerbitkan berkas *selalu* harus melakukan pemeriksaan itu; melupakannya adalah salah satu celah keamanan tertua di web, yang disebut *path traversal*. `SimpleHTTPRequestHandler` milik Python melakukan pemeriksaan yang sama di dalamnya.

Kode 3.7 `ServeSite.java`: peladen yang sama dengan `com.sun.net.httpserver`, termasuk penjaga *path traversal*.

```
// ServeSite.java - serve one sample site folder on 127.0.0.1 (Chapter 3 lab)
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;

public class ServeSite {
    static String type(Path file) {
        String name = file.getFileName().toString();
        if (name.endsWith(".html")) {
            return "text/html; charset=utf-8";
        } else if (name.endsWith(".txt")) {
            return "text/plain; charset=utf-8";
        } else if (name.endsWith(".xml")) {
            return "application/xml";
        }
        return "application/octet-stream";
    }

    static HttpServer makeServer(String folder, int port, boolean verbose)
            throws IOException {
        Path root = Path.of(folder).toAbsolutePath().normalize();
        HttpServer server =
                HttpServer.create(new InetSocketAddress("127.0.0.1", port), 0);
        server.createContext("/", exchange -> {
            String asked = exchange.getRequestURI().getPath();
            String path = asked.endsWith("/") ? asked + "index.html" : asked;
            Path file = root.resolve(path.substring(1)).normalize();
            int status = 404;
            byte[] body = "404 Not Found\n".getBytes(StandardCharsets.UTF_8);
            String type = "text/plain; charset=utf-8";
            if (file.startsWith(root) && Files.isRegularFile(file)) {
                status = 200;                   // never serve outside the folder
                body = Files.readAllBytes(file);
                type = type(file);
            }
            exchange.getResponseHeaders().set("Content-Type", type);
            exchange.sendResponseHeaders(status, body.length);
            try (OutputStream out = exchange.getResponseBody()) {
                out.write(body);
            }
            if (verbose) {
                System.out.println("  " + exchange.getRequestMethod() + " "
```

Kode 3.7 (Lanjutan) *ServeSite.java: peladen yang sama dengan com.sun.net.httpserver, termasuk penjaga path traversal.*

```
                        + asked + " -> " + status);
            }
        });
        return server;
    }

    public static void main(String[] args) throws IOException {
        if (args.length != 2) {
            System.out.println("usage: java ServeSite <folder> <port 7030-7039>");
            System.exit(1);
        }
        int port = Integer.parseInt(args[1]);
        if (port < 7030 || port > 7039) {
            System.out.println("Chapter 3 uses ports 7030 to 7039 only.");
            System.exit(1);
        }
        HttpServer server = makeServer(args[0], port, true);
        server.start();
        System.out.println("Serving " + args[0] + " at http://127.0.0.1:"
                + port + "/ (Ctrl+C stops)");
    }
}
```

3.10.4 Membaca robots.txt menurut aturan

Pemeriksa membutuhkan `robots.txt` dua kali: sekali untuk memutuskan apakah *ia sendiri* boleh mengambil halaman, dan sekali untuk melaporkan apa yang boleh dilakukan enam *crawler* AI dan mesin pencari. Bab 2 menguraikan prosedur RFC 9309; **Pseudokode 3.1** menuliskannya secara tepat, diperluas dengan apa yang harus dilakukan *crawler* ketika berkas itu tidak dapat dibaca.

Pseudokode 3.1 Memutuskan apakah sebuah *crawler* boleh mengambil suatu jalur menurut RFC 9309: kode status lebih dulu, lalu pemilihan grup, lalu aturan cocok yang terpanjang, dengan hasil seri dimenangkan `Allow`.

```
DECIDE(robots, agent, path)
 1  if robots.status = 0 or robots.status ≥ 500     // no answer, or server error
 2      return (disallow, "unreachable: assume disallow")
 3  if robots.status is not a 2xx code               // e.g. 404: no file
 4      return (allow, "no rules apply")
 5  G ← the groups whose User-agent equals agent, ignoring case
 6  if G is empty
 7      G ← the groups whose User-agent is "*"
 8  best ← NIL
 9  for each rule r in the groups of G
10      if r.path is a prefix of path
11          if best = NIL or |r.path| > |best.path|
12                 or (|r.path| = |best.path| and r is Allow)
13              best ← r
14  if best = NIL
15      return (allow, "no rule matches")
16  return (best.kind, best)
```

Baris 1 sampai 4 menangani status HTTP berkas itu, dan mengodekan aturan RFC 9309 yang mudah terbalik [7]. Bila peladen menyatakan berkas itu *tidak ada* (status 4xx seperti 404), *crawler* boleh mengambil apa saja: diam berarti setuju. Bila peladen *gagal* (status 5xx) atau tidak menjawab sama sekali, *crawler* harus menganggap semuanya dilarang: situs yang sedang bermasalah mungkin memiliki `robots.txt` yang melarang persis apa yang diinginkan *crawler*, dan anggapan yang aman adalah bahwa memang demikian. Baris 5 sampai 7 memilih grup: grup-

grup yang menyebut nama *crawler* itu, semuanya digabungkan, atau, hanya bila tidak ada satu pun, grup `*`. Baris 8 sampai 13 mencari aturan cocok yang paling spesifik. Ditulis sebagai rumus, bila M adalah himpunan aturan dalam grup terpilih yang jalurnya merupakan awalan dari jalur yang diminta q, aturan penentunya adalah

$$\hat{r} = \operatorname*{argmax}_{r \in M} \ell(r) \tag{3.3}$$

dengan $\ell(r)$ adalah panjang jalur aturan dalam karakter, dan aturan `Allow` menang bila seri. Bila M kosong, jalur itu diizinkan.

Sepatah kata tentang biaya. Bila grup terpilih memuat R aturan dan jalur aturan terpanjang terdiri atas L karakter, satu keputusan menguji R awalan yang masing-masing paling panjang L karakter, sehingga pada kasus terburuk waktunya sebanding dengan $R \cdot L$. `robots.txt` nyata memuat puluhan aturan, jadi sebuah keputusan terjadi seketika; RFC 9309 meminta *crawler* mengurai sedikitnya 500 kibibyte pertama sebuah berkas, yang dalam praktik membatasi R [7].

Kode 3.8 mengimplementasikan prosedur itu dalam Python. Kelas ini menyimpan grup sebagai pasangan daftar, sehingga grup dengan dua baris `User-agent` yang diikuti aturan-aturan adalah satu grup dengan dua agen. Penanda `last_was_agent` adalah cara pengurai mengetahui bahwa baris `User-agent` baru memulai grup baru: hanya bila baris sebelumnya *bukan* juga baris `User-agent`. `decide` mengembalikan pasangan, yaitu keputusan dan *alasan*, karena pemeriksa yang tidak dapat menjelaskan dirinya hampir tidak berguna.

Kode 3.8 `robots_rules.py`: mengurai `robots.txt` menjadi grup dan memutuskan berdasarkan kode status, pemilihan grup, dan kecocokan terpanjang.

```
# robots_rules.py - decide what robots.txt allows, as RFC 9309 describes it
# (plain path prefixes only; the * and $ patterns are left as an exercise)


class RobotsRules:
    def __init__(self, text, status):
        self.status = status        # HTTP status of /robots.txt, 0 = no answer
        self.groups = []            # list of (agents, rules)
        self.sitemaps = []          # values of Sitemap: lines
        if 200 <= status < 300:
            self._parse(text)

    def _parse(self, text):
        agents, rules, last_was_agent = None, None, False
        for raw in text.splitlines():
            line = raw.split("#", 1)[0].strip()          # drop comments
            key, colon, value = line.partition(":")
            if not colon:
                continue
            key, value = key.strip().lower(), value.strip()
            if key == "user-agent":
                if agents is None or not last_was_agent:  # a new group begins
                    agents, rules = [], []
                    self.groups.append((agents, rules))
                agents.append(value)
                last_was_agent = True
            elif key in ("allow", "disallow"):
                if rules is not None and value:          # empty Disallow: none
                    rules.append((key == "allow", value))
                last_was_agent = False
            elif key == "sitemap":
                self.sitemaps.append(value)
            else:
                last_was_agent = False

    def _groups_for(self, agent):
        named = [(a, r) for (a, r) in self.groups
                 if agent.lower() in (x.lower() for x in a)]
        if named:
            return agent, named
        return "*", [(a, r) for (a, r) in self.groups if "*" in a]
```

Kode 3.8 (Lanjutan) *robots_rules.py: mengurai robots.txt menjadi grup dan memutuskan berdasarkan kode status, pemilihan grup, dan kecocokan terpanjang.*

```
    def decide(self, agent, path):
        """Return (allowed, reason) for one crawler and one path."""
        if self.status == 0 or self.status >= 500:
            return False, "robots.txt unreachable: assume disallow"
        if self.status >= 300:
            return True, f"robots.txt {self.status}: no rules apply"
        name, groups = self._groups_for(agent)
        best = None                                        # (allow, path)
        for _, rules in groups:
            for allow, prefix in rules:
                if not path.startswith(prefix):
                    continue
                if (best is None or len(prefix) > len(best[1])
                        or (len(prefix) == len(best[1]) and allow)):
                    best = (allow, prefix)
        if not groups:
            return True, "no group applies"
        if best is None:
            return True, f'no rule matches in group "{name}"'
        word = "Allow" if best[0] else "Disallow"
        return best[0], f'{word}: {best[1]} in group "{name}"'
```

Mengapa tidak memakai `urllib.robotparser` milik Python sendiri, yang dipakai di Bab 2? Karena, sebagaimana ditunjukkan Bab 2, modul itu menerapkan aturan cocok *pertama* menurut urutan berkas, bukan yang terpanjang, dan tidak menggabungkan beberapa grup yang menyebut *crawler* yang sama; permintaan untuk menyelaraskannya dengan RFC 9309 masih terbuka di proyek CPython [20]. Untuk pemeriksa yang tujuannya melaporkan apa yang akan dilakukan *crawler* yang taat standar, kita menginginkan aturan standar itu persis, dan kita ingin versi Python dan Java sepakat. Soal 3.3 membandingkan keduanya pada berkas yang membuat mereka berbeda pendapat.

Kelas Java, **Kode 3.9**, adalah terjemahan baris demi baris. *Record*, yang tersedia sejak Java 16, memberi kita `Rule`, `Group`, dan `Decision` masing-masing dalam satu baris.

Kode 3.9 `RobotsRules.java`: pengurai dan prosedur keputusan yang sama dalam Java, dengan *record* untuk aturan, grup, dan keputusan.

```java
// RobotsRules.java - decide what robots.txt allows, as RFC 9309 describes it
// (plain path prefixes only; the * and $ patterns are left as an exercise)
import java.util.ArrayList;
import java.util.List;

public class RobotsRules {
    record Rule(boolean allow, String path) { }

    record Group(List<String> agents, List<Rule> rules) { }

    record Decision(boolean allowed, String reason) { }

    final int status;               // HTTP status of /robots.txt, 0 = no answer
    final List<Group> groups = new ArrayList<>();
    final List<String> sitemaps = new ArrayList<>();

    RobotsRules(String text, int status) {
        this.status = status;
        if (status >= 200 && status < 300) {
            parse(text);
        }
    }

    private void parse(String text) {
        Group current = null;
        boolean lastWasAgent = false;
        for (String raw : text.split("\r?\n")) {
            String line = raw.split("#", 2)[0].strip();   // drop comments
            int colon = line.indexOf(':');
            if (colon < 0) {
                continue;
            }
            String key = line.substring(0, colon).strip().toLowerCase();
            String value = line.substring(colon + 1).strip();
            if (key.equals("user-agent")) {
                if (current == null || !lastWasAgent) {     // a new group begins
                    current = new Group(new ArrayList<>(), new ArrayList<>());
                    groups.add(current);
                }
                current.agents().add(value);
                lastWasAgent = true;
            } else if (key.equals("allow") || key.equals("disallow")) {
                if (current != null && !value.isEmpty()) { // empty Disallow: none
                    current.rules().add(new Rule(key.equals("allow"), value));
                }
                lastWasAgent = false;
```

Kode 3.9 (Lanjutan) *RobotsRules.java: pengurai dan prosedur keputusan yang sama dalam Java, dengan record untuk aturan, grup, dan keputusan.*

```
            } else if (key.equals("sitemap")) {
                sitemaps.add(value);
            } else {
                lastWasAgent = false;
            }
        }
    }

    private List<Group> groupsFor(String agent) {
        List<Group> named = new ArrayList<>();
        List<Group> star = new ArrayList<>();
        for (Group g : groups) {
            if (g.agents().stream().anyMatch(a -> a.equalsIgnoreCase(agent))) {
                named.add(g);
            } else if (g.agents().contains("*")) {
                star.add(g);
            }
        }
        return named.isEmpty() ? star : named;
    }

    Decision decide(String agent, String path) {
        if (status == 0 || status >= 500) {
            return new Decision(false, "robots.txt unreachable: assume disallow");
        }
        if (status >= 300) {
            return new Decision(true,
                    "robots.txt " + status + ": no rules apply");
        }
        List<Group> chosen = groupsFor(agent);
        String name = chosen.stream().anyMatch(g -> g.agents().stream()
                .anyMatch(a -> a.equalsIgnoreCase(agent))) ? agent : "*";
        Rule best = null;
        for (Group g : chosen) {
            for (Rule r : g.rules()) {
                if (!path.startsWith(r.path())) {
                    continue;
                }
                int len = r.path().length();
                if (best == null || len > best.path().length()
                        || (len == best.path().length() && r.allow())) {
                    best = r;
                }
            }
        }
        if (chosen.isEmpty()) {
```

Kode 3.9 (Lanjutan) *RobotsRules.java: pengurai dan prosedur keputusan yang sama dalam Java, dengan record untuk aturan, grup, dan keputusan.*

```
            return new Decision(true, "no group applies");
        }
        if (best == null) {
            return new Decision(true,
                    "no rule matches in group \"" + name + "\"");
        }
        String word = best.allow() ? "Allow" : "Disallow";
        return new Decision(best.allow(), word + ": " + best.path()
                + " in group \"" + name + "\"");
    }
}
```

3.10.5 Mengumpulkan fakta sebuah halaman

Masukan kedua adalah halaman itu sendiri. Dari HTML-nya kita memerlukan: *title*, deskripsi meta, atribut `lang`, tautan kanonis, daftar judul beserta tingkatnya, teks setiap blok JSON-LD, dan *kata-kata yang terlihat*, yaitu kata-kata yang akan dilihat pembaca manusia sebagai konten. Pengurai HTML lengkap membangun pohon seluruh dokumen; kita memerlukan jauh lebih sedikit. Kita membaca dokumen sekali, dari awal sampai akhir, dan menanggapi tiga jenis peristiwa: tag pembuka, tag penutup, dan rangkaian teks di antara tag. Gaya ini disebut penguraian *berbasis peristiwa* atau *streaming*, dan waktunya sebanding dengan panjang dokumen, karena setiap karakter diperiksa sejumlah tetap kali.

Kata mana yang "terlihat"? Jelas bukan yang berada di dalam `<script>` dan `<style>`, dan bukan yang di `<head>`. Kita juga melewati `<nav>` dan `<footer>`, karena tautan navigasi dan baris hak cipta muncul di setiap halaman situs dan tidak berkata apa-apa tentang halaman ini; melewatinya adalah versi sederhana dari apa yang oleh alat ekstraksi disebut *boilerplate removal* (pembuangan teks templat). Dengan teks terlihat T, yaitu kata-kata yang digabung dengan satu spasi, dan HTML H sebagai byte, **rasio teks terhadap HTML** adalah

$$\rho = \frac{n_T}{n_H} \tag{3.4}$$

dengan n_T banyaknya karakter T dan n_H banyaknya byte H dalam pengodean UTF-8. Halaman yang sebagian besar berupa teks memiliki ρ tinggi; halaman yang sebagian besar berupa skrip, gaya, dan markah memiliki ρ rendah. Pada halaman contoh, yang hanya memuat karakter dari rentang ASCII, jumlah karakter dan byte sama. Pemeriksaan `text` pada **Tabel 3.4** meminta $\rho \geq 0.20$ dan sedikitnya 120 kata, dua ambang ajar.

Pustaka standar Python menyediakan pengurai berbasis peristiwa dalam `html.parser.HTMLParser` [21]. Kita membuat subkelasnya dan menimpa tiga metode peristiwanya (**Kode 3.10**). Pengaturan `convert_charrefs=True` membuat pengurai menerjemahkan entitas seperti `&` dan menyerahkan teks di antara dua tag dalam satu potong.

Kode 3.10 `page_facts.py`: pengumpul *title*, tag meta, judul, blok JSON-LD, dan kata terlihat berbasis peristiwa, dibangun di atas `html.parser`.

```python
# page_facts.py - collect the facts about one HTML page that the checks need
from html.parser import HTMLParser

SKIP = {"head", "script", "style", "noscript", "template", "nav", "footer"}
HEADINGS = {"h1": 1, "h2": 2, "h3": 3, "h4": 4, "h5": 5, "h6": 6}


class PageFacts(HTMLParser):
    def __init__(self, html_text):
        super().__init__(convert_charrefs=True)
        self.html_bytes = len(html_text.encode("utf-8"))
        self.title = ""
        self.description = None
        self.lang = None
        self.canonical = None
        self.headings = []          # (level, text) in document order
        self.jsonld = []            # the text of every JSON-LD block
        self.words = []             # visible words, boilerplate skipped
        self.main_words = []        # visible words inside <main>
        self.has_main = False
        self._skip = 0              # > 0 while inside a SKIP element
        self._in_main = 0
        self._in_title = False
        self._in_jsonld = False
        self._heading = None        # [level, words] while inside h1..h6
        self.feed(html_text)
        self.close()

    def handle_starttag(self, tag, attrs):
        a = {k: (v or "") for k, v in attrs}
        if tag == "html":
            self.lang = a.get("lang") or None
        elif tag == "title":
            self._in_title = True
        elif tag == "meta" and a.get("name", "").lower() == "description":
            self.description = " ".join(a.get("content", "").split())
        elif tag == "link" and "canonical" in a.get("rel", "").lower().split():
            self.canonical = a.get("href")
        elif tag == "script" and a.get("type", "") == "application/ld+json":
            self._in_jsonld = True
            self.jsonld.append("")
        elif tag == "main":
            self.has_main = True
            self._in_main += 1
        elif tag in HEADINGS:
            self._heading = [HEADINGS[tag], []]
```

Kode 3.10 (Lanjutan) *page_facts.py: pengumpul title, tag meta, judul, blok JSON-LD, dan kata terlihat berbasis peristiwa, dibangun di atas html.parser.*

```
        if tag in SKIP:
            self._skip += 1

    def handle_endtag(self, tag):
        if tag in SKIP and self._skip > 0:
            self._skip -= 1
        if tag == "title":
            self._in_title = False
        elif tag == "script":
            self._in_jsonld = False
        elif tag == "main" and self._in_main > 0:
            self._in_main -= 1
        elif tag in HEADINGS and self._heading is not None:
            self.headings.append((self._heading[0], " ".join(self._heading[1])))
            self._heading = None

    def handle_data(self, data):
        if self._in_title:
            self.title += " ".join(data.split())
        if self._in_jsonld:
            self.jsonld[-1] += data
        if self._skip > 0:
            return
        words = data.split()
        self.words.extend(words)
        if self._in_main > 0:
            self.main_words.extend(words)
        if self._heading is not None:
            self._heading[1].extend(words)

    def visible_text(self):
        return " ".join(self.words)

    def lead_words(self, n):
        """The first n words of the main content (the whole body if no <main>)."""
        return (self.main_words if self.has_main else self.words)[:n]
```

Pencacah `_skip`, bukan sekadar penanda benar/salah, menangani sarang: seandainya `<nav>` berada di dalam `<footer>`, keluar dari `<nav>` tidak boleh membuat teks *footer* menjadi terlihat. Gagasan yang sama, mencacah alih-alih menandai, dipakai untuk `<main>`. Metode `lead_words(n)` mengembalikan n kata terlihat pertama dari elemen `<main>` bila halaman memilikinya, dan dari seluruh badan halaman bila tidak; pemeriksaan `answer-1st` memakainya.

Pustaka standar Java tidak memiliki pengurai HTML, sehingga **Kode 3.11** memuat pengurai kecil: ekspresi reguler `TAG` menemukan tag, teks di antara dua kecocokan adalah rangkaian teks, dan `ATTR` memecah atribut sebuah tag. Dua rincian menjaganya setia pada versi Python. Komentar dan deklarasi `<!DOCTYPE>` dibuang lebih dulu. Dan isi `<script>` serta `<style>` dibaca mentah sampai tag penutup yang sesuai, karena sebuah skrip dapat memuat karakter `<` yang bukan tag; HTML menyebut elemen seperti ini elemen *raw text*, dan pengurai Python memperlakukannya dengan cara yang sama.

Kode 3.11 `PageFacts.java`: pengumpul yang sama dalam Java, dengan pemecah token berbasis ekspresi reguler dan penanganan teks mentah untuk `script` dan `style`.

```java
// PageFacts.java - collect the facts about one HTML page that the checks need
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class PageFacts {
    static final Set<String> SKIP = Set.of("head", "script", "style",
            "noscript", "template", "nav", "footer");
    static final Pattern TAG = Pattern.compile(
            "<(/?)([a-zA-Z][a-zA-Z0-9]*)((?:[^>\"']|\"[^\"]*\"|'[^']*')*)>");
    static final Pattern ATTR = Pattern.compile("([a-zA-Z_:][-a-zA-Z0-9_:.]*)"
            + "(?:\\s*=\\s*(\"[^\"]*\"|'[^']*'|[^\\s\"'>]+))?");

    record Heading(int level, String text) { }

    final int htmlBytes;
    String title = "";
    String description = null;
    String lang = null;
    String canonical = null;
    final List<Heading> headings = new ArrayList<>();
    final List<String> jsonld = new ArrayList<>();
    final List<String> words = new ArrayList<>();      // boilerplate skipped
    final List<String> mainWords = new ArrayList<>();  // inside <main>
    boolean hasMain = false;
    private int skip = 0;
    private int inMain = 0;
    private boolean inTitle = false;
    private boolean inJsonld = false;
    private int headingLevel = 0;                      // 0 = not in a heading
    private final List<String> headingWords = new ArrayList<>();
```

Kode 3.11 (Lanjutan) *PageFacts.java: pengumpul yang sama dalam Java, dengan pemecah token berbasis ekspresi reguler dan penanganan teks mentah untuk script dan style.*

```
    PageFacts(String html) {
        htmlBytes = html.getBytes(StandardCharsets.UTF_8).length;
        String s = html.replaceAll("(?s)<!--.*?-->", "")   // comments
                       .replaceAll("<![^>]*>", "");        // <!DOCTYPE html>
        String lower = s.toLowerCase();
        Matcher m = TAG.matcher(s);
        int pos = 0;
        while (m.find(pos)) {
            data(s.substring(pos, m.start()));
            String name = m.group(2).toLowerCase();
            pos = m.end();
            if (!m.group(1).isEmpty()) {
                end(name);
                continue;
            }
            start(name, attributes(m.group(3)));
            if (name.equals("script") || name.equals("style")) {
                int close = lower.indexOf("</" + name, pos);  // raw text inside
                close = close < 0 ? s.length() : close;
                data(s.substring(pos, close));
                pos = close;
            }
        }
        data(s.substring(pos));
    }

    static Map<String, String> attributes(String text) {
        Map<String, String> map = new HashMap<>();
        Matcher m = ATTR.matcher(text);
        while (m.find()) {
            String v = m.group(2) == null ? "" : m.group(2);
            if (v.startsWith("\"") || v.startsWith("'")) {
                v = v.substring(1, v.length() - 1);
            }
            map.putIfAbsent(m.group(1).toLowerCase(), decode(v));
        }
        return map;
    }
```

Kode 3.11 (Lanjutan) *PageFacts.java: pengumpul yang sama dalam Java, dengan pemecah token berbasis ekspresi reguler dan penanganan teks mentah untuk script dan style.*

```
    static String decode(String t) {                    // the common entities
        Matcher m = Pattern.compile("&#(\\d+);").matcher(t);
        t = m.replaceAll(r -> Character.toString(Integer.parseInt(r.group(1))));
        return t.replace("&lt;", "<").replace("&gt;", ">").replace("&quot;", "\"")
                .replace("&#39;", "'").replace(" ", "\u00a0")
                .replace("&amp;", "&");
    }

    static List<String> split(String text) {
        List<String> out = new ArrayList<>();
        for (String w : text.strip().split("\\s+")) {
            if (!w.isEmpty()) {
                out.add(w);
            }
        }
        return out;
    }

    private void start(String tag, Map<String, String> a) {
        if (tag.equals("html")) {
            lang = a.getOrDefault("lang", "").isEmpty() ? null : a.get("lang");
        } else if (tag.equals("title")) {
            inTitle = true;
        } else if (tag.equals("meta")
                && a.getOrDefault("name", "").equalsIgnoreCase("description")) {
            description = String.join(" ", split(a.getOrDefault("content", "")));
        } else if (tag.equals("link") && List.of(a.getOrDefault("rel", "")
                .toLowerCase().split("\\s+")).contains("canonical")) {
            canonical = a.get("href");
        } else if (tag.equals("script")
                && a.getOrDefault("type", "").equals("application/ld+json")) {
            inJsonld = true;
            jsonld.add("");
        } else if (tag.equals("main")) {
            hasMain = true;
            inMain++;
        } else if (tag.matches("h[1-6]")) {
            headingLevel = tag.charAt(1) - '0';
            headingWords.clear();
        }
        if (SKIP.contains(tag)) {
            skip++;
        }
    }
```

Kode 3.11 (Lanjutan) *PageFacts.java: pengumpul yang sama dalam Java, dengan pemecah token berbasis ekspresi reguler dan penanganan teks mentah untuk script dan style.*

```
    private void end(String tag) {
        if (SKIP.contains(tag) && skip > 0) {
            skip--;
        }
        if (tag.equals("title")) {
            inTitle = false;
        } else if (tag.equals("script")) {
            inJsonld = false;
        } else if (tag.equals("main") && inMain > 0) {
            inMain--;
        } else if (tag.matches("h[1-6]") && headingLevel > 0) {
            String text = String.join(" ", headingWords);
            headings.add(new Heading(headingLevel, text));
            headingLevel = 0;
        }
    }

    private void data(String raw) {
        String text = inJsonld ? raw : decode(raw);
        if (inTitle) {
            title += String.join(" ", split(text));
        }
        if (inJsonld) {
            jsonld.set(jsonld.size() - 1, jsonld.get(jsonld.size() - 1) + text);
        }
        if (skip > 0) {
            return;
        }
        List<String> w = split(text);
        words.addAll(w);
        if (inMain > 0) {
            mainWords.addAll(w);
        }
        if (headingLevel > 0) {
            headingWords.addAll(w);
        }
    }

    String visibleText() {
        return String.join(" ", words);
    }
```

Kode 3.11 (Lanjutan) *PageFacts.java: pengumpul yang sama dalam Java, dengan pemecah token berbasis ekspresi reguler dan penanganan teks mentah untuk script dan style.*

```
    List<String> leadWords(int n) {
        List<String> source = hasMain ? mainWords : words;
        return source.subList(0, Math.min(n, source.size()));
    }
}
```

Pemecah token ini cukup untuk halaman yang tersusun baik seperti contoh kita; ia bukan pengganti pengurai sungguhan di web terbuka, tempat tag dibiarkan terbuka, atribut tidak diberi tanda kutip, dan entitasnya beraneka ragam. Soal 3.11 menanyakan di mana ia gagal.

Untuk JSON, Python memiliki `json.loads`. Pustaka standar Java juga tidak memiliki pengurai JSON, sehingga versi Java memakai ulang `MiniJson.java`, pengurai kecil yang ditulis pada lab Bab 2, yang disalin tanpa perubahan ke folder bab ini. Keduanya menolak koma di akhir pada halaman contoh, sebagaimana dituntut standar JSON.

3.10.6 Delapan pemeriksaan

Setelah fakta terkumpul, setiap pemeriksaan adalah fungsi kecil dari sebuah `Site`, yaitu rekaman segala yang telah diambil, ke sebuah `CheckResult`. **Kode 3.12** memuat kedelapannya, sesuai urutan **Tabel 3.4**.

Kode 3.12 `checks.py`: delapan pemeriksaan. Masing-masing menerima sebuah `Site` dan mengembalikan poin, bukti, dan saran.

```
# checks.py - the eight checks of the lab; each returns a CheckResult
import json
from dataclasses import dataclass

CRAWLERS = ["GPTBot", "OAI-SearchBot", "ClaudeBot", "PerplexityBot",
            "Google-Extended", "Bingbot"]
MIN_WORDS, MIN_RATIO, LEAD = 120, 0.20, 60      # our own teaching choices


@dataclass
class Site:                     # everything fetched for one page
    url: str
    path: str
    facts: object               # PageFacts
    robots: object              # RobotsRules
    llms_status: int
    llms_text: str
    sitemap_status: int         # 0 when not fetched


@dataclass
class CheckResult:
    name: str
    points: float               # 0.0 .. 1.0
    evidence: str
    advice: str                 # empty when there is nothing to fix


def check_crawlers(site):
    blocked = [c for c in CRAWLERS if not site.robots.decide(c, "/")[0]]
    ok = len(CRAWLERS) - len(blocked)
    return CheckResult("crawlers", ok / len(CRAWLERS),
                       f"{ok} of {len(CRAWLERS)} crawlers may fetch /",
                       "unblock " + ", ".join(blocked) if blocked else "")


def check_sitemap(site):
    if site.robots.sitemaps:
        return CheckResult("sitemap", 1.0, "robots.txt names "
                           + site.robots.sitemaps[0], "")
    if site.sitemap_status == 200:
        return CheckResult("sitemap", 1.0, "/sitemap.xml exists", "")
    return CheckResult("sitemap", 0.0, "no Sitemap line, /sitemap.xml "
                       f"{site.sitemap_status}",
                       "publish /sitemap.xml and name it in robots.txt")
```

Kode 3.12 (Lanjutan) *checks.py: delapan pemeriksaan. Masing-masing menerima sebuah Site dan mengembalikan poin, bukti, dan saran.*

```
def check_llms(site):
    if site.llms_status != 200:
        return CheckResult("llms.txt", 0.0, f"/llms.txt {site.llms_status}",
                           "optional: publish /llms.txt (few AI bots read it)")
    lines = [s for s in site.llms_text.splitlines() if s.strip()]
    if lines and lines[0].startswith("# "):
        return CheckResult("llms.txt", 1.0, "present, title " + lines[0], "")
    return CheckResult("llms.txt", 0.5, "present, but no '# Title' line first",
                       "start /llms.txt with a '# Name' line")


def jsonld_nodes(doc):
    """The top-level nodes of one JSON-LD document, @graph members included."""
    nodes = doc if isinstance(doc, list) else [doc]
    out = []
    for n in nodes:
        if isinstance(n, dict):
            if "@type" in n:
                out.append(n)
            out.extend(g for g in n.get("@graph", []) if isinstance(g, dict))
    return out


def valid_nodes(site):
    """All top-level nodes of the JSON-LD blocks that parse."""
    nodes = []
    for block in site.facts.jsonld:
        try:
            nodes += jsonld_nodes(json.loads(block))
        except ValueError:
            pass
    return nodes


def check_jsonld(site):
    valid, invalid, nodes = 0, 0, []
    for block in site.facts.jsonld:
        try:
            nodes += jsonld_nodes(json.loads(block))
            valid += 1
        except ValueError:
            invalid += 1
    if valid + invalid == 0:
        return CheckResult("json-ld", 0.0, "no JSON-LD block",
                           "describe the business with schema.org JSON-LD")
```

Kode 3.12 (Lanjutan) *checks.py: delapan pemeriksaan. Masing-masing menerima sebuah Site dan mengembalikan poin, bukti, dan saran.*

```
    types = []
    for n in nodes:
        t = n.get("@type", [])
        for x in t if isinstance(t, list) else [t]:
            if x not in types:
                types.append(x)
    named = any("name" in n and "url" in n for n in nodes)
    found = ", ".join(types) or "no types"
    evidence = f"{valid} valid, {invalid} invalid: {found}"
    if valid and not invalid and named:
        return CheckResult("json-ld", 1.0, evidence, "")
    advice = "repair the invalid JSON" if invalid else "add name and url"
    return CheckResult("json-ld", 0.5 if valid else 0.0, evidence, advice)


def check_headings(site):
    levels = [lvl for lvl, _ in site.facts.headings]
    h1 = levels.count(1)
    jumps = [(a, b) for a, b in zip([1] + levels, levels) if b > a + 1]
    points = ((h1 == 1) + (not jumps)) / 2
    evidence = f"{h1} h1, {len(levels)} headings, level jumps: {len(jumps)}"
    advice = []
    if h1 != 1:
        advice.append("use exactly one h1")
    if jumps:
        advice.append(f"h{jumps[0][0]} is followed by h{jumps[0][1]}")
    return CheckResult("headings", points, evidence, "; ".join(advice))


def check_text(site):
    f = site.facts
    ratio = len(f.visible_text()) / f.html_bytes
    points = ((len(f.words) >= MIN_WORDS) + (ratio >= MIN_RATIO)) / 2
    return CheckResult("text", points,
                       f"{len(f.words)} words, text/HTML ratio {ratio:.2f}",
                       "" if points == 1 else
                       f"aim for {MIN_WORDS}+ words and ratio {MIN_RATIO:.2f}+")


def entity_name(site):
    for n in valid_nodes(site):
        if isinstance(n.get("name"), str):
            return n["name"], "JSON-LD"
    return site.facts.title.split(" | ")[0].strip(), "<title>"
```

Kode 3.12 (Lanjutan) *checks.py: delapan pemeriksaan. Masing-masing menerima sebuah Site dan mengembalikan poin, bukti, dan saran.*

```
def check_answer_first(site):
    name, source = entity_name(site)
    lead = " ".join(site.facts.lead_words(LEAD)).lower()
    found = bool(name) and name.lower() in lead
    return CheckResult("answer-1st", 1.0 if found else 0.0,
                       f'"{name}" (from {source}) '
                       + ("is" if found else "is not")
                       + f" in the first {LEAD} words",
                       "" if found else "open with who you are and what you do")


def check_meta(site):
    f = site.facts
    dated = [n["dateModified"] for n in valid_nodes(site) if "dateModified" in n]
    have = {"title": len(f.title) >= 15, "description": bool(f.description),
            "lang": bool(f.lang), "canonical": bool(f.canonical),
            "dateModified": bool(dated)}
    missing = [k for k, ok in have.items() if not ok]
    found = len(have) - len(missing)
    return CheckResult("meta", found / len(have),
                       f"title {len(f.title)} chars; {found} of {len(have)} "
                       "present"
                       + (f" (dateModified {dated[0]})" if dated else ""),
                       "add " + ", ".join(missing) if missing else "")


NAMES = ["crawlers", "sitemap", "llms.txt", "json-ld", "headings", "text",
         "answer-1st", "meta"]
CHECKS = [check_crawlers, check_sitemap, check_llms, check_jsonld,
          check_headings, check_text, check_answer_first, check_meta]
```

Tiga aturannya layak ditulis sebagai rumus, karena rumus membuat perilakunya pada batas-batas menjadi tidak bermakna ganda. Bila a dari enam *crawler* boleh mengambil `/`, pemeriksaan *crawler* memberikan

$$p_{\text{crawl}} = \frac{a}{6} \tag{3.5}$$

sehingga setiap *crawler* yang diblokir mengurangi seperenam poin. Pemeriksaan judul memakai kurung Iverson $[P]$, yang bernilai 1 bila pernyataan P benar dan 0 bila salah. Dengan h banyaknya judul `h1` dan J banyaknya lompatan tingkat, yaitu judul yang lebih dalam lebih dari satu tingkat dibandingkan judul sebelumnya,

$$p_{\text{head}} = \frac{[h = 1] + [J = 0]}{2} \tag{3.6}$$

Untuk halaman “before”, $h = 2$ dan $J = 1$ (`h1` “Our Services” langsung diikuti `h4`), sehingga $p_{\text{head}} = (0 + 0)/2 = 0$. Untuk halaman “after”, $h = 1$, dan kerangka `h1`, `h2`, `h3`, `h3`, `h3`, `h2`, … tidak pernah turun lebih dari satu tingkat sekaligus, sehingga $J = 0$ dan $p_{\text{head}} = 1$. Perhatikan bahwa naik, dari `h3` kembali ke `h2`, tidak pernah dihitung sebagai lompatan. Terakhir, bila k dari lima hal dasar ada,

$$p_{\text{meta}} = \frac{k}{5} \tag{3.7}$$

Fungsi `jsonld_nodes` dan `valid_nodes` perlu dijelaskan. Blok JSON-LD dapat berupa satu objek, daftar objek, atau objek dengan daftar `@graph`, yaitu bentuk yang dipakai halaman “after” kita dan

bentuk yang ditulis generator Bab 2; `jsonld_nodes` meratakan ketiganya menjadi satu daftar simpul tingkat atas. `entity_name` lalu mengambil nama usaha dari simpul pertama yang memiliki `name`, dan kembali ke *title* halaman, dipotong pada `" | "` pertama, bila tidak ada JSON-LD yang sah. Karena cadangan itulah halaman "before" akhirnya diuji dengan nama "Home".

Versi Java (**Kode 3.13**) mendefinisikan antarmuka fungsional `Check` dengan satu metode `run`. Daftar `CHECKS` menampung delapan rujukan metode, `Checks::crawlers` dan seterusnya, yang masing-masing *adalah* sebuah `Check`. Inilah bentuk berorientasi objek lab ini: satu antarmuka, banyak implementasi, masing-masing dinilai dan dijelaskan secara mandiri, dan laporan yang tidak perlu tahu pemeriksaan apa saja yang ada. Menambah pemeriksaan kesembilan berarti menulis satu metode dan menambah satu entri pada daftar.

Kode 3.13 `Checks.java`: *record* `Site` dan `CheckResult`, antarmuka `Check`, dan delapan pemeriksaan.

```
// Checks.java - the eight checks of the lab; each returns a CheckResult
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

public class Checks {
    static final List<String> CRAWLERS = List.of("GPTBot", "OAI-SearchBot",
            "ClaudeBot", "PerplexityBot", "Google-Extended", "Bingbot");
    static final int MIN_WORDS = 120;                  // our own teaching choices
    static final double MIN_RATIO = 0.20;
    static final int LEAD = 60;

    record Site(String url, String path, PageFacts facts, RobotsRules robots,
                int llmsStatus, String llmsText, int sitemapStatus) { }

    record CheckResult(String name, double points, String evidence,
                       String advice) { }

    interface Check {
        CheckResult run(Site site);
    }

    static CheckResult crawlers(Site site) {
        List<String> blocked = new ArrayList<>();
        for (String c : CRAWLERS) {
            if (!site.robots().decide(c, "/").allowed()) {
                blocked.add(c);
            }
        }
        int ok = CRAWLERS.size() - blocked.size();
        return new CheckResult("crawlers", (double) ok / CRAWLERS.size(),
                ok + " of " + CRAWLERS.size() + " crawlers may fetch /",
                blocked.isEmpty() ? "" : "unblock " + String.join(", ", blocked));
    }
```

Kode 3.13 (Lanjutan) *Checks.java: record Site dan CheckResult, antarmuka Check, dan delapan pemeriksaan.*

```
    static CheckResult sitemap(Site site) {
        if (!site.robots().sitemaps.isEmpty()) {
            return new CheckResult("sitemap", 1.0, "robots.txt names "
                    + site.robots().sitemaps.get(0), "");
        }
        if (site.sitemapStatus() == 200) {
            return new CheckResult("sitemap", 1.0, "/sitemap.xml exists", "");
        }
        return new CheckResult("sitemap", 0.0, "no Sitemap line, /sitemap.xml "
                + site.sitemapStatus(),
                "publish /sitemap.xml and name it in robots.txt");
    }

    static CheckResult llms(Site site) {
        if (site.llmsStatus() != 200) {
            return new CheckResult("llms.txt", 0.0,
                    "/llms.txt " + site.llmsStatus(),
                    "optional: publish /llms.txt (few AI bots read it)");
        }
        List<String> lines = site.llmsText().lines()
                .filter(s -> !s.isBlank()).toList();
        if (!lines.isEmpty() && lines.get(0).startsWith("# ")) {
            return new CheckResult("llms.txt", 1.0, "present, title "
                    + lines.get(0), "");
        }
        return new CheckResult("llms.txt", 0.5,
                "present, but no '# Title' line first",
                "start /llms.txt with a '# Name' line");
    }

    /** The top-level nodes of one JSON-LD document, @graph members included. */
    static List<Map<?, ?>> jsonldNodes(Object doc) {
        List<?> nodes = doc instanceof List<?> list ? list : List.of(doc);
        List<Map<?, ?>> out = new ArrayList<>();
        for (Object n : nodes) {
            if (n instanceof Map<?, ?> map) {
                if (map.containsKey("@type")) {
                    out.add(map);
                }
                if (map.get("@graph") instanceof List<?> graph) {
                    for (Object g : graph) {
                        if (g instanceof Map<?, ?> gm) {
                            out.add(gm);
                        }
                    }
                }
```

Kode 3.13 (Lanjutan) *Checks.java: record Site dan CheckResult, antarmuka Check, dan delapan pemeriksaan.*

```
            }
        }
        return out;
    }

    /** All top-level nodes of the JSON-LD blocks that parse. */
    static List<Map<?, ?>> validNodes(Site site) {
        List<Map<?, ?>> nodes = new ArrayList<>();
        for (String block : site.facts().jsonld) {
            try {
                nodes.addAll(jsonldNodes(MiniJson.parse(block)));
            } catch (RuntimeException e) {
                // not JSON: check jsonld counts it as invalid
            }
        }
        return nodes;
    }

    static CheckResult jsonld(Site site) {
        int valid = 0;
        int invalid = 0;
        List<Map<?, ?>> nodes = new ArrayList<>();
        for (String block : site.facts().jsonld) {
            try {
                nodes.addAll(jsonldNodes(MiniJson.parse(block)));
                valid++;
            } catch (RuntimeException e) {
                invalid++;
            }
        }
        if (valid + invalid == 0) {
            return new CheckResult("json-ld", 0.0, "no JSON-LD block",
                    "describe the business with schema.org JSON-LD");
        }
        List<String> types = new ArrayList<>();
        for (Map<?, ?> n : nodes) {
            Object t = n.get("@type");
            List<?> all = t instanceof List<?> list ? list
                    : (t == null ? List.of() : List.of(t));
            for (Object x : all) {
                if (!types.contains(x.toString())) {
                    types.add(x.toString());
                }
            }
        }
        boolean named = nodes.stream()
```

Kode 3.13 (Lanjutan) *Checks.java: record Site dan CheckResult, antarmuka Check, dan delapan pemeriksaan.*

```
                .anyMatch(n -> n.containsKey("name") && n.containsKey("url"));
        String evidence = valid + " valid, " + invalid + " invalid: "
                + (types.isEmpty() ? "no types" : String.join(", ", types));
        if (valid > 0 && invalid == 0 && named) {
            return new CheckResult("json-ld", 1.0, evidence, "");
        }
        String advice = invalid > 0 ? "repair the invalid JSON"
                : "add name and url";
        return new CheckResult("json-ld", valid > 0 ? 0.5 : 0.0, evidence,
                advice);
    }

    static CheckResult headings(Site site) {
        List<Integer> levels = site.facts().headings.stream()
                .map(PageFacts.Heading::level).toList();
        long h1 = levels.stream().filter(l -> l == 1).count();
        List<int[]> jumps = new ArrayList<>();
        int previous = 1;
        for (int level : levels) {
            if (level > previous + 1) {
                jumps.add(new int[] {previous, level});
            }
            previous = level;
        }
        double points = ((h1 == 1 ? 1 : 0) + (jumps.isEmpty() ? 1 : 0)) / 2.0;
        String evidence = h1 + " h1, " + levels.size()
                + " headings, level jumps: " + jumps.size();
        List<String> advice = new ArrayList<>();
        if (h1 != 1) {
            advice.add("use exactly one h1");
        }
        if (!jumps.isEmpty()) {
            int[] j = jumps.get(0);
            advice.add("h" + j[0] + " is followed by h" + j[1]);
        }
        return new CheckResult("headings", points, evidence,
                String.join("; ", advice));
    }
```

Kode 3.13 (Lanjutan) *Checks.java: record Site dan CheckResult, antarmuka Check, dan delapan pemeriksaan.*

```
    static CheckResult text(Site site) {
        PageFacts f = site.facts();
        double ratio = (double) f.visibleText().length() / f.htmlBytes;
        double points = ((f.words.size() >= MIN_WORDS ? 1 : 0)
                + (ratio >= MIN_RATIO ? 1 : 0)) / 2.0;
        return new CheckResult("text", points, String.format(Locale.ROOT,
                "%d words, text/HTML ratio %.2f", f.words.size(), ratio),
                points == 1 ? "" : String.format(Locale.ROOT,
                "aim for %d+ words and ratio %.2f+", MIN_WORDS, MIN_RATIO));
    }

    static String[] entityName(Site site) {
        for (Map<?, ?> n : validNodes(site)) {
            if (n.get("name") instanceof String name) {
                return new String[] {name, "JSON-LD"};
            }
        }
        return new String[] {site.facts().title.split(" \\| ")[0].strip(),
            "<title>"};
    }

    static CheckResult answerFirst(Site site) {
        String[] entity = entityName(site);
        String lead = String.join(" ", site.facts().leadWords(LEAD))
                .toLowerCase();
        boolean found = !entity[0].isEmpty()
                && lead.contains(entity[0].toLowerCase());
        return new CheckResult("answer-1st", found ? 1.0 : 0.0,
                "\"" + entity[0] + "\" (from " + entity[1] + ") "
                + (found ? "is" : "is not") + " in the first " + LEAD + " words",
                found ? "" : "open with who you are and what you do");
    }

    static CheckResult meta(Site site) {
        PageFacts f = site.facts();
        List<Object> dated = validNodes(site).stream()
                .filter(n -> n.containsKey("dateModified"))
                .<Object>map(n -> n.get("dateModified")).toList();
        Map<String, Boolean> have = new LinkedHashMap<>();
        have.put("title", f.title.length() >= 15);
        have.put("description",
                f.description != null && !f.description.isEmpty());
        have.put("lang", f.lang != null);
        have.put("canonical", f.canonical != null && !f.canonical.isEmpty());
        have.put("dateModified", !dated.isEmpty());
        List<String> missing = have.entrySet().stream()
```

Kode 3.13 (Lanjutan) *Checks.java: record Site dan CheckResult, antarmuka Check, dan delapan pemeriksaan.*

```
                .filter(e -> !e.getValue()).map(Map.Entry::getKey).toList();
        int found = have.size() - missing.size();
        return new CheckResult("meta", (double) found / have.size(),
                "title " + f.title.length() + " chars; " + found + " of "
                + have.size() + " present"
                + (dated.isEmpty() ? "" : " (dateModified " + dated.get(0) + ")"),
                missing.isEmpty() ? "" : "add " + String.join(", ", missing));
    }

    static final List<String> NAMES = List.of("crawlers", "sitemap", "llms.txt",
            "json-ld", "headings", "text", "answer-1st", "meta");
    static final List<Check> CHECKS = List.of(Checks::crawlers, Checks::sitemap,
            Checks::llms, Checks::jsonld, Checks::headings, Checks::text,
            Checks::answerFirst, Checks::meta);
}
```

3.10.7 Laporan dan baris perintah

Kode 3.14 menyatukan semuanya. `fetch` membungkus satu permintaan HTTP dengan `User-Agent` khusus, `ReadinessLab/1.0 (book exercise)`, dan batas waktu lima detik, serta mengubah setiap hasil menjadi kode status: kode sebenarnya untuk jawaban HTTP, termasuk galat seperti 404, dan 0 bila tidak ada jawaban sama sekali. Di `gather`-lah kesopanan tinggal. Fungsi ini mengambil `robots.txt` **lebih dulu**, lalu bertanya kepada `decide` apakah *crawler kita sendiri*, `ReadinessLab`, boleh mengambil halaman itu. Alat yang mengaudit `robots.txt` orang lain harus menaatinya juga. Baru setelah itu ia mengambil halaman dan `llms.txt`, serta `/sitemap.xml` hanya bila `robots.txt` belum menyebut sitemap: tidak pernah lebih dari empat permintaan, satu per sumber.

Kode 3.14 `readiness.py`: mengambil dengan sopan, menjalankan pemeriksaan, mencetak laporan, dan menghitung skor ajar.

```
# readiness.py - a mini AI-readiness checker for one web page (Chapter 3 lab)
# usage: python readiness.py <url> [--weight name=number ...]
import sys
import urllib.error
import urllib.parse
import urllib.request

from checks import CHECKS, CRAWLERS, NAMES, Site
from page_facts import PageFacts
from robots_rules import RobotsRules

AGENT = "ReadinessLab"                          # our own crawler's name
HEADERS = {"User-Agent": AGENT + "/1.0 (book exercise)"}


def fetch(url):
    """Return (status, body bytes); status 0 means no answer at all."""
    request = urllib.request.Request(url, headers=HEADERS)
    try:
        with urllib.request.urlopen(request, timeout=5) as response:
            return response.status, response.read()
    except urllib.error.HTTPError as e:
        return e.code, b""
    except (urllib.error.URLError, OSError):
        return 0, b""


def gather(url):
    parts = urllib.parse.urlsplit(url)
    base = f"{parts.scheme}://{parts.netloc}"
    path = parts.path or "/"
    status, body = fetch(base + "/robots.txt")
    robots = RobotsRules(body.decode("utf-8", "replace"), status)
    allowed, reason = robots.decide(AGENT, path)
    if not allowed:                             # robots.txt binds us too
        return None, f"{AGENT} may not fetch {path}: {reason}"
    status, body = fetch(url)
    if status != 200:
        return None, f"the page answered {status}"
    facts = PageFacts(body.decode("utf-8", "replace"))
    llms_status, llms_body = fetch(base + "/llms.txt")
    sitemap_status = 0
    if not robots.sitemaps:                     # only when robots.txt names none
        sitemap_status, _ = fetch(base + "/sitemap.xml")
    return Site(url, path, facts, robots, llms_status,
                llms_body.decode("utf-8", "replace"), sitemap_status), ""
```

Kode 3.14 (Lanjutan) *readiness.py: mengambil dengan sopan, menjalankan pemeriksaan, mencetak laporan, dan menghitung skor ajar.*

```
def label(points):
    return "PASS" if points == 1 else ("FAIL" if points == 0 else "PART")


def report(site, weights):
    print("AI-readiness check (a teaching score, not Profitina's)")
    print("URL:", site.url)
    extra = f", sitemap.xml {site.sitemap_status}" if site.sitemap_status else ""
    print(f"Fetched: page 200 ({site.facts.html_bytes} bytes), robots.txt "
          f"{site.robots.status}, llms.txt {site.llms_status}{extra}")
    print("Crawler access to / according to robots.txt:")
    for crawler in CRAWLERS:
        ok, reason = site.robots.decide(crawler, "/")
        print(f"  {crawler:<16} {'allowed' if ok else 'blocked':<8} {reason}")
    total = weight_sum = 0.0
    for i, check in enumerate(CHECKS, 1):
        r = check(site)
        w = weights.get(r.name, 1.0)
        total += w * r.points
        weight_sum += w
        print(f"{i}. [{label(r.points)}] {r.name:<11} points {r.points:.2f},"
              f" weight {w:g}")
        print(f"{'':10}{r.evidence}")
        if r.advice:
            print(f"{'':10}fix: {r.advice}")
    score = int(100 * total / weight_sum + 0.5) if weight_sum else 0
    shown = ", ".join(f"{k}={v:g}" for k, v in weights.items())
    shown = shown + ", others 1" if shown else "all 1"
    print(f"Teaching score: {score} / 100 (weights: {shown})")
    return score


def parse_weights(args):
    """Turn ["--weight", "text=2", ...] into {"text": 2.0, ...}."""
    weights = {}
    for i in range(0, len(args), 2):
        if args[i] != "--weight" or i + 1 == len(args):
            raise ValueError(f"unexpected argument {args[i]}")
        name, _, value = args[i + 1].partition("=")
        if name not in NAMES:
            raise ValueError(f"unknown check {name!r}")
        weights[name] = float(value)
        if weights[name] < 0:
            raise ValueError("a weight cannot be negative")
```

Kode 3.14 (Lanjutan) *readiness.py: mengambil dengan sopan, menjalankan pemeriksaan, mencetak laporan, dan menghitung skor ajar.*

```
    return weights


def main(argv):
    if not argv or argv[0].startswith("-"):
        print("usage: python readiness.py <url> [--weight name=number ...]")
        return 1
    try:
        weights = parse_weights(argv[1:])
    except ValueError as e:
        print("Bad option:", e)
        print("Checks:", ", ".join(NAMES))
        return 1
    site, problem = gather(argv[0])
    if site is None:
        print("Stopped:", problem)
        return 2
    report(site, weights)
    return 0


if __name__ == "__main__":
    sys.exit(main(sys.argv[1:]))
```

`report` mencetak tabel *crawler*, lalu setiap pemeriksaan dengan labelnya, `PASS` untuk 1 poin, `FAIL` untuk 0, dan `PART` untuk nilai di antaranya, poin dan bobotnya, bukti, dan sarannya, dan akhirnya skor dari Persamaan (3.2). Baris `int(100 * total / weight_sum + 0.5)` adalah pembulatan ke bawah dari "nilai ditambah setengah" untuk nilai tak negatif. Baris itu dipakai alih-alih `round` karena `round` milik Python membulatkan nilai setengah ke bilangan *genap* terdekat, `round(12.5)` menghasilkan 12, sedangkan `Math.round(12.5)` milik Java menghasilkan 13; menuliskan pembulatannya secara eksplisit membuat kedua bahasa sepakat. Karena alasan yang sama, versi Java memformat angka dengan `Locale.ROOT`: dalam *locale* Indonesia, `String.format("%.2f", 0.5)` akan mencetak `0,50`, dengan koma desimal.

`parse_weights` membaca opsi berbentuk `--weight nama=angka`, menolak nama pemeriksaan yang tidak dikenal dan bobot negatif, dan melaporkan kesalahan dalam satu baris, bukan dalam jejak tumpukan (*stack trace*). Versi Java, **Kode 3.15**, memakai `java.net.http.HttpClient` [22], diatur ke HTTP/1.1 dan mengikuti pengalihan, dan membaca setiap badan respons sebagai byte agar hitungan byte pada Persamaan (3.4) tepat.

Kode 3.15 `Readiness.java`: pengambilan, pelaporan, dan penilaian yang sama dengan `java.net.http.HttpClient`.

```java
// Readiness.java - a mini AI-readiness checker for one web page (Chapter 3 lab)
// usage: java Readiness <url> [--weight name=number ...]
import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

public class Readiness {
    static final String AGENT = "ReadinessLab";     // our own crawler's name
    static final HttpClient CLIENT = HttpClient.newBuilder()
            .version(HttpClient.Version.HTTP_1_1)
            .followRedirects(HttpClient.Redirect.NORMAL)
            .connectTimeout(Duration.ofSeconds(5)).build();

    record Fetched(int status, byte[] body) {
        String text() {
            return new String(body, StandardCharsets.UTF_8);
        }
    }

    /** Status 0 means no answer at all. */
    static Fetched fetch(String url) {
        HttpRequest request = HttpRequest.newBuilder(URI.create(url))
                .header("User-Agent", AGENT + "/1.0 (book exercise)")
                .timeout(Duration.ofSeconds(5)).build();
        try {
            HttpResponse<byte[]> r =
                    CLIENT.send(request, HttpResponse.BodyHandlers.ofByteArray());
            return new Fetched(r.statusCode(),
                    r.statusCode() == 200 ? r.body() : new byte[0]);
        } catch (IOException | InterruptedException
                | IllegalArgumentException e) {
            return new Fetched(0, new byte[0]);
        }
    }
```

Kode 3.15 (Lanjutan) *Readiness.java: pengambilan, pelaporan, dan penilaian yang sama dengan java.net.http.HttpClient.*

```
    /** Returns the Site, or null after printing why the check stopped. */
    static Checks.Site gather(String url) {
        URI uri = URI.create(url);
        String base = uri.getScheme() + "://" + uri.getRawAuthority();
        String path = uri.getRawPath() == null || uri.getRawPath().isEmpty()
                ? "/" : uri.getRawPath();
        Fetched robotsFile = fetch(base + "/robots.txt");
        RobotsRules robots =
                new RobotsRules(robotsFile.text(), robotsFile.status());
        RobotsRules.Decision d = robots.decide(AGENT, path);
        if (!d.allowed()) {                          // robots.txt binds us too
            System.out.println("Stopped: " + AGENT + " may not fetch " + path
                    + ": " + d.reason());
            return null;
        }
        Fetched page = fetch(url);
        if (page.status() != 200) {
            System.out.println("Stopped: the page answered " + page.status());
            return null;
        }
        Fetched llms = fetch(base + "/llms.txt");
        int sitemapStatus = 0;
        if (robots.sitemaps.isEmpty()) {    // only when robots.txt names none
            sitemapStatus = fetch(base + "/sitemap.xml").status();
        }
        return new Checks.Site(url, path, new PageFacts(page.text()), robots,
                llms.status(), llms.text(), sitemapStatus);
    }

    static String label(double points) {
        return points == 1 ? "PASS" : (points == 0 ? "FAIL" : "PART");
    }

    static String number(double w) {                       // like Python's {w:g}
        return w == Math.rint(w) ? Long.toString((long) w) : Double.toString(w);
    }
```

Kode 3.15 (Lanjutan) *Readiness.java: pengambilan, pelaporan, dan penilaian yang sama dengan java.net.http.HttpClient.*

```
    static int report(Checks.Site site, Map<String, Double> weights) {
        System.out.println(
                "AI-readiness check (a teaching score, not Profitina's)");
        System.out.println("URL: " + site.url());
        String extra = site.sitemapStatus() != 0
                ? ", sitemap.xml " + site.sitemapStatus() : "";
        System.out.println("Fetched: page 200 (" + site.facts().htmlBytes
                + " bytes), robots.txt " + site.robots().status + ", llms.txt "
                + site.llmsStatus() + extra);
        System.out.println("Crawler access to / according to robots.txt:");
        for (String crawler : Checks.CRAWLERS) {
            RobotsRules.Decision d = site.robots().decide(crawler, "/");
            System.out.printf("  %-16s %-8s %s%n", crawler,
                    d.allowed() ? "allowed" : "blocked", d.reason());
        }
        double total = 0;
        double weightSum = 0;
        int i = 0;
        for (Checks.Check check : Checks.CHECKS) {
            Checks.CheckResult r = check.run(site);
            double w = weights.getOrDefault(r.name(), 1.0);
            total += w * r.points();
            weightSum += w;
            System.out.printf(Locale.ROOT,
                    "%d. [%s] %-11s points %.2f, weight %s%n", ++i,
                    label(r.points()), r.name(), r.points(), number(w));
            System.out.println(" ".repeat(10) + r.evidence());
            if (!r.advice().isEmpty()) {
                System.out.println(" ".repeat(10) + "fix: " + r.advice());
            }
        }
        int score = weightSum == 0 ? 0 : (int) (100 * total / weightSum + 0.5);
        StringBuilder shown = new StringBuilder();
        for (Map.Entry<String, Double> e : weights.entrySet()) {
            shown.append(shown.length() == 0 ? "" : ", ")
                 .append(e.getKey()).append('=').append(number(e.getValue()));
        }
        System.out.println("Teaching score: " + score + " / 100 (weights: "
                + (shown.length() == 0 ? "all 1" : shown + ", others 1") + ")");
        return score;
    }
```

Kode 3.15 (Lanjutan) *Readiness.java: pengambilan, pelaporan, dan penilaian yang sama dengan java.net.http.HttpClient.*

```
    /** Turn ["--weight", "text=2", ...] into {"text": 2.0, ...}. */
    static Map<String, Double> parseWeights(List<String> args) {
        Map<String, Double> weights = new LinkedHashMap<>();
        for (int i = 0; i < args.size(); i += 2) {
            if (!args.get(i).equals("--weight") || i + 1 == args.size()) {
                throw new IllegalArgumentException("unexpected argument "
                        + args.get(i));
            }
            String[] kv = args.get(i + 1).split("=", 2);
            if (!Checks.NAMES.contains(kv[0])) {
                throw new IllegalArgumentException(
                        "unknown check '" + kv[0] + "'");
            }
            double w = Double.parseDouble(kv.length == 2 ? kv[1] : "");
            if (w < 0) {
                throw new IllegalArgumentException("a weight cannot be negative");
            }
            weights.put(kv[0], w);
        }
        return weights;
    }

    public static void main(String[] args) {
        if (args.length == 0 || args[0].startsWith("-")) {
            System.out.println(
                    "usage: java Readiness <url> [--weight name=number ...]");
            System.exit(1);
        }
        Map<String, Double> weights;
        try {
            weights = parseWeights(List.of(args).subList(1, args.length));
        } catch (IllegalArgumentException e) {        // includes a bad number
            System.out.println("Bad option: " + e.getMessage());
            System.out.println("Checks: " + String.join(", ", Checks.NAMES));
            System.exit(1);
            return;
        }
        Checks.Site site = gather(args[0]);
        if (site == null) {
            System.exit(2);
        }
        report(site, weights);
    }
}
```

3.10.8 Menjalankan lab: sebelum dan sesudah

Cara tercepat untuk melihat semuanya adalah program demonstrasi, yang menyalakan dua peladen di latar belakang, satu untuk tiap situs contoh, memeriksa keduanya, dan mematikan kedua peladen sebelum selesai, bahkan bila ada yang gagal di tengah jalan, karena pematiannya terjadi di blok `finally`. **Kode 3.16** adalah versi Python-nya.

Kode 3.16 `demo.py`: menyajikan kedua situs contoh di porta 7030 dan 7031, memeriksanya, dan selalu mematikan peladennya.

```
# demo.py - serve both sample sites, check them, and stop the servers
import threading

from readiness import gather, report
from serve_site import make_server

SITES = [("../sites/before", 7030), ("../sites/after", 7031)]


def main():
    servers = [make_server(folder, port, verbose=False) for folder, port in SITES]
    for s in servers:
        threading.Thread(target=s.serve_forever, daemon=True).start()
    try:
        for _, port in SITES:
            site, problem = gather(f"http://127.0.0.1:{port}/")
            if site is None:
                print("Stopped:", problem)
            else:
                report(site, {})
            print()
    finally:
        for s in servers:                  # always stop what we started
            s.shutdown()
            s.server_close()
    print("Both servers stopped.")


if __name__ == "__main__":
    main()
```

Kode 3.17 adalah versi Java-nya. `HttpServer.stop(0)` menghentikan peladen Java seketika.

Kode 3.17 `Demo.java`: demonstrasi yang sama dalam Java.

```java
// Demo.java - serve both sample sites, check them, and stop the servers
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.util.List;
import java.util.Map;

public class Demo {
    public static void main(String[] args) throws IOException {
        List<String> folders = List.of("../sites/before", "../sites/after");
        List<Integer> ports = List.of(7030, 7031);
        HttpServer[] servers = new HttpServer[2];
        for (int i = 0; i < 2; i++) {
            servers[i] =
                    ServeSite.makeServer(folders.get(i), ports.get(i), false);
            servers[i].start();
        }
        try {
            for (int port : ports) {
                String url = "http://127.0.0.1:" + port + "/";
                Checks.Site site = Readiness.gather(url);
                if (site != null) {
                    Readiness.report(site, Map.of());
                }
                System.out.println();
            }
        } finally {
            for (HttpServer s : servers) {        // always stop what we started
                s.stop(0);
            }
        }
        System.out.println("Both servers stopped.");
    }
}
```

Buka terminal di folder `python` bab ini dan jalankan `python demo.py` (pada sebagian sistem `python3 demo.py`); atau, di folder `java`, kompilasi sekali dengan `javac *.java` lalu jalankan `java Demo`. Berkas `README.md` bab ini memuat perintah persisnya untuk Windows, macOS, dan Linux, di terminal maupun di VS Code. Kedua versi mencetak persis keluaran berikut:

```
AI-readiness check (a teaching score, not Profitina's)
URL: http://127.0.0.1:7030/
Fetched: page 200 (1860 bytes), robots.txt 200, llms.txt 404, sitemap.xml 404
Crawler access to / according to robots.txt:
  GPTBot           blocked  Disallow: / in group "GPTBot"
  OAI-SearchBot    allowed  Allow: / in group "*"
  ClaudeBot        blocked  Disallow: / in group "ClaudeBot"
  PerplexityBot    allowed  Allow: / in group "*"
  Google-Extended  allowed  Allow: / in group "*"
  Bingbot          allowed  Allow: / in group "*"
1. [PART] crawlers    points 0.67, weight 1
          4 of 6 crawlers may fetch /
          fix: unblock GPTBot, ClaudeBot
2. [FAIL] sitemap     points 0.00, weight 1
          no Sitemap line, /sitemap.xml 404
          fix: publish /sitemap.xml and name it in robots.txt
3. [FAIL] llms.txt    points 0.00, weight 1
          /llms.txt 404
          fix: optional: publish /llms.txt (few AI bots read it)
4. [FAIL] json-ld     points 0.00, weight 1
          0 valid, 1 invalid: no types
          fix: repair the invalid JSON
5. [FAIL] headings    points 0.00, weight 1
          2 h1, 5 headings, level jumps: 1
          fix: use exactly one h1; h1 is followed by h4
6. [FAIL] text        points 0.00, weight 1
          40 words, text/HTML ratio 0.13
          fix: aim for 120+ words and ratio 0.20+
7. [FAIL] answer-1st  points 0.00, weight 1
          "Home" (from <title>) is not in the first 60 words
          fix: open with who you are and what you do
8. [PART] meta        points 0.40, weight 1
          title 4 chars; 2 of 5 present
          fix: add title, canonical, dateModified
Teaching score: 13 / 100 (weights: all 1)

AI-readiness check (a teaching score, not Profitina's)
URL: http://127.0.0.1:7031/
Fetched: page 200 (3258 bytes), robots.txt 200, llms.txt 200
Crawler access to / according to robots.txt:
  GPTBot           allowed  Allow: / in group "*"
  OAI-SearchBot    allowed  Allow: / in group "*"
  ClaudeBot        allowed  Allow: / in group "*"
  PerplexityBot    allowed  Allow: / in group "*"
  Google-Extended  allowed  Allow: / in group "*"
  Bingbot          allowed  Allow: / in group "*"
1. [PASS] crawlers    points 1.00, weight 1
          6 of 6 crawlers may fetch /
2. [PASS] sitemap     points 1.00, weight 1
          robots.txt names https://laundry.example/sitemap.xml
3. [PASS] llms.txt    points 1.00, weight 1
          present, title # Profitina Laundry
4. [PASS] json-ld     points 1.00, weight 1
          1 valid, 0 invalid: DryCleaningOrLaundry, WebPage, FAQPage
5. [PASS] headings    points 1.00, weight 1
          1 h1, 10 headings, level jumps: 0
6. [PASS] text        points 1.00, weight 1
          175 words, text/HTML ratio 0.32
7. [PASS] answer-1st  points 1.00, weight 1
          "Profitina Laundry" (from JSON-LD) is in the first 60 words
8. [PASS] meta        points 1.00, weight 1
          title 62 chars; 5 of 5 present (dateModified 2026-09-20)
Teaching score: 100 / 100 (weights: all 1)

Both servers stopped.
```

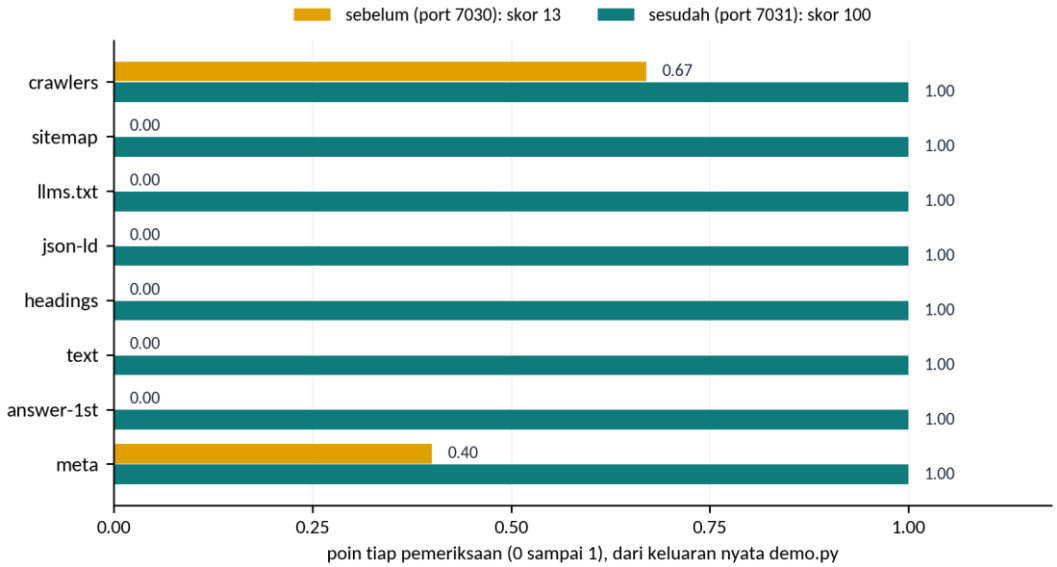
Bacalah laporan pertama sebagaimana Sari membacanya. Akses *crawler* bersifat sebagian: empat dari enam *crawler* boleh membaca halaman, dan tabelnya menyebut aturan yang menutup pintu bagi dua sisanya. Tidak ada sitemap dan tidak ada `llms.txt`. JSON-LD-nya ada tetapi tidak sah, sehingga tidak ada tipe yang ditemukan, dan sarannya adalah memperbaikinya, bukan menambahkannya. Pemeriksaan judul menemukan dua judul `h1` dan satu lompatan, dan sarannya menyebut lompatan itu, `h1` diikuti `h4`. Halaman itu memiliki 40 kata terlihat dan rasio teks terhadap HTML 0,13. Pemeriksaan *answer-first* tidak mendapatkan JSON-LD yang sah untuk diambil namanya, sehingga memakai *title*, "Home", yang tentu saja tidak muncul di kata-kata pembuka. Dua dari lima hal dasar, deskripsi dan atribut `lang`, sudah ada.

Menghitung skor laporan pertama dengan tangan menggunakan Persamaan (3.2) adalah latihan yang baik untuk membaca alat buatan sendiri. Semua bobot bernilai 1, sehingga penyebutnya 8. Poinnya adalah $4/6$ untuk *crawler*, menurut Persamaan (3.5), dan $2/5 = 0.4$ untuk tag meta, menurut Persamaan (3.7); semua yang lain 0. Jumlahnya

$$\frac{4}{6} + 0.4 = 0.6667 + 0.4 = 1.0667,$$

dan $100 \cdot 1.0667/8 = 13.33$; ditambah setengah menjadi 13,83, yang pembulatan ke bawahnya adalah **13**, skor yang tercetak.

Laporan kedua lolos semua pemeriksaan, dan **Gambar 3.12** menampilkan keduanya berdampingan, digambar dari keluaran di atas.



Gambar 3.12 Poin tiap pemeriksaan untuk kedua situs contoh, diambil dari keluaran nyata `demo.py`: sebelum perbaikan (skor 13) dan sesudahnya (skor 100).

Skor 100 adalah saat yang tepat untuk mengulang peringatan Subbab 3.10.1. Skor itu berarti halaman "after" memenuhi delapan aturan yang dipilih untuk pengajaran. Skor itu tidak berarti mesin jawaban mana pun akan mengutip Profitina Laundry, dan tidak berarti tidak ada lagi yang dapat diperbaiki: halaman itu tidak memiliki foto dengan teks `alt` yang deskriptif, tidak mencantumkan jam buka di JSON-LD, dan tidak memuat ulasan. Daftar periksa yang mencapai nilai maksimum telah mencapai batas *pengetahuannya sendiri*, bukan batas pokok bahasannya.

3.10.9 Memilih bobot sendiri

Skor ajar menjadi alat berpikir ketika Anda mengubah bobotnya. Misalkan Anda memutuskan bahwa akses *crawler* tiga kali lebih penting daripada apa pun, dan bahwa `llms.txt`, yang hampir tidak pernah diminta sistem AI [23], sama sekali tidak perlu dihitung. Nyalakan peladen untuk situs "before" di satu terminal:

```
python serve_site.py ../sites/before 7032
```

lalu jalankan pemeriksa dengan dua bobot di terminal kedua:

```
python readiness.py http://127.0.0.1:7032/ --weight llms.txt=0 --weight crawlers=3
```

Terminal peladen memperlihatkan setiap permintaan yang dibuat pemeriksa, dan jumlahnya tepat empat, satu per sumber, dengan `robots.txt` di urutan pertama:

```
Serving ../sites/before at http://127.0.0.1:7032/ (Ctrl+C stops)
  GET /robots.txt -> 200
  GET / -> 200
  GET /llms.txt -> 404
  GET /sitemap.xml -> 404
```

Pemeriksa mencetak delapan pemeriksaan yang sama dengan bobot baru, dan skor baru:

```
AI-readiness check (a teaching score, not Profitina's)
URL: http://127.0.0.1:7032/
Fetched: page 200 (1860 bytes), robots.txt 200, llms.txt 404, sitemap.xml 404
Crawler access to / according to robots.txt:
  GPTBot           blocked  Disallow: / in group "GPTBot"
  OAI-SearchBot    allowed  Allow: / in group "*"
  ClaudeBot        blocked  Disallow: / in group "ClaudeBot"
  PerplexityBot    allowed  Allow: / in group "*"
  Google-Extended  allowed  Allow: / in group "*"
  Bingbot          allowed  Allow: / in group "*"
1. [PART] crawlers    points 0.67, weight 3
          4 of 6 crawlers may fetch /
          fix: unblock GPTBot, ClaudeBot
2. [FAIL] sitemap     points 0.00, weight 1
          no Sitemap line, /sitemap.xml 404
          fix: publish /sitemap.xml and name it in robots.txt
3. [FAIL] llms.txt    points 0.00, weight 0
          /llms.txt 404
          fix: optional: publish /llms.txt (few AI bots read it)
4. [FAIL] json-ld     points 0.00, weight 1
          0 valid, 1 invalid: no types
          fix: repair the invalid JSON
5. [FAIL] headings    points 0.00, weight 1
          2 h1, 5 headings, level jumps: 1
          fix: use exactly one h1; h1 is followed by h4
6. [FAIL] text        points 0.00, weight 1
          40 words, text/HTML ratio 0.13
          fix: aim for 120+ words and ratio 0.20+
7. [FAIL] answer-1st  points 0.00, weight 1
          "Home" (from <title>) is not in the first 60 words
          fix: open with who you are and what you do
8. [PART] meta        points 0.40, weight 1
          title 4 chars; 2 of 5 present
          fix: add title, canonical, dateModified
Teaching score: 27 / 100 (weights: llms.txt=0, crawlers=3, others 1)
```

Menurut Persamaan (3.2), penyebutnya kini $3+1+0+1+1+1+1+1=9$, dan pembilangnya $3\cdot\frac{4}{6}+1\cdot 0.4=2+0.4=2.4$. Jadi $100\cdot 2.4/9=26.67$, ditambah setengah menjadi 27,17, dan skornya **27**. Halaman yang sama mendapat skor 13 atau 27 bergantung pada apa yang Anda putuskan penting, dan justru itulah intinya: skor adalah *pilihan* tentang apa yang dihitung

dan seberapa besar, yang diterapkan secara konsisten. Ketika membaca skor kesiapan mana pun, termasuk skor komersial, pertanyaan pertama yang harus diajukan adalah apa yang dihitungnya dan bagaimana, dan pertanyaan kedua adalah apakah skor itu diterapkan dengan cara yang sama setiap kali, sehingga selisih dari waktu ke waktu bermakna. Tekan Ctrl+C di terminal peladen untuk menghentikannya. Perintah Java-nya sama, dengan `java ServeSite` dan `java Readiness`, dan mencetak baris-baris yang sama.

3.10.10 Ketika pemeriksa berhenti

Dua keadaan membuat pemeriksa berhenti alih-alih melapor, dan keduanya disengaja. Bila tidak ada apa pun yang mendengarkan di alamat itu, permintaan `robots.txt` tidak mendapat jawaban, status 0, dan menurut baris 2 **Pseudokode 3.1** pemeriksa harus menganggap dirinya tidak boleh masuk:

```
python readiness.py http://127.0.0.1:7039/
```

```
Stopped: ReadinessLab may not fetch /: robots.txt unreachable: assume disallow
exit 2
```

Status keluar 2 memungkinkan sebuah skrip membedakan “berhenti” dari “melapor” (status 0) dan “salah pakai” (status 1). Nama pemeriksaan yang salah eja adalah contoh keadaan kedua:

```
python readiness.py http://127.0.0.1:7031/ --weight speed=2
```

```
Bad option: unknown check 'speed'
Checks: crawlers, sitemap, llms.txt, json-ld, headings, text, answer-1st, meta
exit 1
```

Kesalahan pada opsi dilaporkan sebelum permintaan apa pun dibuat, dengan kata-kata, beserta daftar nama yang sah. Soal 3.4 menambahkan keadaan ketiga, yaitu situs yang `robots.txt`-nya menutup pintu khusus bagi `ReadinessLab`.

3.10.11 Apa yang tidak dapat dilihat pemeriksa

Berterus terang tentang batas-batas sebuah alat adalah bagian dari membangunnya. Pemeriksa kita membaca HTML yang dikirim peladen; ia tidak menjalankan JavaScript, sehingga halaman yang menyusun teksnya di peramban tampak kosong baginya, sebagaimana bagi sebagian *crawler*. Ia memeriksa satu halaman, bukan satu situs. Ia memverifikasi bahwa JSON-LD adalah JSON yang sah dengan nama dan alamat, bukan bahwa faktanya benar atau selaras dengan halaman yang terlihat. Ia tidak tahu apa-apa tentang tautan dari situs lain, tentang ulasan, ataupun tentang seberapa sering usaha itu disebut di tempat lain. Dan, seperti Profitina dan dengan penalaran yang sama (Subbab 3.2.3), ia tidak pernah bertanya apa pun kepada mesin jawaban. Setiap batas ini dapat menjadi awal sebuah pengembangan; beberapa di antaranya muncul dalam soal latihan.

3.11 Profitina dalam Praktik

Lab G1 adalah model ajar, dan patut dinyatakan dengan tepat bagaimana hubungannya dengan produk yang dimodelkannya.

Bentuk yang sama, isi yang berbeda. AI Visibility Engine milik Profitina, seperti lab ini, mengubah fakta yang teramati menjadi sinyal, sinyal menjadi poin, poin menjadi estimasi, dan setiap celah menjadi kalimat beserta perbaikannya. Seperti lab ini pula, ia membaca `robots.txt` sebelum yang lain, karena akses didahulukan. Isinya adalah miliknya sendiri: sinyal mana,

bagaimana diberi bobot, bagaimana digabungkan dan dikalibrasi. Delapan pemeriksaan lab dan bobotnya yang setara dipilih untuk dibaca, bukan untuk meramal.

Penjelasan sebagai fitur produk. `CheckResult` dalam lab memiliki ruas `evidence` dan `advice` karena alasan yang sama dengan halaman hasil produk yang memakai sebagian besar ruangnya untuk daftar perbaikan (Subbab 3.4): estimasi yang tidak dapat menjelaskan dirinya tidak menolong siapa pun untuk menjadi lebih baik.

Kegagalan yang jujur. Lab berhenti dengan pesan yang jelas ketika `robots.txt` tidak dapat dibaca, alih-alih menebak, dan menolak opsi yang salah eja alih-alih mengabaikannya. Produk bersikap dengan semangat yang sama di setiap tingkat: detektor kategori berkata "belum yakin" (Subbab 3.3.1); Mesin Konten mengatakan bila tidak ada model yang tersedia alih-alih menyajikan templat sebagai keluaran AI (Subbab 3.5.2); Laporan Visibilitas Peneliti mengembalikan "tidak ditemukan" untuk nama karangan, tidak pernah angka yang tampak masuk akal (Subbab 3.7.1); dan sebuah rilis baru dinyatakan berhasil setelah versi barunya menjawab pemeriksaan kesehatan (Bab 9).

Pelajaran *cache*, sekali lagi. `robots.txt` kelolaan panel pada situs "before" bukan contoh karangan. Berkas sejenis pernah memblokir `GPTBot` dan `ClaudeBot` di profitina.com sendiri, disajikan oleh jaringan pengiriman konten di depan berkas asal yang benar [2]. Pemeriksa yang membaca alamat *live*, seperti yang dilakukan `gather`, menangkap masalah itu; pemeriksa yang membaca berkas dalam kode sumber situs tidak. Bab 10 membangun alat audit yang melihat ke balik *cache*.

Siklus di sekitar lab. Lab G6 di Bab 2 membangkitkan perbaikan untuk langkah *Optimasi*; lab G1 bab ini mengukurnya pada langkah *Pindai*; lab G4 di Bab 9 menjadwalkan pindaian ulang untuk langkah *Pantau*. Bersama-sama, ketiga lab membentuk miniatur siklus pada **Gambar 3.1**, dibangun secara terbuka.

3.12 Kesalahpahaman yang Umum

"Profitina bertanya kepada ChatGPT tentang usaha saya." Tidak. Mesin jawaban adalah sasaran yang kesiapannya untuk mengutip sebuah usaha diperkirakan dari sinyal yang teramati; Profitina tidak mengkuerinya dan tidak bergantung padanya, demi kemandirian, privasi, dan prinsip (Subbab 3.2.3).

"Skor 100 berarti saya akan dikutip." Estimasi kesiapan mengukur seberapa baik sebuah situs membuat dirinya dapat dibaca, dipahami, dan dikutip mesin. Apakah sebuah mesin mengutipnya untuk pertanyaan tertentu juga bergantung pada pertanyaannya, sumber-sumber pesaing, dan mesin itu sendiri. Syarat layanannya menyatakan bahwa hasil adalah "methodological estimates, not guarantees" (estimasi metodologis, bukan jaminan) [3]. Halaman "after" lab G1 meraih 100 pada delapan aturan ajar dan masih punya banyak hal yang dapat diperbaiki.

"GEO cukup diperbaiki sekali." Situs, *crawler*, dan mesin jawaban semuanya berubah. Pembaruan tema atau panel hosting dapat membatalkan perbaikan diam-diam, seperti diperlihatkan kisah `robots.txt` profitina.com. Karena itulah metodenya berupa siklus dan tab ketiganya adalah *Pantau*.

"Bila `robots.txt` diakhiri `Allow: /`, semua *crawler* diizinkan." *Crawler* yang memiliki grup sendiri mengabaikan grup `*` sepenuhnya. Di situs "before", `GPTBot` dan `ClaudeBot` diblokir oleh grup mereka sendiri walaupun berkasnya diakhiri `Allow: /`.

"`robots.txt` yang tidak ada memblokir *crawler*." Justru sebaliknya: menurut RFC 9309, jawaban 4xx berarti *crawler* boleh mengambil apa saja. Galat peladen atau tidak adanya jawabanlah yang berarti "anggap dilarang" [7].

"JSON-LD yang tidak sah lebih baik daripada tidak ada." Bagi pengurai yang ketat, ia *memang* tidak ada: satu koma di akhir membuat seluruh blok tidak terbaca, seperti diperlihatkan halaman "before". Validasikan data terstruktur setiap kali berubah.

"`llms.txt` membuat AI membaca situs saya." Menerbitkannya memang sudah lazim, 9,3% dari 1.000 domain teratas memilikinya pada September 2026 [24], tetapi pada Mei 2026, 97% berkas dalam satu sampel besar sama sekali tidak menerima permintaan [23]. Berkas itu kebersihan yang murah, dan karena itu lab G1 menghitungnya sebagai satu dari delapan pemeriksaan dan membolehkan Anda memberinya bobot nol.

"Laporan peneliti memeringkat dosen." Baik Laporan Visibilitas Peneliti maupun Visibilitas Organisasi bukan papan peringkat orang. Yang pertama menyebut celah yang dapat ditutup dalam rekaman bibliografis satu orang; yang kedua melaporkan celah agregat institusi, seperti 466 ejaan satu afiliasi [2].

"Pengguna Free tidak mendapat hak atas datanya." Ekspor, koreksi, dan penghapusan permanen gratis di semua paket, selamanya, dan hanya pengguna yang dapat memulai penghapusan [2].

"Bobot yang lebih tinggi membuat sebuah pemeriksaan lebih penting dalam kenyataan." Bobot menyatakan penilaian *Anda* tentang kepentingan dalam skor *Anda*. Mengubah bobot mengubah angkanya, seperti diperlihatkan Subbab 3.10.9 dengan 13 dan 27 untuk halaman yang sama, dan tidak berkata apa-apa tentang perilaku mesin mana pun.

3.13 Ringkasan Bab

- Metode Profitina adalah **siklus empat kata kerja**, ukur, jelaskan, perbaiki, pantau (**Gambar 3.1**), yang tampil kepada pengguna sebagai tiga tab: **Pindai · Optimasi · Pantau** (POP) dan **Scan · Optimize · Monitor** (SOM) (**Tabel 3.1**). Bentuknya siklus karena situs, *crawler*, dan mesin jawaban berubah.
- Pindaian berangkat dari nama dan domain; **isi otomatis** mengusulkan kategori, lokasi, dan pertanyaan pelanggan, dan berkata **"belum yakin"** bila ragu. Buktinya berasal dari tiga keluarga sinyal yang teramati, **akses, struktur, dan keterekstrakan**, yang masing-masing berakar pada standar terbuka (**Tabel 3.2**). Profitina **tidak pernah mengkueri** mesin jawaban.
- Hasilnya adalah **estimasi kesiapan 0–100**, bukan jaminan, dengan metode yang ditampilkan dan **daftar perbaikan berprioritas** dalam bahasa sehari-hari; setiap baris memiliki bukti dan langkah berikutnya, dan umpan balik tentang pindaian selalu dapat dikirim.
- **Optimasi** menyediakan **Mesin Konten** (draf jawab-dulu, entitas, FAQ, JSON-LD) di balik **gerbang mutu** yang hanya menyajikan draf AI bila mengalahkan templat (5 dari 6 kasus tolok ukur), dan mengatakannya bila model tidak tersedia; **Pendamping Teknis** menangani JSON-LD, `robots.txt`, dan pengiriman sitemap ke Bing. Pemilik usaha yang menerbitkan.
- **Pantau** menyimpan riwayat pribadi dan **tren mingguan**, memindai ulang secara terjadwal (Free mingguan, Pro harian), dan memberi peringatan atas perubahan yang berarti; aturan peringatan sederhana membandingkan perubahan Δ_t pada Persamaan (3.1) dengan ambang δ yang lebih besar daripada derau biasa.

- **Laporan Visibilitas Peneliti** (OpenAlex: DOI, ORCID, afiliasi, akses terbuka, sitasi) menyebut celah yang dapat ditutup dan memiliki **kontrol negatif** (8 dari 8 nama fiktif bernilai 0,0); **Visibilitas Organisasi** melaporkan celah agregat (466 ejaan dalam 500 karya; 244 dari 607 peneliti tanpa ORCID di ITS) dan bukan papan peringkat orang.
- **Satu Teks, Banyak Platform** menyesuaikan satu teks untuk satu dari enam platform dalam batas karakternya, dengan antrean yang terlihat, dan berbagi ke X melalui akun pengguna sendiri, tanpa API X.
- **Perjalanan pengguna** terdiri atas sembilan langkah (**Gambar 3.2**): datang, mencoba, mendaftar (email dan kata sandi, persetujuan tanpa centang awal, kode email), pindai, optimasi, pantau, berpartisipasi, meningkatkan paket (Pro, Rp50.000 untuk 30 hari melalui QRIS atau PROFIT; **Tabel 3.3**), dan menggunakan hak melalui panel **Akun & Data Pribadi**, gratis selamanya.
- **Lab G1** (**Gambar 3.11**, **Tabel 3.4**) adalah pemeriksa yang sopan dalam Python dan Java yang membaca `robots.txt` lebih dulu (**Pseudokode 3.1**, Persamaan (3.3)), mengumpulkan fakta halaman dalam satu lintasan *streaming* (Persamaan (3.4)), menjalankan delapan pemeriksaan yang dijelaskan (Persamaan (3.5), Persamaan (3.6), Persamaan (3.7)), dan menghitung **skor ajar** yang transparan (Persamaan (3.2)) dengan bobot pilihan Anda. Pada situs contoh, skornya 13 sebelum perbaikan dan 100 sesudahnya (**Gambar 3.12**); skor ini milik kita sendiri, bukan milik Profitina.

3.14 Soal Latihan

Soal 3.1. Untuk setiap fitur berikut, sebutkan kata kerja siklus yang dilayaninya (ukur, jelaskan, perbaiki, atau pantau) dan tab tempat pengguna menjumpainya: (a) ekspor CSV riwayat sebuah merek; (b) permintaan kategori “belum yakin”; (c) daftar *crawler* di Pendamping Teknis; (d) pengurutan perbaikan menurut prioritas; (e) peringatan email setelah pindai ulang; (f) label kemajuan “kueri 3/8”.

Soal 3.2. Skor mingguan sebuah merek adalah 52, 55, 54, 71, 70, 69, 48, 50. Hitung setiap Δ_t dengan Persamaan (3.1). Minggu mana yang memicu peringatan untuk $\delta = 10$, dan mana untuk $\delta = 3$? Ambang mana yang akan Anda pilih, dan mengapa?

Soal 3.3. Perhatikan `robots.txt` berikut:

```
User-agent: *
Disallow: /
Allow: /services

User-agent: GPTBot
User-agent: ClaudeBot
Disallow: /status

User-agent: gptbot
Allow: /status/public
```

Putuskan dengan tangan, menggunakan **Pseudokode 3.1**, apakah masing-masing berikut diizinkan: `GPTBot` mengambil `/`, `/status/42`, dan `/status/public`; `ClaudeBot` mengambil `/status/public`; `PerplexityBot` mengambil `/services` dan `/outlets`. Kemudian tulis program, dalam Python dan Java, yang mencetak keputusan-keputusan itu dengan `RobotsRules`; dalam Python, cetak juga keputusan `urllib.robotparser` milik Python di sampingnya dan jelaskan setiap perbedaan.

Soal 3.4. Folder `sites/closed` berisi sebuah halaman dan `robots.txt` berikut:

```
User-agent: ReadinessLab
Disallow: /

User-agent: *
Allow: /
```

Sajikan folder itu di porta 7034 dan jalankan pemeriksa terhadapnya. Ramalkan dulu keluarannya. Apakah perilaku pemeriksa itu benar? Apa yang boleh dilakukan *crawler* santun bernama `GPTBot` di situs ini?

Soal 3.5. Dengan Persamaan (3.2) dan poin halaman "before" pada Subbab 3.10.8, hitung skor ajar dengan bobot `json-ld=2` dan `headings=2`, sedangkan yang lain 1. Kemudian pastikan dengan program.

Soal 3.6. Sebuah halaman memiliki 4.000 byte HTML dan 150 kata terlihat yang membentuk 900 karakter teks terlihat. Hitung ρ dengan Persamaan (3.4) dan poin pemeriksaan `text`. Halaman kedua memiliki 40 kata, 242 karakter teks terlihat, dan 1.860 byte HTML; hitung hal yang sama. Mengapa rasio itu merupakan sinyal yang lemah bila berdiri sendiri?

Soal 3.7. Tulis paragraf pembuka jawab-dulu, paling banyak tiga kalimat, untuk halaman Profitina Laundry tentang layanan *dry cleaning*-nya, dengan hanya memakai fakta di **Kode 3.5**. Kemudian sebutkan pertanyaan-pertanyaan yang mungkin diajukan pelanggan kepada mesin jawaban yang dijawab oleh paragraf Anda.

Soal 3.8. Sebutkan tiga alasan Profitina tidak mengukur visibilitas dengan mengkueri mesin jawaban. Kemudian kemukakan argumen terkuat *untuk* mengkuerinya, dan jelaskan mengapa, walaupun demikian, cara itu tidak dapat menggantikan estimasi kesiapan.

Soal 3.9. Skor sebuah situs berubah-ubah hingga 3 poin dari minggu ke minggu tanpa ada perubahan pada situsnya. `robots.txt` yang rusak biasanya menurunkannya 15 poin atau lebih. Pilih ambang δ untuk aturan peringatan pada Subbab 3.6.2, beri alasannya, dan gambarkan satu keadaan ketika pilihan Anda tetap gagal memberi peringatan.

Soal 3.10. RFC 9309 juga membolehkan dua karakter khusus dalam jalur aturan: `*` cocok dengan rangkaian karakter apa pun, dan `$` di akhir jalur menambatkannya ke akhir jalur URL. Tulis fungsi, dalam Python dan Java, yang memutuskan apakah sebuah jalur aturan dengan karakter-karakter itu cocok dengan sebuah jalur URL, lalu uji dengan `/*.pdf$` terhadap `/menu.pdf` dan `/menu.pdf?v=2`, `/status/*/print` terhadap `/status/42/print`, `/status$` terhadap `/status` dan `/status/42`, serta `/serv` terhadap `/services`. Bagaimana Anda memasangnya ke dalam `RobotsRules`?

Soal 3.11. Temukan dua masukan yang membuat `PageFacts.java` dan `page_facts.py` mengumpulkan kata-kata yang berbeda, dan satu masukan yang membuat keduanya mengumpulkan kata-kata yang akan dianggap keliru oleh pembaca. Jelaskan setiap kasus.

Soal 3.12. Laporan Visibilitas Peneliti mengembalikan "tidak ditemukan", tidak pernah skor, untuk nama karangan. Jelaskan dalam satu paragraf pendek mengapa *kontrol negatif* membuat semua angka lain dalam sebuah laporan lebih dapat dipercaya, lalu usulkan satu kontrol negatif untuk lab G1.

3.15 Penyelesaian Soal Latihan

Penyelesaian 3.1. (a) Pantau, di tab *Pantau*; ekspor CSV adalah fitur Pro. (b) Ukur, di tab *Pindai*: fitur ini memastikan pindaian mengukur jenis usaha yang tepat. (c) Perbaiki, di tab *Optimasi*. (d) Jelaskan, di tab *Pindai*: halaman hasil adalah tempat pengukuran berubah menjadi rencana. (e)

Pantau, dikirim setelah pindai ulang terjadwal. (f) Ukur, di tab *Pindai*: label itu melaporkan kemajuan pindaian yang sedang berjalan.

Penyelesaian 3.2. Perubahannya adalah $\Delta_2 = 55 - 52 = +3$, $\Delta_3 = 54 - 55 = -1$, $\Delta_4 = 71 - 54 = +17$, $\Delta_5 = 70 - 71 = -1$, $\Delta_6 = 69 - 70 = -1$, $\Delta_7 = 48 - 69 = -21$, dan $\Delta_8 = 50 - 48 = +2$. Dengan $\delta = 10$, minggu 4 ($+17$) dan 7 (-21) memicu peringatan. Dengan $\delta = 3$, minggu 2, 4, dan 7 memicunya, karena $+3 \geq 3$. Variasi biasa di sini sekitar 1 sampai 3 poin, sehingga $\delta = 3$ memberi peringatan atas derau (minggu 2) dan lama-kelamaan akan diabaikan; $\delta = 10$ menangkap kedua peristiwa nyata, perbaikan pada minggu 4 dan penurunan pada minggu 7, yang tampak seperti kerusakan dan perlu segera diperiksa.

Penyelesaian 3.3. Dengan tangan. `GPTBot` disebut oleh dua grup, grup kedua (bersama `ClaudeBot`) dan grup ketiga (sebagai `gptbot`, karena nama dibandingkan tanpa memedulikan huruf besar-kecil); aturan keduanya digabung menjadi `Disallow: /status` dan `Allow: /status/public`, dan grup `*` diabaikan. Jadi `/` tidak cocok dengan aturan mana pun dan diizinkan; `/status/42` hanya cocok dengan `Disallow: /status` dan diblokir; `/status/public` cocok dengan keduanya, dan `Allow: /status/public` yang lebih panjang (14 karakter lawan 7) menang, sehingga diizinkan. `ClaudeBot` hanya disebut oleh grup kedua, sehingga baginya `/status/public` hanya cocok dengan `Disallow: /status`: diblokir. `PerplexityBot` tidak disebut grup mana pun dan memakai `*`: `/services` cocok dengan `Disallow: /` (1 karakter) dan `Allow: /services` (9), dan yang lebih panjang mengizinkannya; `/outlets` hanya cocok dengan `Disallow: /`: diblokir. **Kode 3.18** mencetak keputusan-keputusan ini dan keputusan `urllib.robotparser` milik Python di sampingnya.

Kode 3.18 `robots_compare.py`: keputusan RFC 9309 dari `RobotsRules` di samping keputusan `urllib.robotparser`.

```
# robots_compare.py - Exercise 3.2: RFC 9309 decisions next to urllib.robotparser
import urllib.robotparser

from robots_rules import RobotsRules

TEXT = """User-agent: *
Disallow: /
Allow: /services

User-agent: GPTBot
User-agent: ClaudeBot
Disallow: /status

User-agent: gptbot
Allow: /status/public
"""
CASES = [("GPTBot", "/"), ("GPTBot", "/status/42"), ("GPTBot", "/status/public"),
         ("ClaudeBot", "/status/public"), ("PerplexityBot", "/services"),
         ("PerplexityBot", "/outlets")]

ours = RobotsRules(TEXT, 200)
theirs = urllib.robotparser.RobotFileParser()
theirs.parse(TEXT.splitlines())
print(f"{'crawler':<14}{'path':<16}{'rfc':<7}{'urllib':<8}reason")
for agent, path in CASES:
    ok, reason = ours.decide(agent, path)
    other = theirs.can_fetch(agent, path)
    print(f"{agent:<14}{path:<16}{'allow' if ok else 'block':<7}"
          f"{'allow' if other else 'block':<8}{reason}")
```

```
crawler       path            rfc    urllib  reason
GPTBot        /               allow  allow   no rule matches in group "GPTBot"
GPTBot        /status/42      block  block   Disallow: /status in group "GPTBot"
GPTBot        /status/public  allow  block   Allow: /status/public in group "GPTBot"
ClaudeBot     /status/public  block  block   Disallow: /status in group "ClaudeBot"
PerplexityBot /services       allow  block   Allow: /services in group "*"
PerplexityBot /outlets        block  block   Disallow: / in group "*"
```

Kolomnya berbeda dua kali, dan kedua perbedaan itu adalah perbedaan yang sudah dikenal dari `urllib.robotparser` pada Python 3.11 yang dipakai di sini [20]. Untuk `GPTBot` pada `/status/public`, modul itu hanya memakai grup *pertama* yang menyebut *crawler* tersebut dan tidak menggabungkan grup ketiga, sehingga tidak pernah melihat `Allow: /status/public`. Untuk `PerplexityBot` pada `/services`, modul itu menerapkan aturan cocok *pertama* menurut urutan berkas, `Disallow: /`, alih-alih yang terpanjang. Program Java, **Kode 3.19**, mencetak kolom RFC 9309; Java tidak memiliki pengurai robot sendiri untuk dibandingkan.

Kode 3.19 `RobotsCompare.java`: keputusan RFC 9309 untuk enam kasus yang sama.

```java
// RobotsCompare.java - Exercise 3.2: the RFC 9309 decisions for six cases
public class RobotsCompare {
    static final String TEXT = """
            User-agent: *
            Disallow: /
            Allow: /services

            User-agent: GPTBot
            User-agent: ClaudeBot
            Disallow: /status

            User-agent: gptbot
            Allow: /status/public
            """;
    static final String[][] CASES = {{"GPTBot", "/"}, {"GPTBot", "/status/42"},
        {"GPTBot", "/status/public"}, {"ClaudeBot", "/status/public"},
        {"PerplexityBot", "/services"}, {"PerplexityBot", "/outlets"}};

    public static void main(String[] args) {
        RobotsRules ours = new RobotsRules(TEXT, 200);
        System.out.printf("%-14s%-16s%-7sreason%n", "crawler", "path", "rfc");
        for (String[] c : CASES) {
            RobotsRules.Decision d = ours.decide(c[0], c[1]);
            System.out.printf("%-14s%-16s%-7s%s%n", c[0], c[1],
                    d.allowed() ? "allow" : "block", d.reason());
        }
    }
}
```

```
crawler       path            rfc    reason
GPTBot        /               allow  no rule matches in group "GPTBot"
GPTBot        /status/42      block  Disallow: /status in group "GPTBot"
GPTBot        /status/public  allow  Allow: /status/public in group "GPTBot"
ClaudeBot     /status/public  block  Disallow: /status in group "ClaudeBot"
PerplexityBot /services       allow  Allow: /services in group "*"
PerplexityBot /outlets        block  Disallow: / in group "*"
```

Penyelesaian 3.4. Pemeriksa membaca `robots.txt` lebih dulu, menemukan grup yang menyebut `ReadinessLab` dengan `Disallow: /`, lalu berhenti sebelum mengambil halaman. Python dan Java mencetak hal yang sama:

```
Stopped: ReadinessLab may not fetch /: Disallow: / in group "ReadinessLab"
exit 2
```

Itu benar: pemilik situs meminta *crawler* tertentu ini untuk tidak masuk, dan pemeriksa yang mengaudit `robots.txt` orang lain harus menaatinya. *Crawler* lain mana pun, termasuk `GPTBot`, tidak disebut oleh grup mana pun, memakai `*`, dan boleh mengambil segalanya karena `Allow: /`. (Ingatlah bahwa `robots.txt` adalah permintaan, bukan kunci: catatan staf di halaman ini memerlukan kata sandi agar benar-benar privat.)

Penyelesaian 3.5. Penyebutnya $1 + 1 + 1 + 2 + 2 + 1 + 1 + 1 = 10$. Poin berbobotnya adalah $1 \cdot \frac{4}{6} = 0.6667$ untuk *crawler* dan $1 \cdot 0.4$ untuk tag meta; `json-ld` dan `headings` berbobot 2 tetapi berpoin 0, sehingga tidak menambah pembilang. Jadi $S = \lfloor 100 \cdot 1.0667/10 + 0.5 \rfloor = \lfloor 10.67 + 0.5 \rfloor = 11$. Menaikkan bobot pemeriksaan yang *gagal* dilalui halaman menurunkan skornya dari 13 menjadi 11. Baris terakhir keluaran program memastikannya:

```
Teaching score: 11 / 100 (weights: json-ld=2, headings=2, others 1)
```

(hanya baris terakhir laporan yang ditampilkan; kedelapan pemeriksaannya sama dengan Subbab 3.10.9, dengan bobot yang baru).

Penyelesaian 3.6. Halaman pertama: $\rho = 900/4000 = 0.225 \geq 0.20$ dan $150 \geq 120$ kata, sehingga pemeriksaan memberi $(1 + 1)/2 = 1$ poin. Halaman kedua (halaman "before"): $\rho = 242/1860 = 0.130$ dan 40 kata, sehingga $(0 + 0)/2 = 0$ poin. Bila berdiri sendiri, rasio itu lemah karena dapat dinaikkan tanpa memperbaiki apa pun yang penting bagi pembaca atau mesin, misalnya dengan memindahkan lembar gaya ke berkas terpisah, yang mengurangi byte pada n_H, dan dapat diturunkan oleh hal yang tidak merugikan, seperti blok JSON-LD yang besar, yang justru *baik* bagi kesiapan. Karena itulah pemeriksaan ini memasangkannya dengan jumlah teks minimum, dan tidak satu pun dari kedua ambangnya bersifat keramat.

Penyelesaian 3.7. Salah satu jawaban (dalam bahasa Inggris, bahasa halaman contoh): "Profitina Laundry in Rungkut, Surabaya, dry-cleans clothes from Rp25,000 for a shirt to Rp75,000 for a wedding dress or a suit. A normal order is ready in two days, and every outlet accepts cash and QRIS." Artinya: Profitina Laundry di Rungkut, Surabaya, melayani *dry cleaning* mulai Rp25.000 untuk kemeja sampai Rp75.000 untuk gaun pengantin atau jas; pesanan biasa selesai dalam dua hari, dan setiap gerai menerima tunai dan QRIS. Paragraf ini menjawab: siapa yang melayani *dry cleaning* di Rungkut atau Surabaya; berapa biaya *dry cleaning* kemeja, jas, atau gaun pengantin di sana; berapa lama pengerjaannya; dan apakah bisa membayar dengan QRIS. Paragraf itu menyebut nama usaha pada kata-kata pertamanya, hanya memuat fakta yang dapat diperiksa dari halaman, dan tidak memakai klaim "ter-".

Penyelesaian 3.8. Kemandirian (layanan dan jawaban pihak ketiga berubah, sehingga pengukuran yang bergantung padanya tidak stabil dan dapat rusak), privasi (kueri akan mengirim nama usaha, pesaing, dan pertanyaan pelanggan kepada penyedia luar), serta prinsip dan biaya (aturan internal melarang ketergantungan pada API pihak ketiga yang berbayar) (Subbab 3.2.3). Argumen terkuat *untuk* mengkueri adalah bahwa cara itu mengamati hasilnya sendiri, yaitu apakah usaha itu benar-benar disebut. Walaupun demikian, cara itu tidak dapat menggantikan estimasi kesiapan: jawaban sebuah mesin berbeda-beda antarpercobaan, antarpengguna, dan antarhari, sehingga satu kueri hanyalah anekdot; ia menunjukkan *bahwa* sebuah usaha tidak muncul tetapi tidak menunjukkan *mengapa*, sehingga tidak memberi perbaikan; dan ia mengukur satu mesin pada satu saat, sedangkan kesiapan adalah sifat situs yang berlaku bagi semua mesin.

Penyelesaian 3.9. Pilihan yang masuk akal adalah $\delta = 8$: lebih dari dua kali variasi biasa sebesar 3 poin, sehingga derau saja hampir tidak pernah memicu peringatan, dan kira-kira separuh

penurunan khas akibat `robots.txt` yang rusak, sehingga kerusakan hampir selalu memicunya. Pilihan ini gagal memberi peringatan bila masalah datang berangsur-angsur, misalnya ketika paragraf jawab-dulu tersunting sedikit demi sedikit selama beberapa minggu dan skornya turun 4 poin per minggu; setiap Δ_t tetap di bawah δ padahal penurunan totalnya besar. Aturan kedua yang membandingkan skor dengan rata-ratanya selama empat minggu terakhir akan menangkap hal itu.

Penyelesaian 3.10. Terjemahkan jalur aturan menjadi ekspresi reguler: *escape* setiap karakter biasa, ganti setiap `*` dengan "rangkaian karakter apa pun", `.*`, dan ubah `$` di akhir menjadi jangkar akhir; lalu cocokkan dari awal jalur URL. **Kode 3.20** dan **Kode 3.21** mengerjakan persis hal itu.

Kode 3.20 `robots_patterns.py`: mencocokkan jalur aturan yang memuat `*` dan `$` di akhir.

```
# robots_patterns.py - Exercise 3.10: the * and $ patterns of RFC 9309
import re


def pattern_regex(rule_path):
    """'*' matches any run of characters; a final '$' anchors the end."""
    anchored = rule_path.endswith("$")
    body = rule_path[:-1] if anchored else rule_path
    regex = ".*".join(re.escape(part) for part in body.split("*"))
    return re.compile(regex + ("$" if anchored else ""))


def matches(rule_path, path):
    return pattern_regex(rule_path).match(path) is not None   # from the start


CASES = [("/*.pdf$", "/menu.pdf"), ("/*.pdf$", "/menu.pdf?v=2"),
         ("/status/*/print", "/status/42/print"), ("/status$", "/status"),
         ("/status$", "/status/42"), ("/serv", "/services")]

if __name__ == "__main__":
    for rule, path in CASES:
        print(f"{rule:<17} {path:<18} {'match' if matches(rule, path) else '-'}")
```

Kode 3.21 `RobotsPatterns.java`: pencocokan yang sama dalam Java, dengan `Pattern.quote` untuk bagian-bagian biasa.

```java
// RobotsPatterns.java - Exercise 3.10: the * and $ patterns of RFC 9309
import java.util.regex.Pattern;

public class RobotsPatterns {
    /** '*' matches any run of characters; a final '$' anchors the end. */
    static Pattern patternRegex(String rulePath) {
        boolean anchored = rulePath.endsWith("$");
        String body = anchored ? rulePath.substring(0, rulePath.length() - 1)
                : rulePath;
        StringBuilder regex = new StringBuilder();
        String[] parts = body.split("\\*", -1);
        for (int i = 0; i < parts.length; i++) {
            regex.append(i == 0 ? "" : ".*").append(Pattern.quote(parts[i]));
        }
        return Pattern.compile(regex + (anchored ? "$" : ""));
    }

    static boolean matches(String rulePath, String path) {
        return patternRegex(rulePath).matcher(path).lookingAt(); // from the start
    }

    public static void main(String[] args) {
        String[][] cases = {{"/*.pdf$", "/menu.pdf"},
            {"/*.pdf$", "/menu.pdf?v=2"},
            {"/status/*/print", "/status/42/print"}, {"/status$", "/status"},
            {"/status$", "/status/42"}, {"/serv", "/services"}};
        for (String[] c : cases) {
            System.out.printf("%-17s %-18s %s%n", c[0], c[1],
                    matches(c[0], c[1]) ? "match" : "-");
        }
    }
}
```

Keduanya mencetak:

```
/*.pdf$            /menu.pdf          match
/*.pdf$            /menu.pdf?v=2      -
/status/*/print    /status/42/print   match
/status$           /status            match
/status$           /status/42         -
/serv              /services          match
```

`/menu.pdf?v=2` tidak cocok dengan `/*.pdf$` karena `$` mengharuskan jalur berakhir tepat sesudah `.pdf`. Untuk memasang fungsi itu ke dalam `RobotsRules`, ganti pengujian `path.startswith(prefix)` (`path.startsWith(r.path())` di Java) dengan `matches(prefix, path)`. Panjang yang dipakai untuk "paling spesifik" tetap panjang jalur aturan sebagaimana tertulis, termasuk karakter khususnya, karena RFC 9309 mendefinisikan kecocokan paling spesifik sebagai kecocokan dengan oktet terbanyak pada aturan [7].

Penyelesaian 3.11. Dua perbedaan. Pertama, entitas bernama: untuk `<p>Café</p>` Python mengumpulkan `Café`, karena `html.parser` mengenal setiap entitas HTML, sedangkan pemecah token Java hanya menerjemahkan beberapa entitas umum dan mengumpulkan `Café`. Kedua, markah yang rusak: untuk `<p class="a>text</p><p>next words</p>`, yang kehilangan tanda kutip penutup, pengurai Python membaca segala sesuatu sesudah tanda kutip sebagai bagian nilai atribut dan tidak mengumpulkan satu kata pun, sedangkan ekspresi reguler Java tidak dapat mencocokkan tag yang rusak itu dan mengumpulkan `<p`, `class="a>text`, `next`, dan `words`. Tidak satu pun sesuai dengan maksud orang yang menulis halaman itu. Kasus ketika keduanya keliru: pada `<p>A<b>B</b>C</p>`, yang dilihat pembaca sebagai satu kata "ABC" dengan huruf tengah tebal, keduanya mengumpulkan tiga kata, `A`, `B`, dan `C`, karena memperlakukan setiap tag sebagai batas kata. Alat ekstraksi sungguhan mengetahui elemen mana yang bersifat *inline*

dan menggabungkan teksnya. Tidak satu pun dari kasus ini terjadi pada halaman contoh, sehingga kedua versi sepakat pada halaman-halaman itu; di web terbuka, pengurai lengkaplah alat yang tepat.

Penyelesaian 3.12. Sistem yang mengembalikan angka untuk setiap masukan tidak dapat menunjukkan bahwa angkanya berasal dari data, bukan dari kebiasaannya sendiri. Bila nama karangan selalu mendapat "tidak ditemukan" dan tidak pernah skor, setiap skor yang *dikembalikan* diketahui bersandar pada rekaman nyata; bila suatu saat nama karangan mendapat skor yang tampak masuk akal, metodenya rusak, dan kontrol itu mengungkapkannya sebelum orang sungguhan dinilai keliru. Untuk lab G1, kontrol negatif yang baik adalah halaman tanpa konten sama sekali, `<html></html>` kosong yang disajikan situs tanpa `robots.txt`: setiap pemeriksaan yang bergantung pada konten harus memberi 0 poin, dan laporan harus menyebut alasannya. (Situs tanpa `robots.txt` tetap memperoleh poin *crawler*, yang benar menurut RFC 9309 dan merupakan uji yang baik bahwa pemeriksa membaca standar, bukan harapan kita.)

Daftar Pustaka

[1] P. Aggarwal, V. Murahari, T. Rajpurohit, A. Kalyan, K. Narasimhan, and A. Deshpande, "GEO: Generative Engine Optimization," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD '24)*, 2024. arXiv:2311.09735. [Online]. Available: https://arxiv.org/abs/2311.09735

[2] Profitina, *Profitina Whitepaper — Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[3] Profitina, "Syarat dan Ketentuan" (Terms and Conditions), v1.1, effective Aug. 6, 2026, §7. [Online]. Available: https://app.profitina.com/legal/

[4] OpenAI, "Overview of OpenAI crawlers." [Online]. Available: https://developers.openai.com/api/docs/bots (accessed Sep. 24, 2026).

[5] Anthropic, "Does Anthropic crawl data from the web, and how can site owners block the crawler?" *Claude Help Center*. [Online]. Available: https://support.claude.com/en/articles/8896518 (accessed Sep. 24, 2026).

[6] Perplexity, "Perplexity crawlers." [Online]. Available: https://docs.perplexity.ai/guides/bots (accessed Sep. 24, 2026).

[7] M. Koster, G. Illyes, H. Zeller, and L. Sassman, "Robots Exclusion Protocol," IETF RFC 9309, Sep. 2022.

[8] Sitemaps.org, "Sitemaps XML format," protocol version 0.9. [Online]. Available: https://www.sitemaps.org/protocol.html

[9] Google, "Google's common crawlers," *Google Search Central*. [Online]. Available: https://developers.google.com/search/docs/crawling-indexing/google-common-crawlers (accessed Sep. 24, 2026).

[10] Microsoft, "Which crawlers does Bing use?" *Bing Webmaster Tools Help*. [Online]. Available: https://www.bing.com/webmasters/help/which-crawlers-does-bing-use-8c184ec0 (accessed Sep. 2026).

[11] Schema.org, "DryCleaningOrLaundry," Schema.org v30.1, Sep. 16, 2026. [Online]. Available: https://schema.org/DryCleaningOrLaundry

[12] W3C, "JSON-LD 1.1: A JSON-based Serialization for Linked Data," W3C Recommendation, Jul. 16, 2020. [Online]. Available: https://www.w3.org/TR/json-ld11/

[13] J. Howard, "The /llms.txt file," llmstxt.org, proposed Sep. 3, 2024, updated Aug. 10, 2026. [Online]. Available: https://llmstxt.org/

[14] WHATWG, *HTML Living Standard*, sections "The title element," "The meta element" and "Headings and outlines." [Online]. Available: https://html.spec.whatwg.org/ (accessed Sep. 2026).

[15] OurResearch, "OpenAlex: The open catalog to the global research system." [Online]. Available: https://openalex.org/ (accessed Sep. 2026).

[16] ORCID, "What is ORCID?" [Online]. Available: https://info.orcid.org/what-is-orcid/ (accessed Sep. 2026).

[17] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*, Oct. 17, 2022.

[18] Python Software Foundation, "http.server — HTTP servers" and "urllib.request — Extensible library for opening URLs," *The Python Standard Library*. [Online]. Available: https://docs.python.org/3/library/http.server.html

[19] Oracle, "Package com.sun.net.httpserver," *Java SE 25 & JDK 25 API Specification*. [Online]. Available: https://docs.oracle.com/en/java/javase/25/docs/api/jdk.httpserver/com/sun/net/httpserver/package-summary.html

[20] Python Software Foundation, "Support RFC 9309 in robotparser," CPython issue #138907, Sep. 15, 2025. [Online]. Available: https://github.com/python/cpython/issues/138907

[21] Python Software Foundation, "html.parser — Simple HTML and XHTML parser," *The Python Standard Library*. [Online]. Available: https://docs.python.org/3/library/html.parser.html

[22] Oracle, "Package java.net.http," *Java SE 25 & JDK 25 API Specification*. [Online]. Available: https://docs.oracle.com/en/java/javase/25/docs/api/java.net.http/java/net/http/package-summary.html

[23] PPC Land, "llms.txt adoption rises 8.8x but 97% of files get zero AI requests," Jul. 2, 2026. [Online]. Available: https://ppc.land/llms-txt-adoption-rises-8-8x-but-97-of-files-get-zero-ai-requests/

[24] Rankability, "llms.txt adoption," scan of Sep. 18, 2026 (Tranco list of Sep. 17, 2026). [Online]. Available: https://www.rankability.com/data/llms-txt-adoption/

Bab 4
Komunitas yang Dirancang

Di akhir bab ini Anda akan mengetahui terbuat dari apa komunitas Profitina dan mengapa setiap bagiannya berbentuk seperti itu: umpan langsung yang memperbarui dirinya sendiri, utas dan repost-kutip, kanal yang boleh dibuka siapa saja yang saldo PROFIT-nya cukup tanpa harus membayar, penghitung "N baru" yang hanya menghitung hal yang penting, pesan satu lawan satu yang tetap privat, profil yang setiap butirnya hanya dibagikan atas persetujuan, dan dasbor yang berperilaku seperti kokpit. Anda akan memahami tiga cara sebuah halaman web mengetahui adanya data baru, yaitu *polling*, WebSocket, dan *Server-Sent Events*, cukup dalam untuk memilih di antara ketiganya dan menghitung biaya masing-masing. Anda akan melihat, dengan pengukuran sungguhan dari rilis September 2026, tiga cara umpan waktu-nyata bisa gagal dan bagaimana masing-masing diperbaiki. Dan Anda akan membangun sendiri umpan langsung untuk Profitina Laundry, dengan server dan klien konsol dalam Python dan Java serta halaman peramban yang mengirim seketika, membatalkan kiriman saat gagal, menghitung kiriman belum dibaca per kanal, dan mengabaikan balasan yang terlambat, semuanya diperiksa otomatis di peramban Chromium tanpa layar (*headless*).

4.1 Posisi Kita Sekarang

Bab 3 mengikuti seorang pengguna melewati siklus visibilitas: memindai situs, memperbaiki temuan pindaian, lalu terus memantau. Siklus itu dikerjakan pemilik usaha seorang diri, di mejanya, dengan situs webnya sendiri. Siklus itu berguna, dan bagi banyak pengguna itulah satu-satunya alasan mereka datang. Namun semboyan di setiap halaman Profitina terdiri atas tiga kata, *Bersama · Berkembang · Berbagi* (dalam bahasa Inggris, *Gather · Grow · Give*), dan siklus Bab 3 hanya benar-benar melayani kata yang di tengah. Bab ini membahas kata pertama dan terakhir: tempat orang *berkumpul bersama*, dan tempat pengetahuan mereka *dibagikan* kepada orang lain [1].

Tempat itu adalah komunitas. Komunitas bukan tambahan yang ditempelkan di samping pemindai. Ia tertanam di dalam aplikasi dan, sebagai widget, di seluruh situs pemasaran; di sanalah sebuah kelas berisi sekitar 250 mahasiswa bekerja sejak 31 Agustus 2026; dan di sanalah sebagian besar upaya rekayasa platform dicurahkan pada Agustus dan September 2026. Seperti akan kita lihat, komunitas juga bagian Profitina yang paling mirip dengan media sosial yang sudah dikenal semua orang, sekaligus bagian tempat Profitina paling sengaja berperilaku berbeda.

Bab ini bergerak dari luar ke dalam. Subbab 4.2 bertanya mengapa alat visibilitas memerlukan komunitas. Subbab 4.3 menggambar peta bagian-bagiannya. Subbab 4.4 menjelaskan teknologi yang membuatnya hidup, *Server-Sent Events*, dengan membandingkannya dengan dua alternatifnya dan menghitung biaya masing-masing. Subbab 4.5 sampai 4.10 membahas bagian-bagiannya satu per satu: umpan dan utasnya, kanal, penghitung belum dibaca, pesan langsung dan profil, kokpit dasbor, serta pengiriman seketika. Subbab 4.11 menceritakan tiga kisah nyata tentang hal yang keliru, lengkap dengan angkanya. Subbab 4.12 adalah lab, tempat Anda membangun seluruh mekanisme dalam bentuk mini.

Satu janji sebelum mulai. Semua hal di bab ini dijelaskan pada tingkat *kotak dan panah*: apa yang dikerjakan setiap bagian dan mengapa. Kode lab adalah kode kita sendiri, ditulis untuk buku ini, dan sengaja dibuat cukup sederhana untuk dibaca dalam satu sore. Kode itu bukan kode Profitina, dan memang tidak perlu: yang berpindah adalah gagasannya.

4.2 Mengapa Komunitas Dibangun di dalam Alat Visibilitas

4.2.1 Orang sudah hidup di dalam umpan

Indonesia termasuk masyarakat yang paling terhubung di dunia. Asosiasi penyelenggara jasa internet APJII mencatat 235.261.078 pengguna internet dalam survei 2026, yaitu 81,72 % dari 287.303.234 penduduk, naik dari 80,66 % setahun sebelumnya [2]. Laporan *Digital 2026* DataReportal untuk Indonesia mencatat 180 juta identitas pengguna media sosial aktif pada Oktober 2025, setara dengan 62,9 % penduduk [3]. Bagi kebanyakan orang itu, pengalaman sehari-hari dengan internet *adalah* umpan: kolom butir-butir pendek, yang terbaru di atas, yang terus bertambah selagi kita melihatnya.

Produk yang ingin didatangi kembali harus menemui orang di tempat mereka sudah berada. Laporan pindaian, sebaik apa pun, dibaca sekali lalu disimpan. Percakapan tentang laporan itu (“skor saya naik setelah menambahkan markah FAQ, ini yang saya ubah”; “mengapa kategori saya tertulis *belum yakin*?”) dibaca banyak orang, dijawab sebagian, dan diingat. Komunitas mengubah kerja perorangan menjadi pengetahuan bersama. Itulah *Berbagi* dalam semboyan tadi.

4.2.2 Apa yang dilakukan Profitina secara berbeda

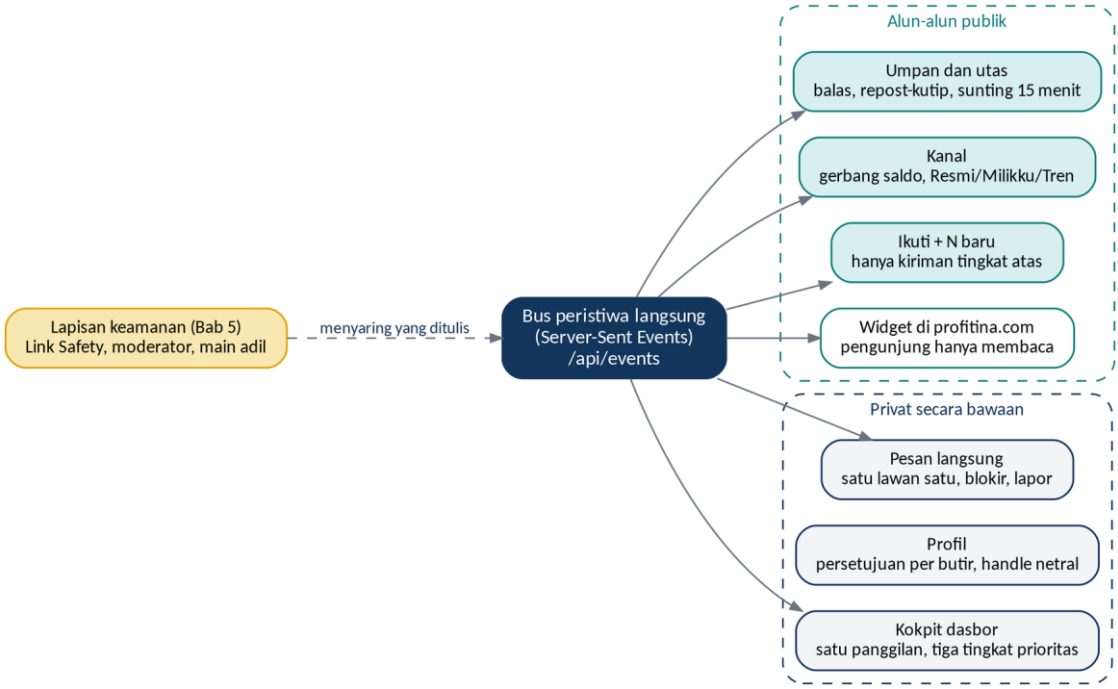
Komunitas Profitina meminjam bentuk-bentuk yang sudah akrab, yaitu umpan yang diperbarui secara waktu-nyata seperti aplikasi obrolan, tampilan utas seperti X, dan repost-kutip dengan komentar dan foto opsional, tetapi mengubah aturan di bawahnya dalam empat hal yang terus muncul sepanjang bab ini [1].

1. **Membaca gratis; bertindak memerlukan satu akun.** Pengunjung anonim profitina.com dapat membaca widget komunitas. Mengirim, membalas, menyukai, atau mengikuti memerlukan masuk (*sign in*). Tidak ada “pengirim tamu” kelas dua.
2. **Yang dihargai adalah kontribusi, bukan perhatian.** Hadiah PROFIT kecil diberikan untuk kontribusi yang berguna dari manusia yang surelnya terverifikasi, dengan batas harian; Bab 5 menjelaskan perangkat main adil dan Bab 12 menjelaskan koinnya. Tidak ada iklan, selamanya, sehingga tidak ada alasan bisnis untuk memaksimalkan waktu yang dihabiskan menatap umpan.
3. **Hal yang privat tetap privat secara bawaan.** Pesan langsung bersifat satu lawan satu, profil dimulai dalam keadaan privat, *handle* bersifat netral, dan alamat surel tidak pernah dapat dipublikasikan. Subbab 4.8 membahas setiap aturannya.
4. **Satu angka, satu makna.** Penghitung yang bertuliskan “3 baru” harus bermakna persis sama di mana pun ia muncul. Subbab 4.7 menunjukkan betapa mudahnya hal ini keliru.

> **WAWASAN KUNCI.** Komunitas bukan fitur yang ditambahkan sekali; komunitas adalah sekumpulan janji yang harus ditepati setiap kali halaman diperbarui. “Ini baru”, “ini privat”, “ini sudah ditinjau”, “angka ini benar”: setiap janji murah untuk diucapkan dan mahal untuk ditepati di bawah lalu lintas sungguhan. Sebagian besar bab ini membahas biaya menepati janji-janji itu.

4.3 Peta Komunitas

Gambar 4.1 menunjukkan bagian-bagian komunitas dan bagaimana semuanya terhubung. Di tengah terdapat *bus peristiwa langsung*: satu aliran pemberitahuan kecil, dilayani di `/api/events`, yang memberi tahu setiap halaman yang terbuka ketika ada perubahan. Di sekelilingnya ada bagian-bagian yang benar-benar dilihat anggota, dikelompokkan menjadi alun-alun publik (umpan dan utas, kanal, ikutan dan penghitung, serta widget di profitina.com) dan ruang privat (pesan langsung, profil, dan dasbor). Di sebelah kiri ada lapisan keamanan, yang menyaring apa yang ditulis anggota sebelum sampai kepada orang lain; lapisan itu menjadi pokok bahasan Bab 5.



Gambar 4.1 Komunitas pada tingkat kotak. Satu bus peristiwa langsung memasok setiap bagian alun-alun publik dan ruang privat; lapisan keamanan Bab 5 menyaring apa yang ditulis sebelum ditampilkan.

Tabel 4.1 mendaftar bagian-bagian itu beserta tanggal masing-masing mulai berjalan di produksi, diambil dari *whitepaper* publik proyek dan riwayat rilisnya [1]. Dua hal menonjol. Pertama, seluruh komunitas dibangun dalam sekitar sepuluh minggu, antara peluncuran Juli 2026 dan akhir September 2026. Kedua, beberapa bagian hadir *karena* kelas berisi sekitar 250 mahasiswa yang mulai memakai platform pada 31 Agustus 2026: moderator kanal dan *handle* netral dibangun pada pertengahan Agustus khusus agar sebuah kelas universitas dapat bekerja secara terbuka tanpa membuka nomor induk mahasiswa siapa pun.

Tabel 4.1 Bagian-bagian komunitas Profitina dan kapan masing-masing mulai berjalan di produksi (tanggal dari *whitepaper* publik dan riwayat rilis).

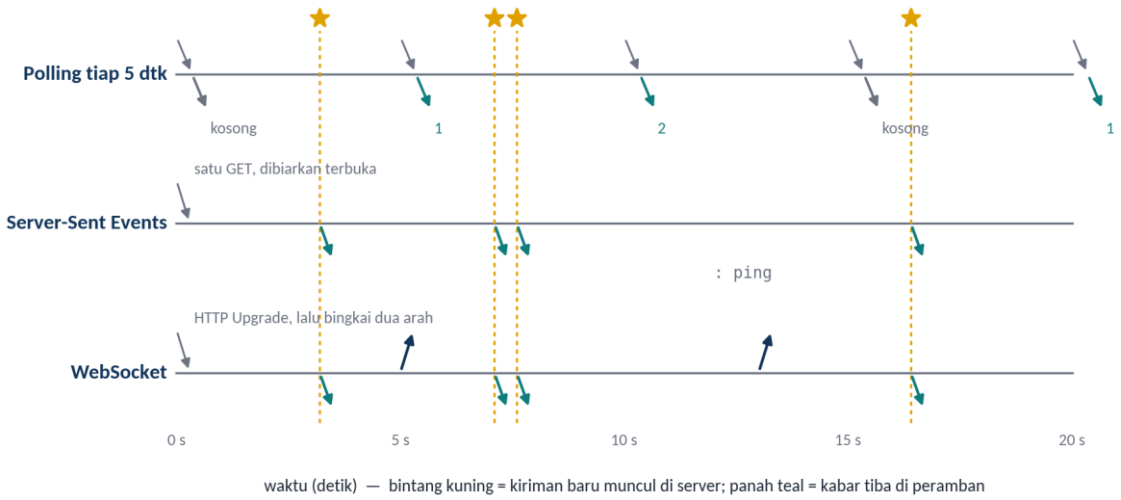
Bagian	Fungsinya	Berjalan di produksi sejak
Umpan langsung lewat SSE	Kiriman baru muncul tanpa memuat ulang	Juli 2026
Utas dan repost-kutip	Balasan bersarang; repost boleh membawa komentar dan foto	Juli 2026
Kanal buatan pengguna	Ruang topik atau kelas, dengan gerbang saldo PROFIT	Juli 2026
Jendela sunting 15 menit	Memperbaiki salah ketik; hadiah dihitung ulang	12 Agustus 2026
Moderator kanal	Menyembunyikan, menyematkan, membisukan di satu kanal saja	13 Agustus 2026
Handle netral	u acak + 8 digit heksadesimal sebagai ganti pengenal asli	14 Agustus 2026
Pesan langsung	Satu lawan satu, dengan blokir dan lapor	20 Agustus 2026
Profil dengan persetujuan per butir	Foto dan empat tautan opsional, masing-masing dibagikan atas pilihan	20 Agustus 2026
Kokpit dasbor	Satu panggilan mengumpulkan semua yang perlu perhatian	15–16 September 2026
Pengiriman seketika (optimistis)	Kiriman tampil sebelum server menjawab	15–16 September 2026
Pindah kanal seketika	Kanal terbuka dalam milidetik; repost tetap di kanalnya	23–24 September 2026

4.4 Waktu-Nyata Tanpa Sihir: Polling, WebSocket, dan Server-Sent Events

4.4.1 Masalahnya: web dibangun untuk bertanya

Percakapan dasar web adalah permintaan yang diikuti tanggapan. Peramban *bertanya* (“berikan halamannya”, “berikan umpannya”) dan server *menjawab*. Tidak ada satu pun dalam pola itu yang memungkinkan server berbicara lebih dulu. Padahal umpan langsung justru situasi ketika server mengetahui sesuatu yang tidak diketahui peramban: seseorang baru saja mengirim. Bagaimana kabar itu sampai dari server ke halaman yang tidak bertanya?

Ada tiga jawaban klasik, digambar berdampingan pada **Gambar 4.2**. Bintang kuning menandai saat kiriman baru muncul di server; panah teal menandai saat kabar itu sampai di peramban.



Gambar 4.2 Tiga cara kabar sampai ke peramban. Polling bertanya setiap lima detik dan sering tidak mendapat apa-apa; Server-Sent Events membiarkan satu tanggapan tetap terbuka dan mendorong setiap kiriman saat terjadi; WebSocket meningkatkan sambungan agar kedua pihak dapat mengirim kapan saja.

4.4.2 Polling: bertanya lagi dan lagi

Jawaban paling sederhana adalah *polling*: halaman bertanya "ada yang baru?" setiap T detik. Cara ini berjalan dengan server dan peramban apa pun, dan begitulah banyak halaman obrolan awal dibangun. Biayanya mudah dihitung, dan menghitungnya adalah cara terbaik untuk melihat mengapa sistem waktu-nyata meninggalkannya.

Biaya 1: permintaan. Jika ada N halaman terbuka dan masing-masing melakukan polling setiap T detik, server menerima

$$R = \frac{3600\,N}{T} \tag{4.1}$$

permintaan per jam. Untuk kelas pertama Profitina saja, $N = 250$ halaman terbuka yang melakukan polling setiap $T = 5$ detik menghasilkan, menurut Persamaan (4.1), $R = 3600 \times 250/5 = 180{,}000$ permintaan per jam, atau 50 per detik, entah ada yang mengirim atau tidak.

Biaya 2: tunda. Kiriman yang muncul pada saat acak harus menunggu polling berikutnya. Jika saat itu sama mungkinnya jatuh di mana saja dalam selang sepanjang T di antara dua polling, waktu tunggu W berdistribusi seragam pada $[0, T]$, dan rata-ratanya

$$\mathbb{E}[W] = \frac{1}{T}\int_0^T w\;dw = \frac{1}{T}\cdot\frac{T^2}{2} = \frac{T}{2}. \tag{4.2}$$

Dengan $T = 5$ detik, kiriman baru rata-rata perlu 2,5 detik untuk tampil, dan paling lama 5 detik. Memperkecil T menjadi separuh memangkas tunda menjadi separuh tetapi, menurut Persamaan (4.1), melipatgandakan permintaan. Polling memaksa tawar-menawar antara kesegaran dan beban, dan tidak ada pengaturan yang baik untuk keduanya.

Biaya 3: pemborosan. Kebanyakan polling pulang dengan tangan kosong. Misalkan kiriman datang secara acak dengan laju rata-rata λ per detik, saling bebas (sebuah *proses Poisson*, model baku untuk kedatangan seperti panggilan telepon atau pelanggan). Peluang bahwa selang sepanjang T tidak berisi kiriman sama sekali adalah

$$P(\text{polling kosong}) = e^{-\lambda T}. \tag{4.3}$$

Ambil kanal yang ramai dengan satu kiriman per menit, jadi $\lambda = 1/60$, yang dipolling setiap $T = 5$ detik. Maka $\lambda T = 5/60 \approx 0.083$ dan Persamaan (4.3) memberikan $e^{-0.083} \approx 0.920$. Sekitar 92 % dari semua polling tidak membawa apa-apa selain jawaban "tidak ada". Jalur polling pada **Gambar 4.2** memperlihatkan dua jawaban kosong seperti itu di antara lima.

4.4.3 WebSocket: pipa dua arah

Jawaban kedua adalah protokol WebSocket, dibakukan pada 2011 sebagai RFC 6455 [4]. Peramban membuka permintaan HTTP biasa yang meminta sambungan *ditingkatkan* (*upgrade*); jika server setuju, sambungan itu berhenti menjadi HTTP dan berubah menjadi pipa dua arah tempat kedua pihak dapat mengirim *bingkai* (*frame*) kecil kapan saja. Gim daring, penyunting kolaboratif, dan layar perdagangan memakai WebSocket karena memerlukan lalu lintas dua arah, berkali-kali dalam satu detik.

Harganya, pipa itu bukan lagi HTTP. *Cache*, *proxy*, dan penyeimbang beban yang memahami HTTP harus diajari tentang peningkatan itu; aplikasi harus menciptakan format pesannya sendiri, logika penyambungan ulangnya sendiri, dan caranya sendiri untuk menyusul setelah terputus. Untuk umpan, ketika server hampir selalu yang berbicara dan peramban hanya sesekali mengirim, sebagian besar kekuatan itu tidak terpakai.

4.4.4 Server-Sent Events: satu jawaban yang panjang

Jawaban ketiga, yang dipakai Profitina, adalah *Server-Sent Events* (SSE), yang didefinisikan dalam HTML Living Standard [5] dan tersedia di setiap peramban masa kini melalui antarmuka `EventSource` [6]. Gagasannya nyaris terlalu sederhana: peramban membuat satu permintaan GET biasa, dan server *tidak pernah selesai menjawabnya*. Server mengirim *header*, lalu membiarkan tanggapan tetap terbuka dan menuliskan beberapa baris teks setiap kali sesuatu terjadi.

> **DEFINISI.** *Aliran Server-Sent Events* adalah tanggapan HTTP dengan tipe konten `text/event-stream` yang badannya berupa rangkaian *peristiwa* (*event*). Setiap peristiwa adalah sekelompok baris berbentuk `field: value`, diakhiri satu baris kosong. Medannya adalah `data` (muatan), `event` (nama jenis peristiwa), `id` (label yang diingat peramban), dan `retry` (berapa milidetik menunggu sebelum menyambung ulang). Baris yang diawali titik dua adalah *komentar* dan diabaikan.

Berikut apa yang benar-benar ditulis server lab di Subbab 4.12, bita demi bita, ketika seorang pendengar tersambung lalu dua kiriman menyusul:

```
retry: 3000

id: 1
event: post
data: {"id":1,"channel":"general","top":true}

: ping

id: 2
event: post
data: {"id":2,"channel":"tips","top":true}
```

Bacalah baris demi baris. Peristiwa pertama hanya membawa `retry: 3000`: "kalau kamu kehilangan aku, tunggu tiga detik lalu kembali". Peristiwa berikutnya menyatakan bahwa kiriman nomor 1 telah muncul di kanal *general* dan bahwa kiriman itu kiriman tingkat atas, bukan

balasan. Lalu datang `: ping`, baris komentar yang sama sekali tidak bermakna, kemudian kiriman nomor 2.

Empat sifat membuat SSE cocok untuk umpan:

- **SSE adalah HTTP biasa.** Permintaannya GET, tanggapannya teks, dan setiap *proxy*, terowongan, dan *firewall* yang meloloskan halaman web juga meloloskannya. *Host* produksi Profitina berada di balik terowongan yang hanya membuat sambungan keluar; SSE tidak memerlukan apa pun yang istimewa darinya.
- **Penyambungan ulang sudah tersedia.** Jika sambungan putus, `EventSource` menunggu selama selang `retry` lalu menyambung ulang dengan sendirinya. Saat menyambung ulang, ia mengirim *header* `Last-Event-ID` berisi `id` peristiwa terakhir yang diterimanya, sehingga server yang menyimpan riwayat pendek dapat *memutar ulang* (*replay*) persis apa yang terlewat. Lab mengimplementasikan pemutaran ulang itu.
- **Detak jantung (*heartbeat*) tanpa biaya.** Baris komentar seperti `: ping` menjaga sambungan yang diam tetap hidup melewati *proxy* yang menutup sambungan senyap, dan memungkinkan server mengetahui bahwa seorang pendengar sudah pergi: menulis ke sambungan yang tertutup akan gagal. Detak jantung lab secara bawaan lima belas detik.
- **Satu arah, dan itu cukup.** Peramban tetap mengirim kiriman dengan permintaan POST biasa. Aliran hanya untuk kabar.

SSE punya dua keterbatasan yang patut diketahui. SSE hanya membawa teks, sehingga data biner seperti foto berjalan lewat permintaan biasa, yang memang diinginkan umpan. Dan di bawah HTTP/1.1 sebuah peramban membuka paling banyak enam sambungan ke situs yang sama, dipakai bersama oleh semua tabnya; setiap aliran yang terbuka memegang satu sambungan, sehingga pengguna dengan tujuh tab terbuka mungkin melihat tab ketujuh macet. Di bawah HTTP/2 aliran-aliran itu berbagi satu sambungan dan batas itu praktis hilang [6].

Tabel 4.2 merangkum perbandingannya.

Tabel 4.2 Perbandingan polling, WebSocket, dan Server-Sent Events.

Sifat	Polling setiap T dtk	WebSocket	Server-Sent Events
Arah	Peramban bertanya	Dua arah	Server ke peramban
Protokol	HTTP	Peningkatan HTTP, lalu bingkai sendiri	HTTP (`text/event-stream`)
Tunda rata-rata	$T/2$, menurut Persamaan (4.2)	Hampir nol	Hampir nol
Permintaan per jam, N halaman	$3600N/T$, menurut Persamaan (4.1)	N sambungan, dipertahankan	N sambungan, dipertahankan
Sambung ulang dan menyusul	Polling berikutnya	Urusan aplikasi	Sudah tersedia (`retry`, `Last-Event-ID`)
Data	Apa saja	Teks atau biner	Teks (UTF-8)
Cocok untuk	Pembaruan jarang, server paling sederhana	Gim, penyunting, lalu lintas dua arah	Umpan, notifikasi, dasbor

JEBAKAN. "Waktu-nyata" tidak berarti "kirim semuanya sekaligus". Aliran seharusnya membawa *pemberitahuan kecil*, bukan kiriman utuh beserta foto dan pohon balasannya. Subbab 4.11.2 menunjukkan apa yang terjadi bila setiap pemberitahuan membuat setiap halaman mengambil ulang segalanya.

4.4.5 Peristiwa tipis

Peristiwa pada contoh di atas hanya membawa tiga hal: nomor kiriman, kanalnya, dan apakah kiriman itu tingkat atas. Peristiwa itu menyatakan *bahwa* sesuatu terjadi, bukan *apa* isinya. Halaman yang peduli mengambil satu kiriman yang dibutuhkannya; halaman yang tidak peduli hanya memperbarui penghitung. Kita menyebut pemberitahuan seperti itu *peristiwa tipis* (*thin event*), dan Subbab 4.11.2 menjelaskan mengapa perbedaan antara *menandai basi* (*invalidate*) dan *mengambil ulang* (*refetch*) ternyata sangat penting di produksi.

4.5 Umpan, Utas, dan Repost-Kutip

4.5.1 Umpan adalah daftar; utas adalah pohon

Umpan sebuah kanal adalah daftar kirimannya, yang terbaru lebih dulu. *Balasan* adalah kiriman yang menyebut kiriman lain sebagai induknya. Ikuti induk-induknya dan kiriman-kiriman itu membentuk *pohon*: kiriman asli adalah akar, balasannya adalah anak, balasan atas balasan adalah cucu, dan seterusnya. Tampilan utas sebuah kiriman menampilkan pohon itu.

Gambar 4.3 memperlihatkan sebuah utas kecil di kanal kelas. Kiriman 1 mengumumkan tugas. Kiriman 2 menanyakan tenggat dan kiriman 3 menjawabnya; kiriman 4 sekadar berterima kasih. Kiriman 5 dan 6 sama sekali bukan balasan: keduanya *repost*, yang dijelaskan subbagian berikutnya.

Gambar 4.3 Utas adalah pohon balasan. Repost-kutip (#5) menunjuk balik ke kiriman yang dikutipnya dan, kecuali pembuatnya memilih lain, mewarisi kanal kiriman asli; repost polos yang sengaja dibagikan ke Umum (#6) pergi ke sana.

Karena setiap balasan menyimpan nomor induknya, pohon itu tidak memerlukan struktur tambahan: utas kiriman r adalah kiriman r bersama setiap kiriman yang rantai induknya mencapai r. Lab menyimpan tepat satu medan untuk ini, `reply_to`, yang kosong (`null`) untuk kiriman tingkat atas.

4.5.2 Repost dan tempatnya

Repost membagikan kiriman yang sudah ada kepada lebih banyak orang. Repost *polos* membagikannya apa adanya; *repost-kutip* menambahkan komentar pembagi dan, bila mau, sebuah foto, lalu menampilkan kiriman asli di bawahnya. Profitina memiliki keduanya [1].

Pertanyaan yang menarik adalah *di mana* repost tinggal. Misalkan seorang mahasiswa me-repost pengumuman tugas dari dosennya. Haruskah repost itu muncul di kanal kelas, tempat pengumuman berada, atau di kanal umum, tempat mahasiswa itu kebetulan sedang melihat? Pada rilis September 2026 aturannya menjadi: **repost mewarisi kanal kiriman asli, kecuali orang yang me-repost dengan sengaja memilih Umum**. Dialog repost menawarkan tepat dua pilihan

itu, dengan kanal kiriman asli terpilih secara bawaan. Alasannya sama dengan yang dipakai alat obrolan tim yang dikenal kebanyakan orang: repost adalah kelanjutan percakapan, dan percakapan milik ruang tempat ia dimulai. Repost yang diam-diam mendarat di ruang lain menceraiberaikan diskusi dan mengacaukan penghitung ruang itu, karena muncul butir baru di kanal yang tidak dikirimi siapa pun di kanal tersebut.

4.5.3 Jendela sunting lima belas menit

Setiap orang yang pernah mengirim sesuatu pasti pernah melihat salah ketik satu detik setelah menekan *Kirim*. Sejak 12 Agustus 2026 kiriman Profitina dapat disunting dalam lima belas menit setelah dikirim, ala WhatsApp [1]. Ditulis sebagai aturan, kiriman yang dibuat pada waktu t_0 boleh disunting pada waktu t jika dan hanya jika

$$t - t_0 \le 15 \text{ menit}. \tag{4.4}$$

Dua rincian membuat aturan ini jujur. Pertama, ketika kiriman disunting, hadiah PROFIT kecil yang diperolehnya *dihitung ulang sepenuhnya* dari teks yang baru, dan bisa turun maupun naik; kalau tidak, jendela itu menjadi celah: kiriman remeh ditulis lebih dulu lalu diam-diam ditingkatkan, atau kiriman berguna diberi hadiah lalu dikosongkan. Kedua, setelah jendela tertutup teksnya final, sehingga balasan yang ditulis untuk sebuah kiriman tidak akan mendapati dirinya menjawab kiriman yang berbeda. Persamaan (4.4) memang pendek, tetapi melindungi setiap balasan dalam utas di bawahnya.

4.6 Kanal dan Gerbang Saldo PROFIT

4.6.1 Ruang dengan tujuan

Kanal adalah ruang bernama dengan umpannya sendiri. Profitina memiliki kanal resmi yang dijalankan platform, dan kanal yang dibuat anggota: sebuah kelas, para pemilik usaha kecil di satu kota, sebuah topik seperti data terstruktur. Direktori kanal punya tiga tab, *Resmi*, *Milikku*, dan *Tren*, agar pendatang baru dapat menemukan ruang yang penting tanpa menggulir setiap percobaan yang pernah dibuat orang [1].

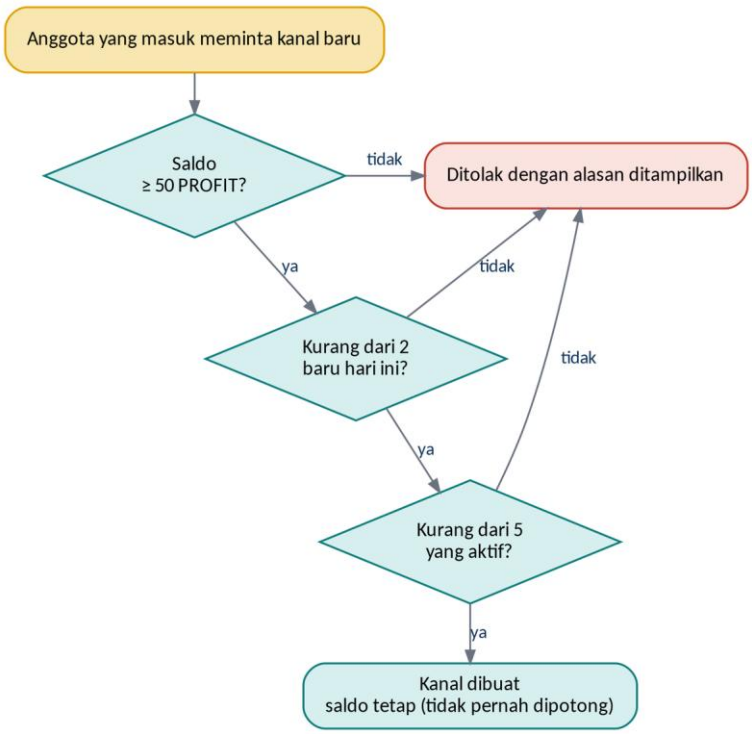
Setiap kanal yang benar-benar dipakai memerlukan seseorang yang menjaganya. Kanal kelas adalah contoh paling jelas: dosen menyematkan pengumuman dan menjaga ruang tetap pada topiknya. Moderator yang cakupannya satu kanal mengerjakan persis itu, hanya di dalam kanalnya sendiri, dan Bab 5 menjelaskan apa yang boleh dan tidak boleh mereka lakukan.

4.6.2 Gerbangnya

Siapa pun boleh membuat kanal, tetapi tidak tanpa batas. Seorang anggota hanya boleh membuat kanal bila ketiga syarat berikut semuanya terpenuhi:

$$\text{boleh} \Leftrightarrow (b \ge 50 \text{ PROFIT}) \wedge (d < 2) \wedge (a < 5), \tag{4.5}$$

dengan b saldo PROFIT anggota itu, d banyaknya kanal yang telah dibuatnya hari ini (hari berganti pukul 00.00 WIB, Waktu Indonesia Barat, UTC+7), dan a banyaknya kanal miliknya yang sedang aktif [1]. **Gambar 4.4** menggambar aturan yang sama sebagai diagram alir.



Gambar 4.4 Membuat kanal. Saldo sekurang-kurangnya 50 PROFIT diperlukan tetapi tidak pernah dipotong; paling banyak dua kanal baru per hari dan lima kanal aktif per anggota.

Kata-kata terpenting pada gambar itu adalah *tidak pernah dipotong*. Saldo adalah *gerbang*, bukan *biaya*. Anggota dengan 50 PROFIT yang membuat kanal tetap memiliki 50 PROFIT sesudahnya. Mengapa dirancang demikian dan tidak menarik bayaran untuk kanal?

- **Biaya berarti menjual visibilitas.** Jika kanal berharga PROFIT, orang yang membeli PROFIT dapat membeli ruang, dan direktori akan condong ke siapa pun yang membelanjakan paling banyak. Gerbang saldo hanya menyatakan "Anda sudah cukup berpartisipasi untuk memperoleh atau memiliki sebanyak ini"; tidak ada yang dipindahkan.
- **Gerbang memperlambat spam tanpa menghukum siapa pun.** Pengirim spam membuat banyak ruang dengan cepat; batas harian dan batas aktif menghentikannya, dan syarat saldo berarti akun sekali pakai yang baru dibuat sama sekali tidak dapat membuat ruang.
- **Mudah dijelaskan.** Satu kalimat ("pegang sekurang-kurangnya 50 PROFIT; tidak ada yang diambil") adalah aturan yang dapat diperkirakan anggota, dan itu penting ketika aturan ditegakkan perangkat lunak, bukan oleh orang yang dapat menjelaskan dirinya.

> **CONTOH.** Sari memegang 62 PROFIT, pagi ini membuat satu kanal, dan memiliki tiga kanal aktif. Menurut Persamaan (4.5): $62 \geq 50$ terpenuhi, $1 < 2$ terpenuhi, dan $3 < 5$ terpenuhi, jadi ia boleh membuat kanal kedua hari ini. Sesudahnya ia tetap memegang 62 PROFIT, telah membuat $d = 2$ kanal hari ini, dan memiliki $a = 4$ kanal aktif. Percobaan ketiga pada hari yang sama gagal di $d < 2$; pesan penolakannya menyebutkan hal itu, dan penghitungnya berganti pukul 00.00 WIB.

PROFIT sendiri, status *devnet*-nya, serta apa yang boleh dan tidak boleh dilakukan dengannya menjadi pokok bahasan Bab 12. Tidak ada satu pun dalam bab ini yang merupakan pernyataan tentang nilainya; gerbang hanya memakainya sebagai hitungan.

4.6.3 Papan peringkat per kanal

Setiap kanal juga memiliki papan peringkatnya sendiri, gratis bagi setiap anggota, di samping tiga belas papan peringkat global platform dan papan rujukan [1]. Bagi kanal kelas, inilah tempat yang

wajar untuk melihat siapa yang paling banyak menyumbang bahan berguna di ruang itu, tanpa membandingkan kelas tahun pertama dengan seluruh platform.

4.7 Mengikuti Kanal dan Penghitung “N baru”

4.7.1 Apa yang ingin diketahui pengikut

Anggota yang mengikuti beberapa kanal ingin tahu, dalam sekali lihat, *di mana ada sesuatu yang baru*. Profitina menampilkan lencana merah kecil di samping setiap kanal yang diikuti: “3 baru”. Menentukan apa yang harus dihitung angka itu ternyata memerlukan pemikiran lebih dari yang diduga.

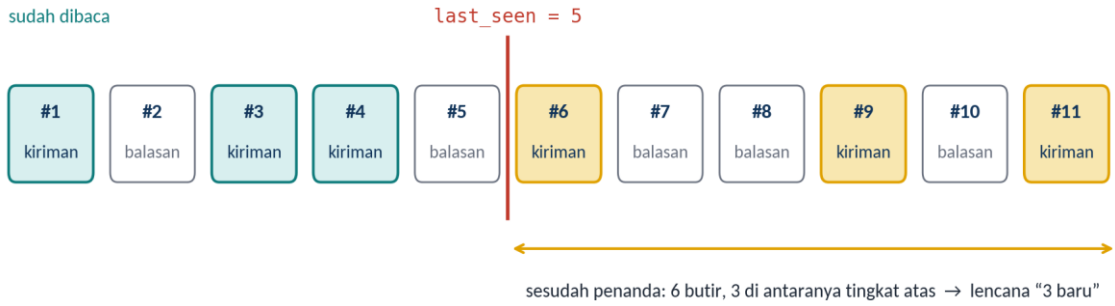
4.7.2 Definisinya

Untuk setiap anggota dan setiap kanal yang diikuti c, platform mengingat satu bilangan s_c, yaitu *penanda terakhir-dilihat*: nomor kiriman terbaru di c yang sudah dilihat anggota itu. Nomor kiriman selalu bertambah, sehingga setiap kiriman dengan nomor lebih besar lebih baru. Misalkan P_c himpunan kiriman di c. Lencana menampilkan

$$N_{\text{baru}}(c) = \left|\{\, p \in P_c : p.\text{id} > s_c \ \wedge\ p.\text{reply_to} = \text{null} \,\}\right|, \tag{4.6}$$

yaitu banyaknya kiriman *tingkat atas* yang lebih baru daripada penanda [1]. Membuka kanal menyetel s_c ke nomor kiriman terbaru, dan lencana pun hilang.

Gambar 4.5 mengerjakan satu contoh. Penanda berdiri sesudah kiriman 5. Enam butir menyusul, tetapi tiga di antaranya balasan, sehingga lencana bertuliskan “3 baru”, bukan “6 baru”.



Gambar 4.5 Penghitung “N baru”. Hanya kiriman tingkat atas sesudah penanda terakhir-dilihat yang dihitung, sehingga enam butir baru yang tiga di antaranya balasan menghasilkan lencana “3 baru”.

Mengapa balasan tidak dihitung? Karena lencana adalah janji tentang apa yang akan ditemukan anggota. “6 baru” menyiratkan enam percakapan baru; membuka kanal akan menampilkan tiga kiriman baru di atas dan tiga balasan yang terkubur di dalam utas yang mungkin tidak dipedulikan anggota itu. Hanya menghitung kiriman tingkat atas menepati janji: setiap butir yang dihitung terlihat di bagian atas umpan. Balasan tidak hilang; anggota yang menulis kiriman induknya mendapat notifikasi untuknya di dasbor (Subbab 4.9).

4.7.3 Satu kata, satu angka

Rilis September 2026 juga memberi pelajaran kecil tentang penghitung. Pemilih kanal menampilkan “6 kiriman” untuk sebuah kanal sementara kartu di bagian lain halaman bertuliskan “kanal ini: 5 kiriman”; satu hitungan memasukkan repost dan yang lain tidak. Kedua definisi sama-sama dapat dipertahankan, tetapi pengguna yang melihat kata yang sama di samping dua angka berbeda menyimpulkan bahwa salah satunya rusak. Perbaikannya adalah

membuat setiap tempat yang menulis *kiriman* untuk sebuah kanal memakai definisi yang sama, dan memberi nama lain untuk hitungan berbeda yang dipakai bagi hadiah dan peringkat (hanya kiriman asli). Aturan ini berlaku umum: **kata yang sama di layar yang sama harus berarti angka yang sama.**

4.7.4 Menjaga penghitung tetap hidup

Lencana harus diperbarui selagi halaman terbuka. Di sinilah peristiwa tipis dari Subbab 4.4.5 menunjukkan gunanya. Setiap peristiwa membawa kanal dan penanda `top`. Ketika peristiwa datang untuk kanal yang *tidak* sedang tampil, halaman tidak mengambil apa pun; ia menambah satu pada lencana kanal itu bila `top` bernilai benar, dan tidak melakukan apa-apa bila tidak. Itulah Persamaan (4.6) yang dipelihara secara bertahap, satu peristiwa demi satu peristiwa, tanpa biaya jaringan. Halaman peramban lab melakukan persis hal ini, dan pemeriksaan peramban otomatis di Subbab 4.12.6 memastikan bahwa balasan tidak menaikkan lencana.

4.8 Pesan Langsung dan Profil: Privat secara Bawaan

4.8.1 Pesan langsung

Sejak 20 Agustus 2026 anggota dapat saling mengirim *pesan langsung* (*direct message*, DM) [1]. Rancangannya sengaja dibuat sempit, dan setiap pembatasan menutup satu masalah yang sudah dikenal di platform yang lebih besar. **Tabel 4.3** mendaftar pembatasan itu beserta alasannya.

Tabel 4.3 Aturan pesan langsung dan alasan masing-masing.

Aturan	Alasan
Hanya satu lawan satu; tanpa grup, tanpa @mention	Grup dan sebutan adalah jalan menyebarnya spam dan keroyokan; diskusi publik tempatnya di kanal
Tidak ada PROFIT untuk DM	Hadiah untuk pesan privat tidak dapat diperiksa siapa pun dan akan mengundang “ternak” hadiah
Hanya notifikasi dalam aplikasi, lencana belum dibaca	Tidak ada jalur surel atau telepon yang bisa bocor; dasbor menampilkan DM baru sebagai mendesak
Blokir dan lapor seketika	Orang yang diganggu bertindak saat itu juga, tanpa menunggu siapa pun
Administrator tidak dapat membaca DM	Privat berarti privat, termasuk dari operator
Pengecualian: salinan beku pesan yang *dilaporkan*	Pelaporan harus mungkin; hanya pesan yang dilaporkan, sebagaimana adanya saat dilaporkan, yang terlihat oleh peninjau

Dua baris terakhir patut dicermati, karena bersama-sama keduanya memecahkan dilema nyata. Jika tidak seorang pun dapat membaca DM, pelecehan lewat DM tidak akan pernah dapat dibuktikan. Jika operator dapat membaca setiap DM, DM tidak privat. Jawaban Profitina adalah menjadikan tindakan melapor sebagai *satu-satunya* hal yang membuka sebuah pesan, dan yang dibuka adalah *salinan beku*: pesan sebagaimana adanya saat dilaporkan, yang tidak dapat lagi disunting atau dihapus pengirimnya untuk menyembunyikan apa yang telah dikatakan. Tidak ada bagian lain dari percakapan itu yang menjadi terlihat.

4.8.2 Profil dengan persetujuan per butir

Halaman *profil* anggota dapat menampilkan foto dan paling banyak empat tautan opsional ke akunnya di platform sosial lain. Masing-masing dari kelima butir itu punya sakelar persetujuannya sendiri, setiap sakelar dimulai dalam keadaan *mati*, dan masing-masing dapat dimatikan lagi kapan saja [1]. Alamat surel tidak pernah dapat dipublikasikan, dan sama sekali

tidak ada kolom nomor telepon. Aturan-aturan ini mengikuti langsung Undang-Undang Pelindungan Data Pribadi, UU PDP No. 27/2022, yang prinsip pembatasan tujuan dan minimisasinya dibahas mendalam di Bab 11: data yang tidak diperlukan sebuah fitur tidak boleh dikumpulkan, dan data yang dikumpulkan hanya boleh ditampilkan dengan persetujuan yang spesifik [7], [8].

4.8.3 Handle netral

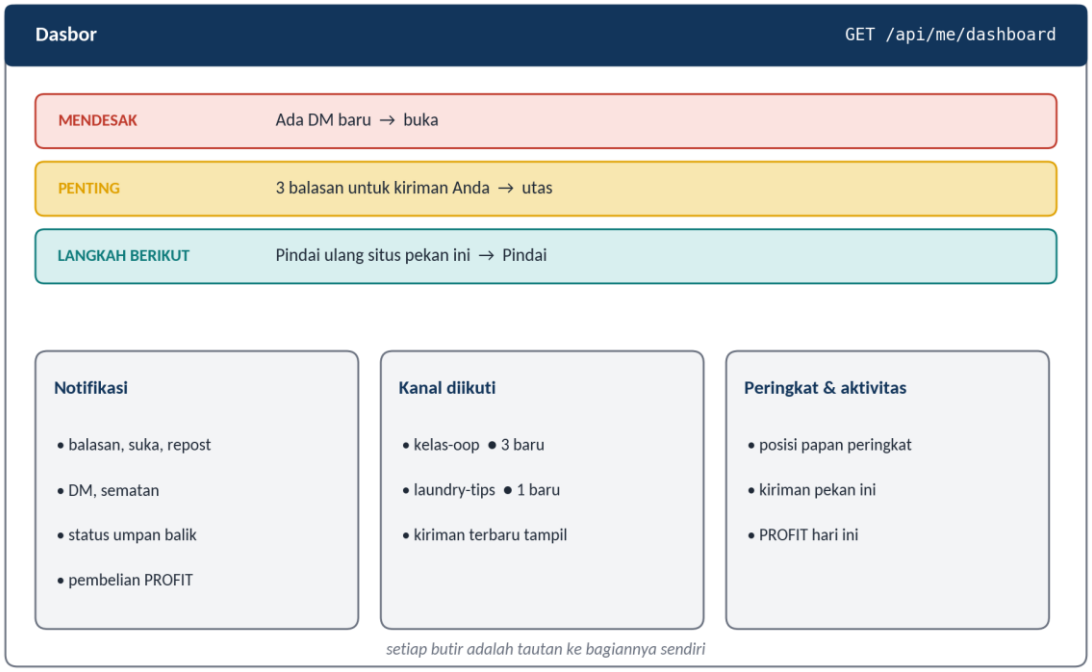
Setiap anggota dikenali di komunitas melalui sebuah *handle*. Ketika sebuah kelas universitas bergabung, satu masalah langsung muncul: *handle* yang paling mudah dipilih mahasiswa adalah nomor induknya, dan nomor induk mahasiswa adalah pengenal pribadi yang mengaitkan kiriman dengan catatan resmi. Sejak 14 Agustus 2026 anggota baru menerima *handle netral* acak berbentuk `u` diikuti delapan digit heksadesimal, misalnya `u3f9a1c20`; anggota boleh menggantinya, dan platform memeriksa bahwa *handle* pilihan tidak tampak seperti nomor induk mahasiswa [1]. Pengunjung anonim melihat nama dalam keadaan tersamar. Lab di Subbab 4.12 menegakkan pola *handle* yang sama, dan salah satu kiriman ujinya ditolak karena melanggarnya.

> **WAWASAN KUNCI.** Privasi secara bawaan sebagian besar soal *nilai awal*. Halaman profil yang sama, dengan lima sakelar yang sama, menjadi privat atau publik semata-mata bergantung pada apakah sakelarnya dimulai dalam keadaan mati atau hidup. Kebanyakan orang tidak pernah mengubah bawaan, jadi bawaan *adalah* kebijakannya.

4.9 Dasbor sebagai Kokpit

Pilot tidak berkeliling pesawat untuk memeriksa setiap instrumen; kokpit menghadirkan setiap instrumen di depan satu kursi, disusun menurut tingkat kedesakan. Sejak pertengahan September 2026, dasbor Profitina mengikuti gagasan yang sama [1].

Satu permintaan, `GET /api/me/dashboard`, mengumpulkan segala hal yang mungkin memerlukan perhatian anggota: notifikasi (balasan, suka, dan repost atas kirimannya, DM baru, pengumuman yang disematkan di kanal-kanalnya, status umpan balik yang dikirimnya, dan pembelian PROFIT yang telah selesai), kanal-kanal yang diikutinya beserta kiriman terbaru dan lencana “N baru” masing-masing, serta peringkat dan aktivitas terakhirnya. Di bagian atas ada *pita prioritas* dengan tiga tingkat: **mendesak** (misalnya ada DM yang menunggu), **penting** (misalnya balasan untuk kiriman Anda), dan **langkah berikut** (panduan seperti “pindai ulang situs Anda pekan ini”). Setiap butir adalah tautan ke bagiannya sendiri di aplikasi. **Gambar 4.6** adalah kerangka rancangan (*wireframe*) susunannya.



Gambar 4.6 Kerangka rancangan kokpit dasbor. Satu permintaan mengisi pita prioritas tiga tingkat dan tiga kartu; setiap butir menaut ke bagiannya sendiri.

Mengapa satu permintaan, bukan satu per kartu? Karena setiap permintaan punya biaya tetap (sambungan, autentikasi, perjalanan pulang-pergi melintasi internet), dan dasbor yang membuat sepuluh permintaan membayarnya sepuluh kali dan muncul berkedip sepotong demi sepotong. Satu permintaan yang mengembalikan segalanya memungkinkan halaman menggambar dirinya sekali. Satu permintaan juga memudahkan penegakan aturan di server: dasbor dirakit dari data yang telah diambil lebih dulu dalam sejumlah kecil kueri basis data yang tetap. Pada rilis September 2026, setelah perubahan yang dijelaskan di Subbab 4.11.3, jawaban dasbor memerlukan sekitar 15 ms, bukan 145 ms.

Satu rincian kecil tetapi bermakna menyangkut kartu kanal yang diikuti. Semula kartu itu menampilkan pengumuman yang *disematkan* di sebuah kanal; anggota meminta kiriman *terbaru* sebagai gantinya, siapa pun pengirimnya, termasuk dirinya sendiri. Kini kartu menampilkan kiriman terbaru dan menandainya "disematkan" hanya bila kiriman terbaru itu kebetulan yang disematkan. Dasbor hanya berguna bila menjawab pertanyaan yang benar-benar diajukan penggunanya.

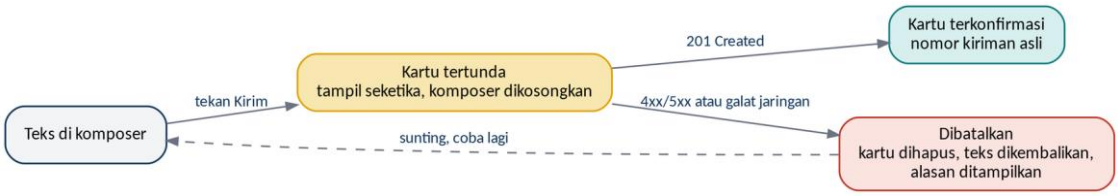
4.10 Pengiriman Seketika (Optimistis)

4.10.1 Menunggu terasa lebih lama daripada sebenarnya

Perhatian manusia punya ambang yang sudah dikenal. Sekitar 0,1 detik adalah batas agar suatu tindakan terasa seketika; sekitar 1 detik adalah batas agar alur pikiran tidak terputus; sekitar 10 detik adalah batas untuk tetap memusatkan perhatian pada tugas [9]. Sebuah kiriman harus menempuh perjalanan ke server, diperiksa (termasuk oleh lapisan keamanan Bab 5), disimpan, dan diakui. Pada hari yang baik itu beberapa ratus milidetik; pada sambungan seluler yang lambat, beberapa detik. Jika kotak penyusun (*composer*) sekadar menunggu, anggota menatap kotak yang membeku dan bertanya-tanya apakah ada yang terjadi.

4.10.2 Tampilkan dulu, pastikan kemudian

Pengiriman *optimistis* membalik urutannya. Begitu anggota menekan *Kirim*, halaman mengosongkan kotak penyusun dan menggambar kiriman di bagian atas umpan, bertanda tertunda. Baru setelah itu permintaan dikirim. **Gambar 4.7** menunjukkan ketiga keadaannya.



Gambar 4.7 Riwayat hidup kiriman optimistis. Kiriman langsung tampil sebagai tertunda; keberhasilan mengonfirmasinya dengan nomor aslinya, dan kegagalan menghapusnya serta mengembalikan teks beserta alasannya.

- Jika server menjawab *201 Created*, kartu tertunda menjadi kartu biasa dengan nomor kiriman yang asli.
- Jika server menolak (misalnya karena kiriman terlalu panjang, atau sebuah tautan di dalamnya diblokir) atau jaringan gagal, halaman **menghapus kartu itu dan mengembalikan teksnya ke kotak penyusun**, dengan alasan ditampilkan. Tidak ada yang diketik anggota yang hilang.

Optimisme ini aman karena hanya server yang memutuskan: kartu tertunda adalah ramalan, dan ramalan itu dikoreksi ke arah mana pun dalam satu perjalanan pulang-pergi. Profitina mengirim dengan cara ini sejak pertengahan September 2026; foto dapat ditempel dari papan klip atau diseret ke kotak penyusun dan menempuh jalur yang sama [1].

Ada satu kehalusan yang mudah terlewat. Aliran langsung Subbab 4.4 juga akan mengabarkan kiriman anggota kepada halaman anggota itu sendiri. Jika halaman tidak mengenalinya, kiriman muncul dua kali: sekali sebagai kartu optimistis dan sekali sebagai “kabar”. Lab memecahkan hal ini dengan mengingat nomor-nomor kiriman yang telah dikonfirmasinya dan mengabaikan peristiwa untuk nomor-nomor itu, serta dengan menghapus kartu tertunda bila peristiwanya kebetulan tiba sebelum konfirmasi.

4.11 Ketika Waktu-Nyata Keliru: Tiga Pelajaran Rekayasa

Sistem waktu-nyata gagal dengan cara-cara yang tidak pernah dialami halaman tanya-jawab sederhana. Rilis September 2026 memperbaiki tiga kegagalan seperti itu setelah anggota melaporkan bahwa berpindah kanal di komunitas “sangat lambat”. Semua angka di bawah ini diukur pada kode platform itu sendiri dengan basis data PostgreSQL 16 selama rilis tersebut, untuk satu halaman umpan berisi 30 kiriman.

4.11.1 Pelajaran 1: jalur baca tidak boleh memanggil model

Penyaringan tautan Profitina (Bab 5) memiliki lapisan kedua opsional, tempat sebuah model AI lokal kecil memberi pendapat tentang tautan yang mencurigakan. Sebelum perbaikan, fungsi yang *menampilkan* setiap kiriman, yang berjalan setiap kali seseorang membuka umpan, dapat menanyai model itu tentang tautan mencurigakan di kiriman bila belum ada putusan yang tersimpan. Pada umpan yang sibuk itu berarti panggilan model selagi seorang anggota menunggu halamannya, enam kali pada satu halaman uji.

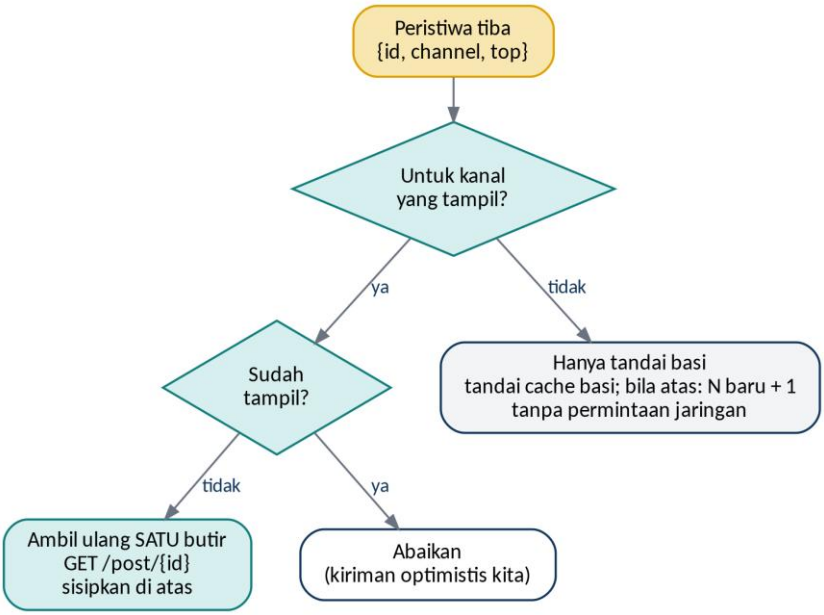
Perbaikannya berupa aturan, bukan tambalan: **fungsi yang berjalan sekali per butir saat menampilkan tidak boleh memanggil model dan tidak boleh membuka sambungan basis**

datanya sendiri. Pendapat model diminta sekali, di jalur *tulis*, ketika kiriman disimpan, dan putusannya disimpan bersama kiriman. Membaca hanya membaca.

4.11.2 Pelajaran 2: tandai basi, jangan ambil ulang semuanya

Kegagalan kedua berupa badai. Setiap peristiwa di aliran langsung membuat halaman memuat ulang umpan yang sedang ditampilkannya, bahkan ketika peristiwa itu menyangkut kanal lain. Dengan banyak anggota daring, satu kiriman di satu kanal menyebabkan pemuatan ulang umpan penuh di setiap halaman yang terbuka, dan setiap pemuatan ulang menambah pekerjaan basis data.

Obatnya adalah perbedaan yang diperkenalkan di Subbab 4.4.5, digambar sebagai keputusan pada **Gambar 4.8**. Peristiwa untuk kanal yang *tidak* sedang tampil hanya *menandai basi*: halaman menandai salinan tersimpan kanal itu sebagai basi dan, bila peristiwanya tingkat atas, menambah satu pada lencana, tanpa permintaan jaringan sama sekali. Peristiwa untuk kanal yang *sedang* tampil menyebabkan *pengambilan ulang tepat satu butir*, yaitu kiriman baru itu, yang disisipkan di bagian atas.



Gambar 4.8 Apa yang dilakukan halaman terhadap peristiwa tipis. Peristiwa untuk kanal lain hanya menandai basi *cache* dan memperbarui penghitung; peristiwa untuk kanal yang tampil mengambil satu kiriman; peristiwa untuk kiriman optimistis milik anggota sendiri diabaikan.

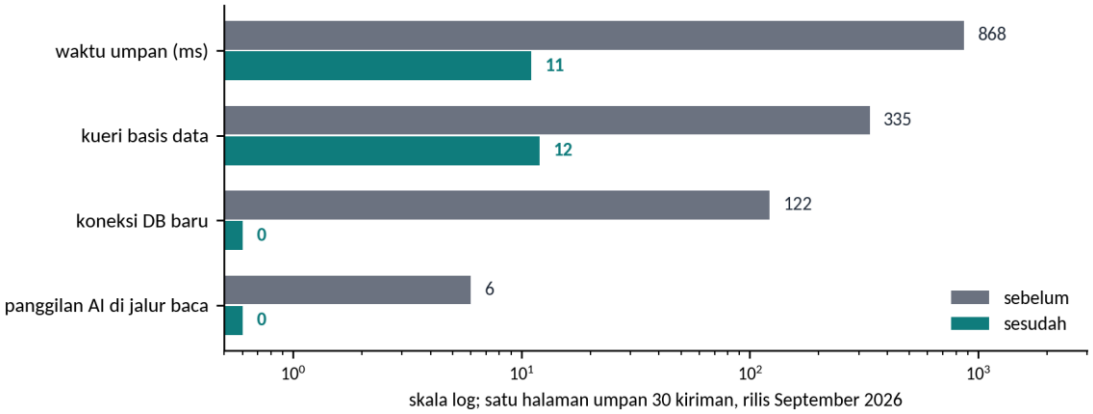
4.11.3 Pelajaran 3: hitung kueri Anda

Kegagalan ketiga adalah *masalah kueri N+1* yang klasik. Untuk menampilkan satu halaman berisi n kiriman, kode menjalankan satu kueri untuk mendaftar kiriman lalu, untuk setiap kiriman, sekitar sebelas kueri lagi: nama tampilan pengirimnya, jumlah suka, apakah penonton menyukainya, kiriman induknya, nama kanalnya, dan seterusnya. Banyaknya kueri tumbuh linear terhadap ukuran halaman:

$$Q_{\text{sebelum}}(n) \approx 1 + 11n, \qquad Q_{\text{sesudah}}(n) = \text{konstan}. \tag{4.7}$$

Untuk $n = 30$, Persamaan (4.7) meramalkan $1 + 330 = 331$ kueri sebelum perbaikan; pengukuran mendapati 335. Lebih buruk lagi, setiap kueri membuka sambungan basis data *baru*, 122 sambungan untuk satu halaman. Perbaikannya terdiri atas dua bagian: *kolam sambungan*

(*connection pool*) di dalam proses, yang menjaga beberapa sambungan tetap terbuka dan meminjamkannya, dan *pengambilan awal* (*prefetching*), yang memuat nama, hitungan, dan induk untuk ke-30 kiriman dalam segelintir kueri sebelum satu pun ditampilkan. **Gambar 4.9** menunjukkan hasilnya.

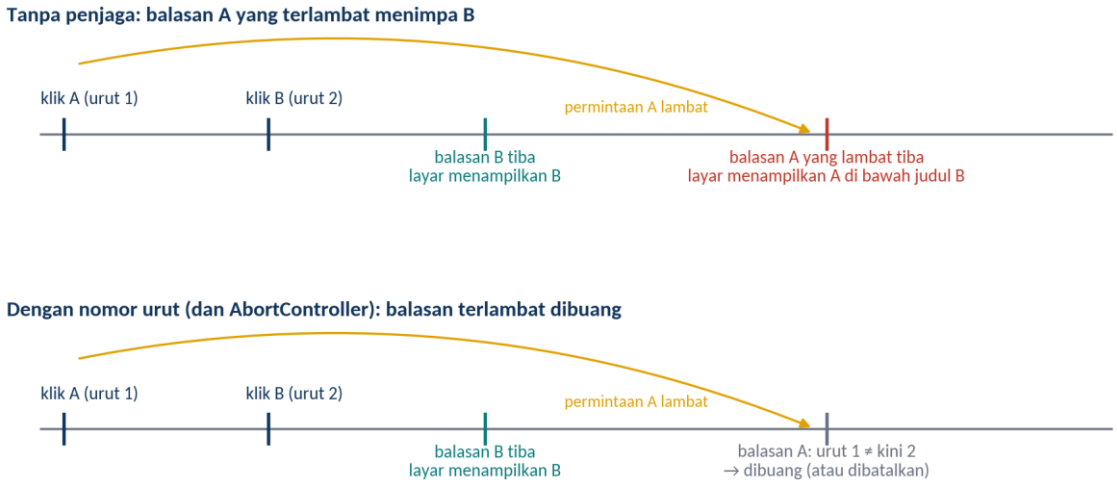


Gambar 4.9 Satu halaman umpan berisi 30 kiriman sebelum dan sesudah perbaikan September 2026 (skala log; nilai nol digambar sebagai potongan pendek). Waktu turun dari 868 ms menjadi 11 ms dan kueri dari 335 menjadi 12.

Pendekatan yang sama membawa widget tersemat di profitina.com dari 264 kueri menjadi 8 (sekitar 100 ms menjadi sekitar 10 ms), dan di peramban Chromium sungguhan 30 kartu umpan tampil 143 ms setelah pindah kanal.

4.11.4 Pelajaran 4 (bonus): jawaban yang terlambat

Ketika pindah kanal menjadi cepat, sebuah *bug* yang lebih halus menjadi terlihat. Seorang anggota mengeklik kanal A dan, sebelum jawaban A tiba, mengeklik kanal B. B menjawab dengan cepat dan layar menampilkan B. Lalu jawaban A yang lambat tiba dan halaman, yang tidak tahu bahwa jawaban itu terlambat, menggambar kiriman A di bawah judul B. **Gambar 4.10** menunjukkan balapan itu dan obatnya.



Gambar 4.10 Balapan jawaban basi. Tanpa penjaga, balasan lambat untuk kanal A menimpa kanal B; dengan nomor urut permintaan, balasan terlambat dikenali dan dibuang.

Obatnya terdiri atas dua bagian. Setiap permintaan membawa *nomor urut*; ketika jawaban tiba, halaman membandingkan nomornya dengan yang terakhir dan membuang jawaban itu bila lebih tua. Dan permintaan yang lebih tua *dibatalkan* dengan `AbortController` milik peramban [10], sehingga berhenti memakai jaringan. Rilis yang memperbaiki umpan pada awalnya hanya

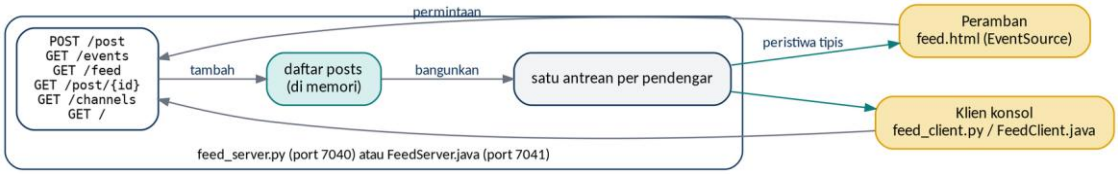
menjaga umpan itu sendiri; pita pengumuman yang disematkan dan papan peringkat kanal, yang dimuat oleh permintaan terpisah, masih mengalami balapan yang sama sampai keduanya juga dijaga. Pelajarannya: **setiap panel yang memuat data per kanal memerlukan pemeriksaan "apakah ini masih kanal yang aktif?" sendiri.**

4.12 Lab G2: Umpan Langsung dengan Server-Sent Events

4.12.1 Apa yang akan kita bangun

Lab ini membangun, untuk Profitina Laundry, umpan komunitas dengan tiga kanal (*general*, *tips*, dan *promo*) yang diperbarui secara langsung. Ada lima bagian, digambar pada **Gambar 4.11**:

- sebuah **server**, sekali dalam Python (`feed_server.py`, port 7040) dan sekali dalam Java (`FeedServer.java`, port 7041), yang menyimpan kiriman di memori, menerima kiriman baru, dan mengalirkan peristiwa tipis;
- sebuah **halaman peramban**, `feed.html`, dipakai bersama kedua server, dengan tab kanal, lencana "N baru", pengiriman optimistis, dan penjaga balasan basi;
- sebuah **klien konsol**, lagi-lagi dalam Python dan Java, yang mendengarkan aliran atau mengirim kiriman;
- **pembantu JSON** kecil untuk Java, karena pustaka standar Java tidak memiliki pengurai JSON;
- sebuah **pemeriksaan otomatis** yang menjalankan halaman di Chromium tanpa layar.



Gambar 4.11 Lab ini. Kedua server menyimpan daftar kiriman dan satu antrean per pendengar; peramban dan klien konsol mengirim permintaan biasa dan menerima peristiwa tipis lewat satu aliran yang terbuka.

Semuanya hanya memakai pustaka standar: `http.server`, `urllib`, dan `json` di Python [11], serta `com.sun.net.httpserver` dan `java.net.http` di Java [12], [13]. Kodenya ada di folder kode buku di bawah `Chapter04/`, dengan `README.md` yang memberikan perintah untuk Windows, macOS, dan Linux. **Tabel 4.4** mendaftar antarmuka server.

Tabel 4.4 Antarmuka server lab (sama persis dalam Python dan Java).

Permintaan	Jawaban
`GET /`	Halaman peramban `feed.html`
`GET /channels`	`["general","tips","promo"]`
`GET /events`	Aliran peristiwa; menghormati `Last-Event-ID`
`GET /feed?channel=c&delay=ms`	Kiriman kanal `c`, terbaru lebih dulu; `delay` meniru jawaban yang lambat
`GET /post/n`	Kiriman nomor `n`
`POST /post`	Badan `{"channel","handle","text","reply_to"}`; menjawab 201 dengan nomor baru, atau 422 dengan alasannya

4.12.2 Server Python

Kode 4.1 adalah seluruh server Python. Bacalah dalam empat bagian.

Kode 4.1 `feed_server.py`: server umpan langsung dengan Server-Sent Events, hanya pustaka standar.

```
"""A tiny live-feed server with Server-Sent Events (standard library only).

Run:  python feed_server.py [--port 7040] [--heartbeat 15]
Then open http://127.0.0.1:7040/ in a browser.
"""
import argparse
import json
import queue
import re
import threading
import time
from http.server import BaseHTTPRequestHandler, ThreadingHTTPServer
from pathlib import Path
from urllib.parse import parse_qs, urlparse

CHANNELS = ["general", "tips", "promo"]
HANDLE = re.compile(r"^u[0-9a-f]{8}$")     # neutral handles such as u3f9a1c20
MAX_TEXT = 280
PAGE = Path(__file__).resolve().parent.parent / "web" / "feed.html"


class Hub:
    """Keeps every post in memory and one queue per connected listener."""

    def __init__(self):
        self.lock = threading.Lock()
        self.posts = []                      # post number n is posts[n - 1]
        self.listeners = []

    def add_post(self, channel, handle, text, reply_to):
        with self.lock:
            post = {"id": len(self.posts) + 1, "channel": channel,
                    "handle": handle, "text": text, "reply_to": reply_to}
            self.posts.append(post)
            for q in self.listeners:
                q.put(post)                  # wake every listener
        return post

    def get(self, post_id):
        with self.lock:
            if 1 <= post_id <= len(self.posts):
                return self.posts[post_id - 1]
        return None
```

Kode 4.1 (Lanjutan) *feed_server.py: server umpan langsung dengan Server-Sent Events, hanya pustaka standar.*

```
    def channel_posts(self, channel, limit=50):
        with self.lock:
            found = [p for p in self.posts if p["channel"] == channel]
        return list(reversed(found[-limit:]))   # newest first

    def subscribe(self, last_id):
        """Register a listener; return its queue and the posts it missed."""
        q = queue.Queue()
        with self.lock:
            self.listeners.append(q)
            missed = self.posts[last_id:]
            count = len(self.listeners)
        return q, missed, count

    def unsubscribe(self, q):
        with self.lock:
            self.listeners.remove(q)
            return len(self.listeners)


def thin_event(post):
    """A small notification: which post, which channel, top-level or not."""
    data = {"id": post["id"], "channel": post["channel"],
            "top": post["reply_to"] is None}
    return "id: %d\nevent: post\ndata: %s\n\n" % (
        post["id"], json.dumps(data, separators=(",", ":")))


def validate(body):
    """Return (post fields, None) or (None, error message)."""
    channel = body.get("channel")
    handle = body.get("handle")
    text = body.get("text")
    reply_to = body.get("reply_to")
    if channel not in CHANNELS:
        return None, "unknown channel"
    if not isinstance(handle, str) or not HANDLE.match(handle):
        return None, "handle must be neutral, like u3f9a1c20"
    if not isinstance(text, str) or not text.strip():
        return None, "empty post"
    if len(text) > MAX_TEXT:
        return None, "post longer than %d characters" % MAX_TEXT
    if reply_to is not None and not isinstance(reply_to, int):
        return None, "reply_to must be a post number"
    return (channel, handle, text.strip(), reply_to), None
```

Kode 4.1 (Lanjutan) *feed_server.py: server umpan langsung dengan Server-Sent Events, hanya pustaka standar.*

```
class Handler(BaseHTTPRequestHandler):
    hub = None                                  # set in main()
    heartbeat = 15.0

    def log_message(self, fmt, *args):          # keep the console quiet
        pass

    def send_json(self, status, obj):
        raw = json.dumps(obj, separators=(",", ":")).encode("utf-8")
        self.send_response(status)
        self.send_header("Content-Type", "application/json")
        self.send_header("Content-Length", str(len(raw)))
        self.end_headers()
        self.wfile.write(raw)

    def do_GET(self):
        url = urlparse(self.path)
        query = parse_qs(url.query)
        if url.path == "/":
            raw = PAGE.read_bytes()
            self.send_response(200)
            self.send_header("Content-Type", "text/html; charset=utf-8")
            self.send_header("Content-Length", str(len(raw)))
            self.end_headers()
            self.wfile.write(raw)
        elif url.path == "/events":
            self.stream_events()
        elif url.path == "/channels":
            self.send_json(200, CHANNELS)
        elif url.path == "/feed":
            channel = query.get("channel", ["general"])[0]
            delay = min(int(query.get("delay", ["0"])[0]), 5000)
            time.sleep(delay / 1000)            # lets you simulate a slow reply
            self.send_json(200, {"channel": channel,
                                 "posts": self.hub.channel_posts(channel)})
        elif url.path.startswith("/post/"):
            post = self.hub.get(int(url.path[6:] or 0))
            if post:
                self.send_json(200, post)
            else:
                self.send_json(404, {"error": "no such post"})
        else:
            self.send_json(404, {"error": "not found"})
```

Kode 4.1 (Lanjutan) *feed_server.py: server umpan langsung dengan Server-Sent Events, hanya pustaka standar.*

```
    def do_POST(self):
        if self.path != "/post":
            self.send_json(404, {"error": "not found"})
            return
        length = int(self.headers.get("Content-Length", 0))
        try:
            body = json.loads(self.rfile.read(length) or b"{}")
        except json.JSONDecodeError:
            self.send_json(400, {"error": "body is not JSON"})
            return
        fields, error = validate(body)
        if not error and fields[3] is not None and not self.hub.get(fields[3]):
            error = "no such post to reply to"
        if error:
            print("rejected:", error, flush=True)
            self.send_json(422, {"error": error})
            return
        post = self.hub.add_post(*fields)
        print("post #%d [%s] by %s" % (post["id"], post["channel"],
                                       post["handle"]), flush=True)
        self.send_json(201, {"id": post["id"]})

    def stream_events(self):
        last = self.headers.get("Last-Event-ID") or "0"
        last_id = int(last) if last.isdigit() else 0
        q, missed, count = self.hub.subscribe(last_id)
        print("listener connected, last id %d (%d open)" % (last_id, count),
              flush=True)
        self.send_response(200)
        self.send_header("Content-Type", "text/event-stream")
        self.send_header("Cache-Control", "no-cache")
        self.end_headers()
        try:
            self.wfile.write(b"retry: 3000\n\n")      # reconnect after 3 s
            for post in missed:                       # replay what was missed
                self.wfile.write(thin_event(post).encode("utf-8"))
            self.wfile.flush()
            while True:
                try:
                    post = q.get(timeout=self.heartbeat)
                    chunk = thin_event(post)
                except queue.Empty:
                    chunk = ": ping\n\n"              # comment line = heartbeat
                self.wfile.write(chunk.encode("utf-8"))
                self.wfile.flush()
        except (BrokenPipeError, ConnectionResetError):
```

Kode 4.1 (Lanjutan) *feed_server.py: server umpan langsung dengan Server-Sent Events, hanya pustaka standar.*

```
            pass
        finally:
            left = self.hub.unsubscribe(q)
            print("listener gone (%d open)" % left, flush=True)


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("--port", type=int, default=7040)
    parser.add_argument("--heartbeat", type=float, default=15.0)
    args = parser.parse_args()
    Handler.hub = Hub()
    Handler.heartbeat = args.heartbeat
    server = ThreadingHTTPServer(("127.0.0.1", args.port), Handler)
    server.daemon_threads = True
    print("feed server on http://127.0.0.1:%d/" % args.port, flush=True)
    try:
        server.serve_forever()
    except KeyboardInterrupt:
        print("stopped")


if __name__ == "__main__":
    main()
```

Hub. Kelas `Hub` menyimpan daftar kiriman dan daftar *pendengar*, satu `queue.Queue` per aliran yang terbuka. `add_post` menambahkan kiriman baru dan memasukkannya ke antrean setiap pendengar sambil memegang kunci (*lock*), sehingga seorang pendengar tidak akan pernah melewatkan kiriman yang ditambahkan di antara saat ia berlangganan dan pembacaan pertamanya. `subscribe` mendaftarkan sebuah antrean dan, dalam langkah terkunci yang sama, mengembalikan kiriman yang terlewat oleh pendengar: semua yang sesudah `last_id`. Satu kunci itulah yang membuat pemutaran ulang tepat.

Validasi. `validate` menolak kanal yang tidak dikenal, kiriman kosong, kiriman lebih dari 280 karakter, dan *handle* yang tidak cocok dengan pola netral `u` ditambah delapan digit heksadesimal dari Subbab 4.8.3. Hanya server yang memutuskan; kartu optimistis peramban hanyalah ramalan.

Permintaan biasa. `do_GET` dan `do_POST` menjawab permintaan-permintaan pada **Tabel 4.4**. Parameter `delay` pada `/feed` ada semata-mata agar penjaga balasan basi dapat diperagakan.

Aliran. `stream_events` adalah jantung lab. Fungsi ini membaca `Last-Event-ID`, berlangganan, mengirim *header* dengan tipe konten `text/event-stream`, menulis `retry: 3000`, memutar ulang yang terlewat, lalu berulang tanpa henti: menunggu antrean paling lama `heartbeat` detik; jika ada kiriman, tulis peristiwa tipisnya; jika tidak, tulis komentar `: ping`. Ketika pendengar menghilang, upaya menulis berikutnya memunculkan `BrokenPipeError` atau `ConnectionResetError`, perulangan berakhir, dan `finally` mencabut langganan antreannya. Karena `ThreadingHTTPServer` menjalankan setiap permintaan di utas (*thread*) tersendiri, aliran yang menunggu selamanya hanya menahan utasnya sendiri.

4.12.3 Server Java

Kode 4.2 mengerjakan hal yang sama dengan server HTTP bawaan JDK. Sebuah `record Post` menyimpan satu kiriman dan tahu cara menuliskan dirinya sebagai peta dan sebagai peristiwa tipis; para pendengar disimpan dalam `CopyOnWriteArrayList` berisi objek `BlockingQueue`, daftar yang aman ditelusuri selagi utas lain menambah atau menghapus antrean.

`sendResponseHeaders(200, 0)` dengan panjang 0 memberi tahu server bahwa panjangnya tidak diketahui, sehingga server mengalirkan badan tanggapan memakai pengodean transfer *chunked* HTTP/1.1 [14], persis yang dibutuhkan aliran peristiwa. `q.poll` dengan batas waktu memainkan peran `q.get(timeout=...)` di Python.

Kode 4.2 `FeedServer.java`: server yang sama dengan `com.sun.net.httpserver`, memakai satu antrean pemblokir per pendengar.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.CopyOnWriteArrayList;
import java.util.concurrent.Executors;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.TimeUnit;
import java.util.regex.Pattern;

/** A tiny live-feed server with Server-Sent Events (JDK only).
 *  Run: java -cp out FeedServer [--port 7041] [--heartbeat 15] */
public class FeedServer {
    static final List<String> CHANNELS = List.of("general", "tips", "promo");
    static final Pattern HANDLE = Pattern.compile("^u[0-9a-f]{8}$");
    static final int MAX_TEXT = 280;

    /** One post; replyTo is null for a top-level post. */
    record Post(int id, String channel, String handle, String text,
                Long replyTo) {
        Map<String, Object> toMap() {
            Map<String, Object> m = new LinkedHashMap<>();
            m.put("id", id);
            m.put("channel", channel);
            m.put("handle", handle);
            m.put("text", text);
            m.put("reply_to", replyTo);
            return m;
        }
```

Kode 4.2 (Lanjutan) *FeedServer.java: server yang sama dengan com.sun.net.httpserver, memakai satu antrean pemblokir per pendengar.*

```
        /** The small notification that travels over the stream. */
        String thinEvent() {
            Map<String, Object> m = new LinkedHashMap<>();
            m.put("id", id);
            m.put("channel", channel);
            m.put("top", replyTo == null);
            return "id: " + id + "\nevent: post\ndata: " + MiniJson.write(m)
                    + "\n\n";
        }
    }

    static final List<Post> posts = new ArrayList<>();       // guarded by posts
    static final List<BlockingQueue<Post>> listeners =
            new CopyOnWriteArrayList<>();
    static double heartbeat = 15.0;
    static Path page;

    static Post addPost(String channel, String handle, String text,
                        Long replyTo) {
        Post post;
        synchronized (posts) {
            post = new Post(posts.size() + 1, channel, handle, text, replyTo);
            posts.add(post);
        }
        for (BlockingQueue<Post> q : listeners) q.add(post);   // wake everyone
        return post;
    }

    static Post get(long id) {
        synchronized (posts) {
            if (id < 1 || id > posts.size()) return null;
            return posts.get((int) id - 1);
        }
    }

    static List<Object> channelPosts(String channel) {
        List<Object> out = new ArrayList<>();
        synchronized (posts) {
            for (int k = posts.size() - 1; k >= 0 && out.size() < 50; k--) {
                Post p = posts.get(k);
                if (p.channel().equals(channel)) out.add(p.toMap());
            }
        }
        return out;                                            // newest first
    }
```

Kode 4.2 (Lanjutan) *FeedServer.java: server yang sama dengan com.sun.net.httpserver, memakai satu antrean pemblokir per pendengar.*

```
    static void sendJson(HttpExchange ex, int status, Object obj)
            throws IOException {
        byte[] raw = MiniJson.write(obj).getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", "application/json");
        ex.sendResponseHeaders(status, raw.length);
        try (OutputStream out = ex.getResponseBody()) { out.write(raw); }
    }

    static Map<String, String> query(HttpExchange ex) {
        Map<String, String> q = new HashMap<>();
        String raw = ex.getRequestURI().getRawQuery();
        if (raw == null) return q;
        for (String pair : raw.split("&")) {
            int eq = pair.indexOf('=');
            if (eq > 0) q.put(pair.substring(0, eq), pair.substring(eq + 1));
        }
        return q;
    }

    static String validate(Map<String, Object> body) {
        Object channel = body.get("channel"), handle = body.get("handle");
        Object text = body.get("text"), replyTo = body.get("reply_to");
        if (!CHANNELS.contains(channel)) return "unknown channel";
        if (!(handle instanceof String a) || !HANDLE.matcher(a).matches())
            return "handle must be neutral, like u3f9a1c20";
        if (!(text instanceof String t) || t.isBlank()) return "empty post";
        if (t.length() > MAX_TEXT)
            return "post longer than " + MAX_TEXT + " characters";
        if (replyTo != null && !(replyTo instanceof Long))
            return "reply_to must be a post number";
        if (replyTo != null && get((Long) replyTo) == null)
            return "no such post to reply to";
        return null;
    }

    static void handle(HttpExchange ex) throws IOException {
        String path = ex.getRequestURI().getPath();
        if (ex.getRequestMethod().equals("POST")) {
            if (!path.equals("/post")) {
                sendJson(ex, 404, Map.of("error", "not found"));
                return;
            }
            Map<String, Object> body;
            try {
                byte[] raw = ex.getRequestBody().readAllBytes();
                body = MiniJson.parseObject(
```

Kode 4.2 (Lanjutan) *FeedServer.java: server yang sama dengan com.sun.net.httpserver, memakai satu antrean pemblokir per pendengar.*

```
                    new String(raw, StandardCharsets.UTF_8));
            } catch (RuntimeException e) {
                sendJson(ex, 400, Map.of("error", "body is not JSON"));
                return;
            }
            String error = validate(body);
            if (error != null) {
                System.out.println("rejected: " + error);
                sendJson(ex, 422, Map.of("error", error));
                return;
            }
            Post p = addPost((String) body.get("channel"),
                    (String) body.get("handle"),
                    ((String) body.get("text")).strip(),
                    (Long) body.get("reply_to"));
            System.out.printf("post #%d [%s] by %s%n",
                    p.id(), p.channel(), p.handle());
            sendJson(ex, 201, Map.of("id", p.id()));
            return;
        }
        Map<String, String> q = query(ex);
        switch (path) {
            case "/" -> {
                byte[] raw = Files.readAllBytes(page);
                ex.getResponseHeaders().set("Content-Type",
                        "text/html; charset=utf-8");
                ex.sendResponseHeaders(200, raw.length);
                try (OutputStream out = ex.getResponseBody()) { out.write(raw); }
            }
            case "/events" -> streamEvents(ex);
            case "/channels" -> sendJson(ex, 200, CHANNELS);
            case "/feed" -> {
                String channel = q.getOrDefault("channel", "general");
                int delay = Math.min(
                        Integer.parseInt(q.getOrDefault("delay", "0")), 5000);
                try {
                    Thread.sleep(delay);        // simulate a slow reply
                } catch (InterruptedException e) {
                    return;
                }
                Map<String, Object> m = new LinkedHashMap<>();
                m.put("channel", channel);
                m.put("posts", channelPosts(channel));
                sendJson(ex, 200, m);
            }
            default -> {
```

Kode 4.2 (Lanjutan) *FeedServer.java: server yang sama dengan com.sun.net.httpserver, memakai satu antrean pemblokir per pendengar.*

```
                Post p = null;
                if (path.startsWith("/post/") && path.length() > 6
                        && path.substring(6).chars().allMatch(Character::isDigit)) {
                    p = get(Long.parseLong(path.substring(6)));
                }
                if (p != null) sendJson(ex, 200, p.toMap());
                else sendJson(ex, 404, Map.of("error", path.startsWith("/post/")
                        ? "no such post" : "not found"));
            }
        }
    }

    static void streamEvents(HttpExchange ex) throws IOException {
        String last = ex.getRequestHeaders().getFirst("Last-Event-ID");
        int lastId = last != null && last.matches("\\d+")
                ? Integer.parseInt(last) : 0;
        BlockingQueue<Post> q = new LinkedBlockingQueue<>();
        List<Post> missed;
        synchronized (posts) {                  // subscribe and snapshot together
            listeners.add(q);
            int from = Math.min(lastId, posts.size());
            missed = new ArrayList<>(posts.subList(from, posts.size()));
        }
        System.out.printf("listener connected, last id %d (%d open)%n",
                lastId, listeners.size());
        ex.getResponseHeaders().set("Content-Type", "text/event-stream");
        ex.getResponseHeaders().set("Cache-Control", "no-cache");
        ex.sendResponseHeaders(200, 0);          // 0 = length unknown, streamed
        OutputStream out = ex.getResponseBody();
        try {
            out.write("retry: 3000\n\n".getBytes(StandardCharsets.UTF_8));
            for (Post p : missed) {                 // replay what was missed
                out.write(p.thinEvent().getBytes(StandardCharsets.UTF_8));
            }
            out.flush();
            while (true) {
                Post p = q.poll((long) (heartbeat * 1000),
                                TimeUnit.MILLISECONDS);
                String chunk = p != null ? p.thinEvent() : ": ping\n\n";
                out.write(chunk.getBytes(StandardCharsets.UTF_8));
                out.flush();
            }
        } catch (IOException | InterruptedException gone) {
            // the listener closed the connection
        } finally {
            listeners.remove(q);
```

Kode 4.2 (Lanjutan) *FeedServer.java: server yang sama dengan com.sun.net.httpserver, memakai satu antrean pemblokir per pendengar.*

```
            System.out.printf("listener gone (%d open)%n", listeners.size());
            ex.close();
        }
    }

    public static void main(String[] args) throws IOException {
        int port = 7041;
        for (int k = 0; k + 1 < args.length; k += 2) {
            if (args[k].equals("--port")) port = Integer.parseInt(args[k + 1]);
            if (args[k].equals("--heartbeat"))
                heartbeat = Double.parseDouble(args[k + 1]);
        }
        page = Files.exists(Path.of("web/feed.html"))
                ? Path.of("web/feed.html") : Path.of("../web/feed.html");
        HttpServer server =
                HttpServer.create(new InetSocketAddress("127.0.0.1", port), 0);
        server.createContext("/", ex -> {
            try { handle(ex); } catch (IOException e) { ex.close(); }
        });
        server.setExecutor(Executors.newCachedThreadPool()); // thread per stream
        server.start();
        System.out.println("feed server on http://127.0.0.1:" + port + "/");
    }
}
```

Java memerlukan pengurai dan penghasil JSON untuk badan permintaan dan jawaban. Alih-alih menambahkan pustaka, **Kode 4.3** adalah pembantu kecil yang menangani persis apa yang dipakai lab (objek datar berisi string, bilangan, boolean, dan `null`, serta daftar string) dan menulis JSON karakter demi karakter sama seperti `json.dumps` Python dengan pemisah ringkas, sehingga kedua server memberikan jawaban yang identik bita demi bita.

Kode 4.3 `MiniJson.java`: JSON secukupnya untuk lab, sesuai dengan keluaran Python.

```java
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

/** Just enough JSON for this lab: flat objects of strings, numbers,
 *  booleans and null, plus lists of strings. Output matches Python's
 *  json.dumps(obj, separators=(",", ":")) character for character. */
public final class MiniJson {
    private final String s;
    private int i;

    private MiniJson(String s) { this.s = s; }

    /** Parse one flat JSON object such as {"id":3,"text":"hi"}. */
    public static Map<String, Object> parseObject(String text) {
        MiniJson p = new MiniJson(text.trim());
        Map<String, Object> map = new LinkedHashMap<>();
        p.expect('{');
        p.skip();
        if (p.peek() == '}') { p.i++; return map; }
        while (true) {
            p.skip();
            String key = p.string();
            p.skip();
            p.expect(':');
            p.skip();
            map.put(key, p.value());
            p.skip();
            char c = p.s.charAt(p.i++);
            if (c == '}') return map;
            if (c != ',') throw new IllegalArgumentException("bad JSON");
        }
    }

    private Object value() {
        char c = peek();
        if (c == '"') return string();
        if (s.startsWith("null", i)) { i += 4; return null; }
        if (s.startsWith("true", i)) { i += 4; return Boolean.TRUE; }
        if (s.startsWith("false", i)) { i += 5; return Boolean.FALSE; }
        int start = i;
        while (i < s.length() && "-0123456789".indexOf(s.charAt(i)) >= 0) i++;
        if (start == i) throw new IllegalArgumentException("bad JSON value");
        return Long.parseLong(s.substring(start, i));
    }
```

Kode 4.3 (Lanjutan) *MiniJson.java: JSON secukupnya untuk lab, sesuai dengan keluaran Python.*

```
    private String string() {
        expect('"');
        StringBuilder out = new StringBuilder();
        while (true) {
            char c = s.charAt(i++);
            if (c == '"') return out.toString();
            if (c != '\\') { out.append(c); continue; }
            char e = s.charAt(i++);
            switch (e) {
                case 'n' -> out.append('\n');
                case 't' -> out.append('\t');
                case 'r' -> out.append('\r');
                case 'b' -> out.append('\b');
                case 'f' -> out.append('\f');
                case 'u' -> {
                    out.append((char) Integer.parseInt(s.substring(i, i + 4), 16));
                    i += 4;
                }
                default -> out.append(e);          // \" \\ \/
            }
        }
    }

    private char peek() {
        if (i >= s.length()) throw new IllegalArgumentException("bad JSON");
        return s.charAt(i);
    }

    private void skip() {
        while (i < s.length() && Character.isWhitespace(s.charAt(i))) i++;
    }

    private void expect(char c) {
        if (peek() != c) throw new IllegalArgumentException("expected " + c);
        i++;
    }
```

Kode 4.3 (Lanjutan) *MiniJson.java: JSON secukupnya untuk lab, sesuai dengan keluaran Python.*

```
    /** Write a value the way Python's json.dumps does (ASCII only). */
    public static String write(Object v) {
        if (v == null) return "null";
        if (v instanceof Boolean || v instanceof Number) return v.toString();
        if (v instanceof List<?> list) {
            StringBuilder out = new StringBuilder("[");
            for (Object item : list) {
                if (out.length() > 1) out.append(',');
                out.append(write(item));
            }
            return out.append(']').toString();
        }
        if (v instanceof Map<?, ?> map) {
            StringBuilder out = new StringBuilder("{");
            for (Map.Entry<?, ?> e : map.entrySet()) {
                if (out.length() > 1) out.append(',');
                out.append(write(e.getKey().toString())).append(':')
                   .append(write(e.getValue()));
            }
            return out.append('}').toString();
        }
        StringBuilder out = new StringBuilder("\"");
        for (char c : v.toString().toCharArray()) {
            switch (c) {
                case '"' -> out.append("\\\"");
                case '\\' -> out.append("\\\\");
                case '\n' -> out.append("\\n");
                case '\r' -> out.append("\\r");
                case '\t' -> out.append("\\t");
                case '\b' -> out.append("\\b");
                case '\f' -> out.append("\\f");
                default -> {
                    if (c < 0x20 || c > 0x7f) {
                        out.append(String.format("\\u%04x", (int) c));
                    } else {
                        out.append(c);
                    }
                }
            }
        }
        return out.append('"').toString();
    }
}
```

4.12.4 Halaman peramban

Kode 4.4 adalah halaman yang dilayani kedua server di `/`. Halaman ini HTML biasa dengan skrip kecil, tanpa *framework*, dan mengimplementasikan empat gagasan bab ini.

Kode 4.4 `feed.html`: halaman langsung dengan lencana kanal, pengiriman optimistis, pengambilan ulang satu butir, dan penjaga balasan basi.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Laundry Live Feed</title>
<style>
  body { font-family: system-ui, sans-serif; margin: 0; background: #f3f4f6;
         color: #12355b; }
  header { background: #12355b; color: #fff; padding: 12px 16px;
           display: flex; justify-content: space-between; }
  main { max-width: 640px; margin: 0 auto; padding: 16px; }
  nav button { border: 1px solid #0f7c7c; background: #fff; color: #0f7c7c;
               padding: 6px 12px; border-radius: 16px; margin-right: 6px; }
  nav button.active { background: #0f7c7c; color: #fff; }
  .badge { background: #c0392b; color: #fff; border-radius: 9px;
           padding: 0 6px; margin-left: 4px; font-size: 12px; }
  textarea { width: 100%; box-sizing: border-box; height: 64px; }
  .card { background: #fff; border-radius: 8px; padding: 10px 12px;
          margin: 8px 0; border-left: 4px solid #0f7c7c; }
  .card.pending { opacity: 0.6; border-left-color: #e0a100; }
  .card.reply { margin-left: 24px; border-left-color: #6b7280; }
  .meta { font-size: 12px; color: #6b7280; }
  #error { color: #c0392b; min-height: 1.2em; }
</style>
</head>
<body>
<header><strong id="title"></strong><span id="status"></span></header>
<main>
  <nav id="tabs"></nav>
  <h2 id="heading"></h2>
  <textarea id="text"></textarea>
  <button id="send"></button> <span id="error"></span>
  <div id="feed"></div>
</main>
<script>
const WORDS = {
  en: { title: "Profitina Laundry community (lab)", live: "live",
        wait: "reconnecting...", send: "Post", sending: "sending...",
        loading: "Loading...", fresh: "new", reply: "reply to #" },
  id: { title: "Komunitas Profitina Laundry (lab)", live: "langsung",
        wait: "menyambung ulang...", send: "Kirim", sending: "mengirim...",
        loading: "Memuat...", fresh: "baru", reply: "balasan untuk #" } };
const params = new URLSearchParams(location.search);
const T = WORDS[params.get("lang")] || WORDS.en;
const me = params.get("me") ||
```

Kode 4.4 (Lanjutan) *feed.html: halaman langsung dengan lencana kanal, pengiriman optimistis, pengambilan ulang satu butir, dan penjaga balasan basi.*

```
  "u" + Math.floor(Math.random() * 2 ** 32).toString(16).padStart(8, "0");
const slow = params.get("slow") || "";      // e.g. "tips:1500" for a demo
const unread = {}, known = new Set();
let active = "general", seq = 0, aborter = null, tempId = 0;
const $ = (id) => document.getElementById(id);

function card(post, pending) {
  const div = document.createElement("div");
  div.className = "card" + (pending ? " pending" : "") +
                  (post.reply_to ? " reply" : "");
  div.dataset.id = post.id;
  const meta = document.createElement("div");
  meta.className = "meta";
  meta.textContent = post.handle + (post.reply_to ?
    " - " + T.reply + post.reply_to : "") + (pending ? " - " + T.sending : "");
  const body = document.createElement("div");
  body.textContent = post.text;             // never innerHTML for user text
  div.append(meta, body);
  return div;
}

function drawTabs(names) {
  $("tabs").replaceChildren(...names.map((name) => {
    const b = document.createElement("button");
    b.textContent = name;
    b.className = name === active ? "active" : "";
    if (unread[name]) {
      const s = document.createElement("span");
      s.className = "badge";
      s.textContent = unread[name] + " " + T.fresh;
      b.append(s);
    }
    b.onclick = () => openChannel(name);
    return b;
  }));
}
```

Kode 4.4 (Lanjutan) *feed.html: halaman langsung dengan lencana kanal, pengiriman optimistis, pengambilan ulang satu butir, dan penjaga balasan basi.*

```
async function openChannel(name) {
  active = name;
  unread[name] = 0;
  drawTabs(channels);
  $("heading").textContent = "# " + name;
  $("feed").textContent = T.loading;          // skeleton at once
  const mine = ++seq;                         // sequence number of this request
  if (aborter) aborter.abort();               // cancel the previous request
  aborter = new AbortController();
  const [slowName, ms] = slow.split(":");
  const delay = slowName === name ? ms : 0;
  try {
    const r = await fetch("/feed?channel=" + name + "&delay=" + delay,
                          { signal: aborter.signal });
    const data = await r.json();
    if (mine !== seq || data.channel !== active) return;   // stale reply
    $("feed").replaceChildren(...data.posts.map((p) => card(p, false)));
    data.posts.forEach((p) => known.add(p.id));
  } catch (e) {
    if (e.name !== "AbortError") $("feed").textContent = String(e);
  }
}

async function onEvent(note) {
  if (note.channel !== active) {              // invalidate: just count it
    if (note.top) unread[note.channel] = (unread[note.channel] || 0) + 1;
    drawTabs(channels);
    return;
  }
  if (known.has(note.id)) return;           // our own post, already shown
  const r = await fetch("/post/" + note.id); // refetch exactly one item
  const post = await r.json();
  if (post.channel !== active || known.has(post.id)) return;
  known.add(post.id);
  $("feed").prepend(card(post, false));
}
```

Kode 4.4 (Lanjutan) *feed.html: halaman langsung dengan lencana kanal, pengiriman optimistis, pengambilan ulang satu butir, dan penjaga balasan basi.*

```
async function send() {
  const text = $("text").value;
  if (!text.trim()) return;
  $("error").textContent = "";
  const draft = { id: "tmp" + (++tempId), channel: active, handle: me,
                  text: text, reply_to: null };
  const node = card(draft, true);
  $("feed").prepend(node);                    // optimistic: show it now
  $("text").value = "";                       // and clear the composer
  try {
    const r = await fetch("/post", { method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ channel: draft.channel, handle: me,
                             text: text }) });
    const answer = await r.json();
    if (!r.ok) throw new Error(answer.error);
    known.add(answer.id);
    if ($("feed").querySelector('[data-id="' + answer.id + '"]')) {
      node.remove();                          // the live event was faster
    } else {
      node.replaceWith(card({ ...draft, id: answer.id }, false));
    }
  } catch (e) {
    node.remove();                            // roll back the optimistic card
    $("text").value = text;                   // give the text back
    $("error").textContent = e.message;
  }
}

let channels = [];
async function start() {
  document.title = T.title;
  $("title").textContent = T.title;
  $("send").textContent = T.send;
  $("send").onclick = send;
  $("text").oninput = () => { $("error").textContent = ""; };
  channels = await (await fetch("/channels")).json();
  openChannel("general");
  const source = new EventSource("/events");  // reconnects by itself
  source.onopen = () => { $("status").textContent = T.live; };
  source.onerror = () => { $("status").textContent = T.wait; };
  source.addEventListener("post", (e) => onEvent(JSON.parse(e.data)));
}
start();
</script>
</body>
```

Kode 4.4 (Lanjutan) *feed.html: halaman langsung dengan lencana kanal, pengiriman optimistis, pengambilan ulang satu butir, dan penjaga balasan basi.*

```
</html>
```

- **Aliran.** `new EventSource("/events")` membuka aliran; penangan `onopen` dan `onerror` menampilkan *live* atau *reconnecting…* (dalam mode `?lang=id`: *langsung* atau *menyambung ulang…*) di kepala halaman. Penyambungan ulang sama sekali tidak memerlukan kode.
- **Tandai basi atau ambil ulang.** `onEvent` mengikuti **Gambar 4.8**: peristiwa untuk kanal lain hanya menaikkan lencana kanal itu (bila `top`); peristiwa untuk kanal yang tampil mengambil tepat satu kiriman dengan `/post/n`.

- **Pengiriman optimistis.** `send` menggambar kartu tertunda dan mengosongkan kotak penyusun *sebelum* permintaan; bila berhasil, kartu diganti kiriman yang terkonfirmasi, dan bila gagal, kartu dihapus dan teks dikembalikan, seperti pada **Gambar 4.7**.
- **Penjaga balasan basi.** `openChannel` menaikkan `seq` dan membatalkan permintaan sebelumnya; ketika jawaban tiba, jawaban itu hanya dipakai bila nomor urutnya masih yang terbaru dan kanalnya masih aktif.

Perhatikan juga `body.textContent = post.text`: teks pengguna selalu disisipkan sebagai teks, tidak pernah sebagai HTML, sehingga kiriman yang berisi `<script>` menampilkan karakter-karakter itu alih-alih menjalankannya. Bab 5 kembali ke gagasan ini dari sisi keamanan.

4.12.5 Klien konsol

Kode 4.5 dan **Kode 4.6** adalah klien konsol. `listen` membaca aliran baris demi baris, mengumpulkan baris `field: value` menjadi sebuah peristiwa, dan pada baris kosong yang mengakhiri peristiwa mengambil satu kiriman yang diumumkannya; `post` mengirim sebuah kiriman lalu mencetak kode status dan jawaban server.

Kode 4.5 `feed_client.py`: mendengarkan aliran dan mengirim, dalam Python.

```
"""Console client for the live feed: listen to the stream, or send a post.

  python feed_client.py listen [--max 5] [--last-id 0]
  python feed_client.py post --text "Hello" [--channel general] [--reply-to 1]
Add --url http://127.0.0.1:7041 to talk to the Java server instead.
"""
import argparse
import json
import urllib.error
import urllib.request


def fetch_post(base, post_id):
    with urllib.request.urlopen("%s/post/%d" % (base, post_id)) as reply:
        return json.load(reply)


def show(post):
    where = " (reply to #%d)" % post["reply_to"] if post["reply_to"] else ""
    print("#%d [%s] %s%s: %s" % (post["id"], post["channel"], post["handle"],
                                 where, post["text"]), flush=True)


def listen(base, max_events, last_id):
    request = urllib.request.Request(base + "/events")
    request.add_header("Accept", "text/event-stream")
    if last_id:
        request.add_header("Last-Event-ID", str(last_id))   # ask for a replay
    seen = 0
    event = {}
    with urllib.request.urlopen(request) as stream:
        for raw in stream:                      # one line at a time
            line = raw.decode("utf-8").rstrip("\r\n")
            if line.startswith(":"):
                print("(heartbeat)", flush=True)
            elif line.startswith("retry:"):
                print("server asks us to reconnect after",
                      line[6:].strip(), "ms", flush=True)
            elif line and ":" in line:
                field, _, value = line.partition(":")
                event[field] = value.lstrip(" ")
            elif not line and "data" in event:   # blank line ends an event
                note = json.loads(event["data"])
                show(fetch_post(base, note["id"]))   # refetch one item
                event = {}
                seen += 1
```

Kode 4.5 (Lanjutan) *feed_client.py: mendengarkan aliran dan mengirim, dalam Python.*

```
                if seen == max_events:
                    return


def send(base, channel, handle, text, reply_to):
    body = {"channel": channel, "handle": handle, "text": text,
            "reply_to": reply_to}
    request = urllib.request.Request(
        base + "/post", data=json.dumps(body).encode("utf-8"),
        headers={"Content-Type": "application/json"}, method="POST")
    try:
        with urllib.request.urlopen(request) as reply:
            print(reply.status, reply.read().decode("utf-8"))
    except urllib.error.HTTPError as err:
        print(err.code, err.read().decode("utf-8"))


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("command", choices=["listen", "post"])
    parser.add_argument("--url", default="http://127.0.0.1:7040")
    parser.add_argument("--max", type=int, default=5)
    parser.add_argument("--last-id", type=int, default=0)
    parser.add_argument("--channel", default="general")
    parser.add_argument("--handle", default="u3f9a1c20")
    parser.add_argument("--text", default="")
    parser.add_argument("--reply-to", type=int, default=None)
    args = parser.parse_args()
    if args.command == "listen":
        listen(args.url, args.max, args.last_id)
    else:
        send(args.url, args.channel, args.handle, args.text, args.reply_to)


if __name__ == "__main__":
    main()
```

Kode 4.6 `FeedClient.java`: klien yang sama dengan `java.net.http.HttpClient` dan `BodyHandlers.ofLines()`.

```
import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.stream.Stream;

/** Console client for the live feed: listen to the stream, or send a post.
 *    java -cp out FeedClient listen [--max 5] [--last-id 0]
 *    java -cp out FeedClient post --text "Hello" [--channel general]
 *  Add --url http://127.0.0.1:7040 to talk to the Python server instead. */
public class FeedClient {
    static final HttpClient HTTP = HttpClient.newBuilder()
            .version(HttpClient.Version.HTTP_1_1).build();

    static Map<String, Object> fetchPost(String base, long id)
            throws IOException, InterruptedException {
        HttpRequest req = HttpRequest.newBuilder(
                URI.create(base + "/post/" + id)).build();
        return MiniJson.parseObject(
                HTTP.send(req, HttpResponse.BodyHandlers.ofString()).body());
    }

    static void show(Map<String, Object> post) {
        Object reply = post.get("reply_to");
        String where = reply != null ? " (reply to #" + reply + ")" : "";
        System.out.println("#" + post.get("id") + " [" + post.get("channel") + "] "
                + post.get("handle") + where + ": " + post.get("text"));
    }

    static void listen(String base, int maxEvents, int lastId)
            throws IOException, InterruptedException {
        HttpRequest.Builder b = HttpRequest
                .newBuilder(URI.create(base + "/events"))
                .header("Accept", "text/event-stream");
        if (lastId > 0) b.header("Last-Event-ID", String.valueOf(lastId));
        HttpResponse<Stream<String>> resp =
                HTTP.send(b.build(), HttpResponse.BodyHandlers.ofLines());
        int seen = 0;
        Map<String, String> event = new HashMap<>();
        try (Stream<String> lines = resp.body()) {
            Iterator<String> it = lines.iterator();       // one line at a time
```

Kode 4.6 (Lanjutan) *FeedClient.java: klien yang sama dengan java.net.http.HttpClient dan BodyHandlers.ofLines().*

```
            while (it.hasNext()) {
                String line = it.next();
                if (line.startsWith(":")) {
                    System.out.println("(heartbeat)");
                } else if (line.startsWith("retry:")) {
                    System.out.println("server asks us to reconnect after "
                            + line.substring(6).strip() + " ms");
                } else if (!line.isEmpty() && line.contains(":")) {
                    int colon = line.indexOf(':');
                    String value = line.substring(colon + 1);
                    event.put(line.substring(0, colon),
                              value.startsWith(" ") ? value.substring(1) : value);
                } else if (line.isEmpty() && event.containsKey("data")) {
                    Map<String, Object> note =
                            MiniJson.parseObject(event.get("data"));
                    show(fetchPost(base, (Long) note.get("id")));  // refetch one
                    event.clear();
                    if (++seen == maxEvents) return;
                }
            }
        }
    }

    static void send(String base, String channel, String handle, String text,
                     Long replyTo) throws IOException, InterruptedException {
        Map<String, Object> body = new LinkedHashMap<>();
        body.put("channel", channel);
        body.put("handle", handle);
        body.put("text", text);
        body.put("reply_to", replyTo);
        HttpRequest req = HttpRequest.newBuilder(URI.create(base + "/post"))
                .header("Content-Type", "application/json")
                .POST(HttpRequest.BodyPublishers.ofString(MiniJson.write(body)))
                .build();
        HttpResponse<String> resp =
                HTTP.send(req, HttpResponse.BodyHandlers.ofString());
        System.out.println(resp.statusCode() + " " + resp.body());
    }
```

Kode 4.6 (Lanjutan) *FeedClient.java: klien yang sama dengan java.net.http.HttpClient dan BodyHandlers.ofLines().*

```
    public static void main(String[] args) throws Exception {
        Map<String, String> opt = new HashMap<>(Map.of(
                "--url", "http://127.0.0.1:7041", "--max", "5",
                "--last-id", "0", "--channel", "general",
                "--handle", "u3f9a1c20", "--text", ""));
        for (int k = 1; k + 1 < args.length; k += 2) {
            opt.put(args[k], args[k + 1]);          // --name value pairs
        }
        String command = args.length > 0 ? args[0] : "";
        if (command.equals("listen")) {
            listen(opt.get("--url"), Integer.parseInt(opt.get("--max")),
                   Integer.parseInt(opt.get("--last-id")));
        } else if (command.equals("post")) {
            String reply = opt.get("--reply-to");
            send(opt.get("--url"), opt.get("--channel"), opt.get("--handle"),
                 opt.get("--text"), reply == null ? null : Long.parseLong(reply));
        } else {
            System.out.println("usage: FeedClient listen|post [options]");
        }
    }
}
```

4.12.6 Menjalankannya

Buka tiga terminal di folder `Chapter04`. Di terminal pertama, jalankan server Python dengan detak jantung pendek dua detik agar kita dapat melihatnya:

```
cd python
python feed_server.py --heartbeat 2
```

Di terminal kedua, jalankan pendengar yang berhenti setelah empat peristiwa:

```
python python/feed_client.py listen --max 4
```

Di terminal ketiga, kirim lima kiriman, salah satunya memakai *handle* yang tidak sah:

```
python python/feed_client.py post --channel general --text "Machine 3 is fixed; express
orders are back."
python python/feed_client.py post --channel tips --handle u7d20be41 --text "Tip: wash
batik inside out in cold water."
python python/feed_client.py post --channel general --handle budi --text "Hello!"
python python/feed_client.py post --channel general --handle u7d20be41 --reply-to 1 --
text "Great news, thanks!"
python python/feed_client.py post --channel promo --text "Weekend promo: 10% off bed
covers."
```

Terminal ketiga mencetak jawaban server:

```
201 {"id":1}
201 {"id":2}
422 {"error":"handle must be neutral, like u3f9a1c20"}
201 {"id":3}
201 {"id":4}
```

Pendengar mencetak:

```
server asks us to reconnect after 3000 ms
(heartbeat)
#1 [general] u3f9a1c20: Machine 3 is fixed; express orders are back.
#2 [tips] u7d20be41: Tip: wash batik inside out in cold water.
#3 [general] u7d20be41 (reply to #1): Great news, thanks!
#4 [promo] u3f9a1c20: Weekend promo: 10% off bed covers.
```

Setiap baris layak dikomentari. Baris pertama adalah medan `retry`. `(heartbeat)` adalah komentar `: ping`, dikirim karena dua detik berlalu tanpa kiriman. Lalu datang keempat kiriman yang diterima, masing-masing diambil satu per satu setelah peristiwa tipisnya; kiriman yang ditolak tidak pernah sampai ke aliran. Setelah peristiwa keempat, pendengar keluar.

Sekarang anggaplah seorang pendengar terputus setelah peristiwa 2 lalu menyambung ulang, seperti yang dilakukan peramban, dengan `Last-Event-ID: 2`:

```
python python/feed_client.py listen --max 2 --last-id 2
```

```
server asks us to reconnect after 3000 ms
#3 [general] u7d20be41 (reply to #1): Great news, thanks!
#4 [promo] u3f9a1c20: Weekend promo: 10% off bed covers.
```

Server memutar ulang tepat dua kiriman yang terlewat oleh pendengar itu. Log terminal pertama menceritakan seluruh kisah dari sisi server:

```
feed server on http://127.0.0.1:7040/
listener connected, last id 0 (1 open)
post #1 [general] by u3f9a1c20
post #2 [tips] by u7d20be41
rejected: handle must be neutral, like u3f9a1c20
post #3 [general] by u7d20be41
post #4 [promo] by u3f9a1c20
listener gone (0 open)
listener connected, last id 2 (1 open)
listener gone (0 open)
```

Perhatikan baris `listener gone (0 open)`: server baru menyadari bahwa pendengar pertama telah pergi ketika sebuah detak jantung gagal ditulis. Tanpa detak jantung, server dapat terus memegang sambungan ke pembaca yang sudah lama menghilang. Hentikan server dengan **Ctrl+C**.

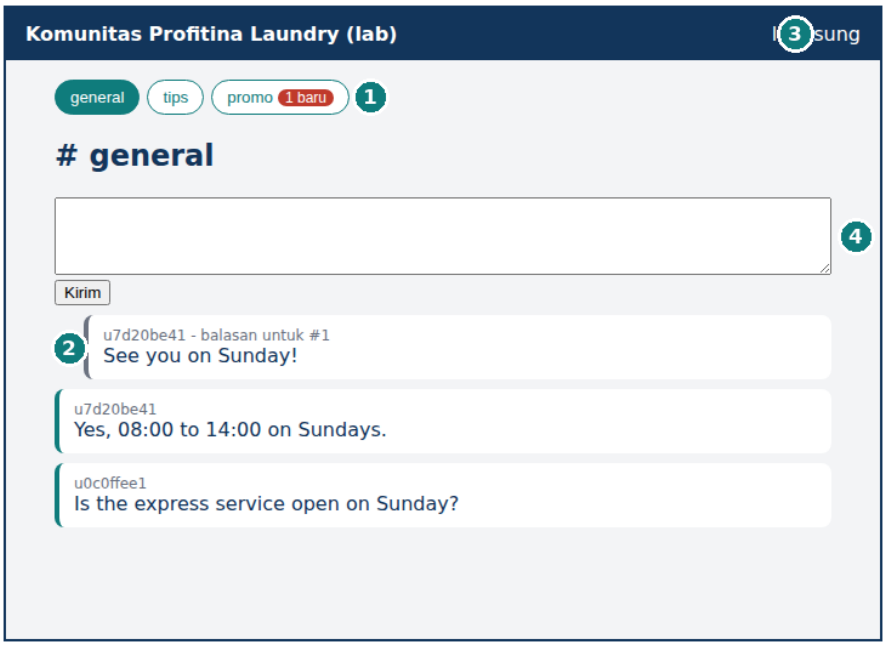
Server dan klien Java menerima perintah yang sama (`java -cp out FeedServer --heartbeat 2` di folder `java`, dan `java -cp java/out FeedClient ...` sebagai ganti `python python/feed_client.py ...`; alamat bawaan klien Java adalah port 7041 milik server Java) dan menghasilkan keluaran yang sama. Satu-satunya perbedaan yang mungkin Anda lihat adalah baris `(heartbeat)` tambahan, karena menyalakan mesin virtual Java baru untuk setiap perintah `post` di beberapa mesin memerlukan lebih dari dua detik. Setiap klien bekerja dengan server mana pun: klien Python terhadap server Java memberikan keluaran persis seperti di atas.

4.12.7 Memeriksa halaman di peramban tanpa layar

Buka `http://127.0.0.1:7040/` di peramban untuk mencoba halaman secara manual; tambahkan `?lang=id` untuk label berbahasa Indonesia. Buka di dua jendela berdampingan lalu kirim di salah satunya: kiriman muncul di jendela lain tanpa memuat ulang. Untuk memeriksa halaman secara otomatis, folder kode menyertakan `tools/check_page.py`, yang memakai Playwright [15] untuk menjalankan peramban Chromium sungguhan tanpa jendela. Alat ini memerlukan `pip install playwright` dan `python -m playwright install chromium` sekali, serta server yang baru dinyalakan. Alat ini menahan setiap POST selama setengah detik untuk meniru jaringan yang lambat, lalu memeriksa delapan hal:

```
PASS optimistic card shown at once as pending
PASS composer cleared at once
PASS card confirmed by the server
PASS failed post rolled back, text restored
PASS event for the open channel fetched one post
PASS counter counts top-level posts only (1 new)
PASS late reply for 'tips' discarded
PASS no JavaScript errors
8 of 8 checks passed
```

Pemeriksaan-pemeriksaan itu adalah gagasan-gagasan bab ini yang dibuat dapat diuji: kartu optimistis langsung muncul dan kotak penyusun kosong; server mengonfirmasinya; kiriman 300 karakter dibatalkan dengan teks dikembalikan; peristiwa untuk kanal yang terbuka mengambil satu kiriman; kiriman tingkat atas di kanal lain menaikkan lencananya menjadi "1 new" sedangkan balasan tidak; balasan lambat untuk *tips* yang tiba setelah anggota berpindah ke *promo* dibuang; dan tidak ada galat JavaScript. Kedelapan pemeriksaan yang sama juga lulus terhadap server Java. **Gambar 4.12** adalah halaman sebagaimana ditinggalkan pemeriksaan, dalam mode bahasa Indonesia.



Gambar 4.12 Halaman lab di Chromium tanpa layar setelah pemeriksaan otomatis (`?lang=id`). Penanda 1: lencana "1 baru" milik kanal yang tidak tampil; 2: sebuah balasan, menjorok; 3: status sambungan; 4: kotak penyusun, langsung kosong saat kiriman dikirim.

4.12.8 Apa yang tidak dicakup lab

Lab ini jujur tentang batas-batasnya, dan menyebutkannya adalah bagian dari pelajaran. Kiriman tinggal di memori dan lenyap ketika server berhenti; sistem sungguhan menyimpannya di basis data. Tidak ada proses masuk, sehingga siapa pun boleh mengirim atas nama *handle* sah mana pun. Setiap aliran menempati satu utas, yang wajar untuk ruang kelas dan boros untuk ribuan pendengar. Pemutaran ulang menyimpan seluruh riwayat; server sungguhan menyimpan jendela yang terbatas. Dan tidak ada yang menyaring tautan dalam kiriman; itulah lab bab berikutnya.

4.13 Profitina dalam Praktik

Gagasan-gagasan bab ini dapat dilihat bekerja di aplikasi yang berjalan di `app.profitina.com`.

Buka komunitas dan amati. Tab komunitas diperbarui selagi Anda membaca; tidak ada yang perlu dimuat ulang. Buka sebuah utas dengan rute *hash* `#/komunitas/post/{id}` dan Anda sedang melihat pohon dari Subbab 4.5.1. Di profitina.com, umpan yang sama muncul di widget tersemat, hanya-baca bagi pengunjung.

Ikuti dua kanal. Lencana merah hanya menghitung kiriman tingkat atas, menurut Persamaan (4.6). Repost mendarat di kanal kiriman asli kecuali Anda memilih Umum di dialog repost.

Periksa gerbangnya. Dialog *Buat kanal* menjelaskan gerbang saldo Persamaan (4.5); saldo di bawah 50 PROFIT cukup untuk melihat penolakannya, dan tidak ada yang dipotong ketika Anda berhasil.

Lihat dasbor Anda. Pita prioritas pada **Gambar 4.6** adalah hal pertama yang Anda lihat setelah masuk; setiap butir membawa Anda ke bagiannya.

Yang berjalan dan yang masih rencana. Segala hal pada **Tabel 4.1** berjalan pada rilis September 2026 (versi `0.14.5-testnet`). Pesan grup, @mention, dan hadiah untuk DM tidak direncanakan; ketiadaannya adalah keputusan rancangan, bukan fitur yang belum selesai. Pertanyaan tentang komunitas dapat dikirim ke `info@profitina.com`.

4.14 Kesalahpahaman yang Umum

“Waktu-nyata memerlukan WebSocket.” Hanya bila data harus mengalir dua arah berkali-kali dalam satu detik. Umpan adalah kabar dari server ke peramban ditambah kiriman sesekali, dan SSE di atas HTTP biasa mengerjakannya dengan penyambungan ulang dan pemutaran ulang yang sudah tersedia (Subbab 4.4.4).

“Polling setiap detik hampir sama baiknya.” Menurut Persamaan (4.1), cara itu melipatlimakan permintaan dibandingkan polling lima detik, dan menurut Persamaan (4.3) hampir semuanya pulang tanpa hasil. Bukan hampir sama baiknya; jauh lebih mahal.

“Peristiwa sebaiknya membawa kiriman utuh agar halaman tidak perlu bertanya.” Peristiwa gemuk dikirim ke setiap halaman terbuka, termasuk semua yang tidak menampilkan kanal itu. Peristiwa tipis ditambah pengambilan ulang satu butir mengirim kiriman utuh hanya ke halaman yang menampilkannya (Subbab 4.11.2).

“Penghitung belum dibaca menghitung semua yang baru.” Penghitung Profitina menghitung kiriman *tingkat atas* yang baru, agar angkanya cocok dengan apa yang terlihat di bagian atas umpan (Subbab 4.7.2).

“Gerbang kanal memungut 50 PROFIT.” Gerbang itu mensyaratkan saldo 50 PROFIT dan tidak memotong apa pun (Subbab 4.6.2).

“Pengiriman optimistis berpura-pura kiriman berhasil.” Pengiriman optimistis menampilkan ramalan tertunda dan mengoreksinya dalam satu perjalanan pulang-pergi; bila gagal, teks kembali beserta alasannya (Subbab 4.10.2).

“Kalau server cepat, balapan tidak mungkin terjadi.” Kecepatan membuat balapan lebih jarang, bukan mustahil; balapan balasan terlambat pada **Gambar 4.10** justru baru *terlihat* setelah umpan menjadi cepat. Penjaga diperlukan di setiap panel.

“Administrator dapat membaca DM bila perlu.” Tidak bisa. Satu-satunya pengecualian adalah salinan beku pesan yang dilaporkan penerimanya (Subbab 4.8.1).

“Detak jantung hanya untuk membuat proxy tetap terjaga.” Detak jantung juga cara server mengetahui bahwa seorang pendengar sudah pergi, seperti diperlihatkan log lab.

4.15 Ringkasan Bab

- Komunitas Profitina melayani *Bersama* dan *Berbagi* dalam semboyannya: umpan langsung, utas dan repost-kutip, kanal, ikutan dengan lencana “N baru”, DM, profil berbasis persetujuan, dan kokpit dasbor, semuanya dibangun antara Juli dan September 2026 (**Tabel 4.1**).
- Web dibangun untuk bertanya. **Polling** berbiaya $3600N/T$ permintaan per jam (Persamaan (4.1)), tunda rata-rata $T/2$ (Persamaan (4.2)), dan jawaban yang sebagian besar kosong (Persamaan (4.3)). **WebSocket** memberi pipa dua arah di luar HTTP. **Server-Sent Events** membiarkan satu tanggapan HTTP tetap terbuka dan mendorong peristiwa teks, dengan `retry`, pemutaran ulang `Last-Event-ID`, dan detak jantung berupa komentar yang sudah tersedia (**Tabel 4.2**).
- Utas adalah pohon balasan. Repost mewarisi kanal kiriman asli kecuali anggota memilih Umum. Kiriman dapat disunting selama 15 menit (Persamaan (4.4)), dan hadiahnya dihitung ulang.
- Pembuatan kanal dijaga oleh kepemilikan sekurang-kurangnya 50 PROFIT, tidak pernah dipotong, dengan paling banyak dua kanal baru per hari dan lima yang aktif (Persamaan (4.5)).
- Lencana “N baru” menghitung kiriman tingkat atas sesudah penanda terakhir-dilihat (Persamaan (4.6)); lencana dijaga tetap hidup oleh peristiwa tipis tanpa biaya jaringan. Kata yang sama harus berarti angka yang sama di mana pun.
- DM bersifat satu lawan satu, tanpa hadiah, dan tidak dapat dibaca administrator kecuali salinan beku pesan yang dilaporkan. Profil membagikan setiap butir hanya atas persetujuannya sendiri, yang dimulai dalam keadaan mati; *handle* bersifat netral.
- Dasbor adalah satu permintaan yang mengisi pita prioritas tiga tingkat dan tiga kartu.
- Pengiriman optimistis menampilkan kiriman tertunda seketika dan membatalkannya dengan teks dikembalikan bila gagal.
- Tiga pelajaran dari rilis September 2026: jalur baca tidak pernah memanggil model; peristiwa menandai basi atau mengambil satu butir, bukan memuat ulang segalanya; kueri tidak boleh tumbuh bersama ukuran halaman (Persamaan (4.7)), yang membawa satu halaman umpan dari 868 ms menjadi 11 ms. Pelajaran keempat: jaga setiap panel dari balasan terlambat.
- Lab G2 membangun semua ini dalam bentuk mini dalam Python dan Java, dengan halaman peramban yang diperiksa otomatis di Chromium tanpa layar.

4.16 Soal Latihan

Soal 4.1. Sebuah kanal kelas memiliki 250 halaman terbuka. (a) Dengan Persamaan (4.1), berapa permintaan per jam yang dihasilkan polling setiap 10 detik? (b) Berapa tunda rata-ratanya menurut Persamaan (4.2)? (c) Jika kanal menerima 30 kiriman per jam secara acak, berapa bagian polling yang kosong menurut Persamaan (4.3)?

Soal 4.2. Dengan SSE dan detak jantung lab 15 detik, berapa komentar detak jantung per jam yang ditulis server ke 250 halaman yang sama bila tidak ada yang mengirim? Bandingkan dengan jawaban Soal 4.1(a) dan jelaskan mengapa kedua angka itu tidak mengukur jenis biaya yang sama.

Soal 4.3. Tuliskan, bita demi bita, peristiwa yang dikirim server lab untuk kiriman nomor 7, sebuah balasan di kanal *tips*.

Soal 4.4. Penanda terakhir-dilihat seorang anggota untuk kanal *kelas-oop* adalah 40. Kemudian kiriman 41 sampai 48 tiba: 41, 43, 44, dan 47 tingkat atas, sisanya balasan. Apa yang ditampilkan lencana menurut Persamaan (4.6)? Apa yang akan ditampilkan lencana yang menghitung semuanya, dan mengapa itu menyesatkan?

Soal 4.5. Budi memegang 55 PROFIT, belum membuat kanal hari ini, dan memiliki 5 kanal aktif. Dapatkah ia membuat kanal? Apa perubahan terkecil pada keadaannya yang akan membolehkannya, dan apa yang terjadi pada saldonya bila ia kemudian membuatnya?

Soal 4.6. Sebuah kiriman dibuat pukul 09.52.30 WIB. Dengan Persamaan (4.4), sampai pukul berapa kiriman itu boleh disunting? Anggota menyuntingnya pukul 10.05 dan menghapus satu-satunya kalimat yang berguna. Apa yang terjadi pada hadiahnya, dan mengapa itu aturan yang tepat?

Soal 4.7. Pada halaman lab, bayangkan Anda menghapus baris `if (aborter) aborter.abort();` dan baris `if (mine !== seq || data.channel !== active) return;`. Jelaskan dengan tepat apa yang akan dilihat anggota dalam skenario **Gambar 4.10**, dan pemeriksaan otomatis mana di antara kedelapannya yang akan gagal. Kemudian kembalikan hanya baris `abort`: apakah halaman kembali benar, dan mengapa pemeriksaan nomor urut tetap layak dipertahankan?

Soal 4.8. Misalkan menampilkan satu kiriman memerlukan 11 kueri basis data dan 2 ms waktu basis data per kueri, dan satu halaman menampilkan n kiriman. (a) Tuliskan waktu basis data halaman sebelum pengambilan awal sebagai fungsi n. (b) Jika pengambilan awal memakai 12 kueri berapa pun n, mulai n berapa pengambilan awal lebih cepat? (c) Mengapa satu sambungan baru per kueri membuat keadaan jauh lebih buruk daripada yang disiratkan (a)?

Soal 4.9. Lab memutar ulang seluruh riwayat sesudah `Last-Event-ID`. Usulkan perubahan yang hanya menyimpan 100 kiriman terakhir di memori untuk pemutaran ulang, dan sebutkan apa yang harus dilakukan server ketika seorang pendengar meminta pemutaran ulang yang lebih tua daripada jendela itu.

Soal 4.10. Jelaskan mengapa DM bersifat satu lawan satu, tanpa hadiah, dan tidak dapat dibaca administrator, serta mengapa satu-satunya pengecualian (salinan beku pesan yang dilaporkan) tidak melanggar janji privasi.

Soal 4.11. Jalankan lab dengan server Java di port 7041 dan klien Python (`--url http://127.0.0.1:7041`). Lalu matikan server selagi seorang pendengar tersambung. Apa yang dilakukan klien Python? Apa yang akan dilakukan `EventSource` milik peramban, dan medan mana dalam aliran yang mengendalikannya?

Soal 4.12. Pertanyaan rancangan. Seorang anggota meminta "@mention di DM agar saya bisa menambahkan orang ketiga". Dengan prinsip-prinsip Subbab 4.2.2 dan 4.8.1, tuliskan jawaban singkat dan sopan yang menjelaskan mengapa platform tidak menyediakannya dan di mana percakapan semacam itu seharusnya berlangsung.

4.17 Penyelesaian Soal Latihan

Penyelesaian 4.1. (a) $R = 3600 \times 250/10 = 90{,}000$ permintaan per jam, 25 per detik. (b) $T/2 = 5$ detik rata-rata, paling lama 10 detik. (c) $\lambda = 30/3600 = 1/120$ per detik, jadi $\lambda T = 10/120 \approx 0.0833$ dan $P(\text{kosong}) = e^{-0.0833} \approx 0.920$: sekitar 92 % polling tidak menghasilkan apa-apa.

Penyelesaian 4.2. Satu detak jantung per halaman setiap 15 detik berarti $3600/15 = 240$ per jam per halaman, jadi $240 \times 250 = 60{,}000$ komentar per jam. Angka itu sebanding dengan 90.000

polling, tetapi biayanya sangat berbeda: detak jantung hanya sekitar delapan bita yang ditulis ke sambungan yang sudah terbuka, tanpa permintaan yang harus diurai, tanpa autentikasi, dan tanpa kueri basis data, sedangkan setiap polling adalah permintaan dan tanggapan HTTP yang lengkap. Menghitung pesan tidak sama dengan menghitung pekerjaan.

Penyelesaian 4.3. Fungsi `thin_event` menulis

```
id: 7
event: post
data: {"id":7,"channel":"tips","top":false}
```

dengan satu baris kosong di akhir; `top` bernilai `false` karena kiriman itu balasan.

Penyelesaian 4.4. Kiriman bernomor lebih dari 40 yang tingkat atas: 41, 43, 44, dan 47, jadi lencana menampilkan “4 baru”. Menghitung semuanya akan menampilkan “8 baru”, padahal membuka kanal hanya memperlihatkan empat kiriman baru di atas; empat lainnya balasan di dalam utas, sehingga lencana menjanjikan lebih banyak daripada yang dapat dilihat anggota.

Penyelesaian 4.5. Tidak: $55 \geq 50$ dan $0 < 2$ terpenuhi, tetapi $5 < 5$ gagal. Perubahan terkecil adalah membuat salah satu kanalnya tidak aktif, sehingga $a = 4$. Maka ketiga syarat Persamaan (4.5) terpenuhi, ia dapat membuat kanal, dan saldonya tetap 55 PROFIT karena gerbang tidak pernah memotong.

Penyelesaian 4.6. Sampai pukul 10.07.30 WIB, lima belas menit sesudah 09.52.30. Suntingan pukul 10.05 diperbolehkan. Hadiahnya dihitung ulang sepenuhnya dari teks baru dan akan turun, mungkin sampai nol. Kalau tidak, seorang anggota dapat memperoleh hadiah untuk kiriman yang berguna lalu menggantinya dengan apa saja; penghitungan ulang menjaga hadiah tetap terikat pada apa yang benar-benar dilihat pembaca.

Penyelesaian 4.7. Anggota mengeklik *tips* (lambat), lalu *promo*. Kiriman promo muncul di bawah judul “# promo”. Sekitar 1,5 detik kemudian jawaban tips tiba; tanpa kedua penjaga, jawaban itu menggantikan umpan, sehingga layar menampilkan kiriman tips di bawah judul “# promo”, dan pemeriksaan “late reply for ‘tips’ discarded” gagal. Dengan baris `abort` dikembalikan, permintaan tips dibatalkan ketika *promo* diklik, `fetch`-nya ditolak dengan `AbortError` yang diabaikan halaman, dan pemeriksaan lulus kembali. Pemeriksaan nomor urut tetap layak dipertahankan karena pembatalan bergantung pada waktu: jawaban yang sudah tiba dan sedang diurai ketika klik kedua terjadi tidak dapat dibatalkan lagi, dan jalur kode lain (tombol coba lagi, jawaban dari *cache*) dapat memulai permintaan yang tidak diketahui `AbortController` mana pun. Nomor urut adalah penjaga yang tetap berlaku apa pun yang dilakukan jaringan.

Penyelesaian 4.8. (a) $(1 + 11n) \times 2 = 2 + 22n$ ms. (b) Pengambilan awal berbiaya $12 \times 2 = 24$ ms; $2 + 22n > 24$ bila $n > 1$, jadi mulai $n = 2$ pengambilan awal lebih cepat, dan untuk $n = 30$ perbandingannya 662 ms lawan 24 ms. (c) Membuka sambungan melibatkan jabat tangan jaringan, autentikasi, dan penyiapan proses di sisi basis data, yang biasanya jauh lebih mahal daripada kueri sederhana; membayarnya untuk setiap kueri melipatgandakan total, itulah sebabnya waktu terukur (868 ms) jauh di atas yang disiratkan waktu kueri saja dan mengapa kolam sambungan sendiri menghapus 122 sambungan per halaman.

Penyelesaian 4.9. Simpan kiriman dalam struktur yang terbatas, misalnya `collections.deque(maxlen=100)` di Python atau `ArrayDeque` yang dipangkas menjadi 100 di Java, bersama nomor kiriman tertua yang disimpan. Jika seorang pendengar meminta `Last-Event-ID` yang lebih tua daripada nomor itu, server tidak dapat memutar ulang secara tepat;

server sebaiknya mengirim peristiwa seperti `event: reset` yang meminta klien memuat ulang kanal dengan `GET /feed` biasa, alih-alih diam-diam memutar ulang riwayat yang tidak lengkap.

Penyelesaian 4.10. Satu lawan satu mencegah DM berubah menjadi kanal grup tanpa moderasi, tempat spam dan keroyokan menyebar; diskusi publik tempatnya di kanal yang memiliki moderator. Tanpa hadiah, karena tidak seorang pun dapat memeriksa apakah pesan privat berguna, sehingga hadiah akan mengundang "ternak" hadiah. Tidak dapat dibaca administrator, karena pesan privat yang dapat dibaca operator tidaklah privat. Salinan beku pesan yang *dilaporkan* hanya tercipta oleh tindakan melapor dari penerima sendiri, hanya membuka pesan itu, dan membekukannya agar tidak dapat diubah sesudahnya; sisa percakapan tetap privat.

Penyelesaian 4.11. Aliran `urlopen` pada klien Python berakhir (atau memunculkan galat sambungan) dan program keluar; klien itu tidak memiliki logika penyambungan ulang. `EventSource` milik peramban akan menunggu selama selang `retry`, di sini 3000 ms, lalu menyambung ulang secara otomatis sambil mengirim `Last-Event-ID` berisi `id` terakhir yang diterimanya; ketika server kembali, server memutar ulang yang terlewat (di lab, hanya kiriman yang masih ada di memori, yang setelah server dinyalakan ulang berarti tidak ada).

Penyelesaian 4.12. Salah satu jawaban yang mungkin: "Terima kasih atas usulannya. DM di Profitina sengaja dibuat satu lawan satu: DM adalah percakapan privat yang tidak dapat dibaca orang lain, termasuk platform, dan itu hanya dapat dikelola di antara dua orang. Percakapan tiga orang atau lebih sudah merupakan komunitas kecil, dan komunitas paling baik berlangsung di kanal, tempat moderator dapat menyematkan, menyembunyikan, dan membisukan, dan tempat aturannya terlihat oleh semua orang. Anda dipersilakan membuat kanal untuk kelompok Anda setelah saldo Anda memenuhi gerbangnya; tidak ada yang dipotong."

Daftar Pustaka

[1] Profitina, *Profitina Whitepaper — Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[2] ANTARA, "Survei APJII: Penetrasi internet di Indonesia 2026 capai 81,7 persen," May 21, 2026. [Online]. Available: https://www.antaranews.com/berita/5576225/survei-apjii-penetrasi-internet-di-indonesia-2026-capai-817-persen

[3] S. Kemp, "Digital 2026: Indonesia," DataReportal, Nov. 5, 2025. [Online]. Available: https://datareportal.com/reports/digital-2026-indonesia

[4] I. Fette and A. Melnikov, "The WebSocket Protocol," IETF RFC 6455, Dec. 2011.

[5] WHATWG, "Server-sent events," in *HTML Living Standard*, sec. 9.2. [Online]. Available: https://html.spec.whatwg.org/multipage/server-sent-events.html

[6] MDN Web Docs, "Using server-sent events," Mozilla. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events

[7] Profitina, "Kebijakan Privasi" and "Syarat dan Ketentuan" (Privacy Policy and Terms and Conditions, v1.1, effective Aug. 6, 2026). [Online]. Available: https://app.profitina.com/legal/privacy

[8] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*, Oct. 17, 2022.

[9] J. Nielsen, "Response Times: The 3 Important Limits," Nielsen Norman Group, 1993 (updated 2014). [Online]. Available: https://www.nngroup.com/articles/response-times-3-important-limits/

[10] MDN Web Docs, "AbortController," Mozilla. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/AbortController

[11] Python Software Foundation, "http.server — HTTP servers," *The Python Standard Library*. [Online]. Available: https://docs.python.org/3/library/http.server.html

[12] Oracle, "Package com.sun.net.httpserver," *Java SE 25 & JDK 25 API Specification*, 2025. [Online]. Available: https://docs.oracle.com/en/java/javase/25/docs/api/jdk.httpserver/com/sun/net/httpserver/package-summary.html

[13] Oracle, "Class HttpClient," *Java SE 25 & JDK 25 API Specification*, 2025. [Online]. Available: https://docs.oracle.com/en/java/javase/25/docs/api/java.net.http/java/net/http/HttpClient.html

[14] R. Fielding, M. Nottingham, and J. Reschke, "HTTP/1.1," IETF RFC 9112, Jun. 2022.

[15] Microsoft, "Playwright for Python." [Online]. Available: https://playwright.dev/python/

Bab 5

Menjaga Komunitas Tetap Aman: Link Safety, Moderasi, Main Adil

Di akhir bab ini Anda akan mampu membongkar sebuah tautan dan menunjuk tempat-tempat penipuan bersembunyi di dalamnya; menjelaskan mengapa Profitina menyaring tautan secara berlapis, dengan lantai aturan yang deterministik di bawah dan pendapat AI lokal opsional di atasnya, serta mengapa tidak satu pun lapisan itu berjalan selagi seseorang menunggu untuk membaca umpan; mengukur penyaring apa pun secara jujur dengan matriks kebingungan (*confusion matrix*), presisi, *recall*, F1, dan laju positif palsu, serta menyatakan seberapa banyak yang dapat dan tidak dapat dibuktikan oleh tolok ukur kecil; menguraikan apa yang boleh dan tidak boleh dilakukan moderator kanal, dan mengapa batas-batas itu justru intinya; serta menjelaskan, pada tingkat kotak dan panah, bagaimana sistem hadiah menjaga hadiahnya tetap untuk manusia. Di lab Anda akan menulis penyaring tautan berbasis aturan milik Anda sendiri dalam Python dan Java, dengan aturan pilihan Anda sendiri, lalu mengevaluasinya pada himpunan sampel berlabel yang dapat Anda baca baris demi baris, diakhiri dengan matriks kebingungan, tawar-menawar presisi–*recall*, dan analisis setiap kesalahan.

5.1 Posisi Kita Sekarang

Bab 4 membangun komunitas yang memperbarui dirinya sendiri: kiriman muncul begitu ditulis, penghitung naik selagi halaman terbuka, dan utas tumbuh di depan mata setiap pembacanya. Setiap sifat itu adalah hadiah bagi penipu. Tautan yang dikirim di umpan langsung sampai ke setiap halaman yang terbuka dalam satu detik. Pesan yang tampak seperti pengumuman staf dibaca ratusan orang sebelum ada yang mempertanyakannya. Hadiah untuk mengirim, bagi sebuah program, hanyalah fungsi yang dapat dipanggil sesering mungkin.

Bab ini membahas tiga pertahanan yang membuat komunitas Bab 4 aman dipakai. Yang pertama adalah **keamanan tautan** (*link safety*): setiap tautan dalam kiriman disaring sebelum siapa pun dapat mengekliknya. Yang kedua adalah **moderasi**: orang-orang dengan wewenang sempit dan terlihat menjaga ruang yang menjadi tanggung jawabnya. Yang ketiga adalah **main adil** (*fair play*): sistem hadiah dirancang agar membayar manusia untuk kontribusi yang berguna dan tidak membayar bot sama sekali. Bab ini membahasnya dalam urutan itu, dan paling lama pada yang pertama, karena itulah yang dapat Anda bangun dan ukur sendiri dalam satu sore.

Seperti di Bab 4, Profitina dijelaskan pada tingkat kotak: lapisan apa saja yang ada, dalam urutan apa, dan mengapa. Aturan, bobot, ambang, dan *prompt* yang sebenarnya adalah milik Profitina sendiri. Penyaring lab adalah milik kita, dibangun dari aturan buku teks yang dapat ditemukan siapa pun dalam literatur keamanan, dan dievaluasi pada himpunan sampel yang ditulis untuk buku ini.

5.2 Mengapa Komunitas Perlu Dilindungi

5.2.1 Tiga angka

Gambar 5.1 menjajarkan tiga angka terbaru. Masing-masing mengukur hal yang berbeda, dan bersama-sama ketiganya menggambarkan lingkungan tempat setiap platform komunitas Indonesia hidup.

Gambar 5.1 Tiga ukuran lingkungan ancaman pada 2026: serangan phishing yang teramati di seluruh dunia dalam satu kuartal, laporan penipuan yang diterima Indonesia Anti-Scam Centre sejak dibuka, dan porsi lalu lintas web yang bersifat otomatis.

- **Phishing terus tumbuh.** Anti-Phishing Working Group (APWG) mengamati 971.181 serangan *phishing* pada kuartal pertama 2026, 13,8 % lebih banyak daripada 853.244 pada kuartal keempat 2025. Laporannya, yang terbit 27 Mei 2026, juga mendapati volume ancaman naik di setiap platform media sosial yang dilacaknya, dengan penipuan mencakup 27,1 % dan peniruan identitas 43,8 % dari ancaman yang tercatat di sana, serta *phishing* lewat suara dan SMS bersama-sama naik 15 % dalam satu kuartal [1].
- **Orang Indonesia disasar dalam skala besar.** Indonesia Anti-Scam Centre (IASC), yang dijalankan Otoritas Jasa Keuangan (OJK) bersama bank dan lembaga lain, menerima 636.014 laporan penipuan sejak diluncurkan pada 22 November 2024 hingga 30 Juli 2026. Dalam periode itu IASC memblokir 604.923 rekening yang berisi Rp723,7 miliar dan mengembalikan Rp204,3 miliar kepada korban [2].
- **Sebagian besar web bukan manusia.** *Bad Bot Report 2026* dari Imperva, yang terbit 29 April 2026, mendapati bahwa lalu lintas otomatis mencakup 53 % lalu lintas web pada 2025, melampaui lalu lintas manusia (47 %) [3].

Platform komunitas berdiri tepat di titik pertemuan ketiganya. Anggotanya orang Indonesia biasa, banyak di antaranya pemilik usaha kecil yang menangani pembayaran setiap hari; umpannya tempat sebuah tautan dapat menjangkau ratusan orang dalam hitungan detik; dan hadiahnya adalah undangan bagi otomatisasi.

5.2.2 Tiga ancaman, tiga pertahanan

Menamai ancaman dengan tepat itu membantu, karena masing-masing memerlukan pertahanan yang berbeda.

1. **Tautan berbahaya.** Sebuah kiriman membawa tautan ke halaman yang mencuri kata sandi, kode OTP, dompet, atau nomor kartu, atau yang memasang sesuatu yang merugikan. Pertahanannya adalah *keamanan tautan*: saring tautan sebelum dapat diklik.
2. **Penyalahgunaan oleh orang.** Pelecehan, peniruan staf, banjir kiriman di luar topik, penipuan tanpa tautan sama sekali (“kirimkan kode OTP yang baru saja kami kirim”). Perangkat lunak

menangkap sebagian; orang menangkap sisanya. Pertahanannya adalah *moderasi* dengan wewenang sempit yang dicatat, dan *pelaporan* yang dapat dipakai setiap anggota.

3. **Penyalahgunaan oleh otomatisasi.** Skrip yang mengirim, menyukai, dan merujuk secara massal untuk "menernak" hadiah. Pertahanannya adalah *main adil*: pastikan hanya manusia terverifikasi yang memperoleh hadiah, kontribusi remeh atau berulang tidak memperoleh apa-apa, dan penyalahgunaan yang ditemukan belakangan ditarik kembali.

Tidak ada satu pertahanan pun yang mencakup ketiganya. Penyaring tautan yang sempurna tidak berdaya menghadapi penipuan OTP tanpa tautan; moderator terbaik tidak dapat membaca seribu kiriman per menit; dan main adil melindungi hadiah, bukan pembaca. Sisa bab ini membahasnya satu per satu.

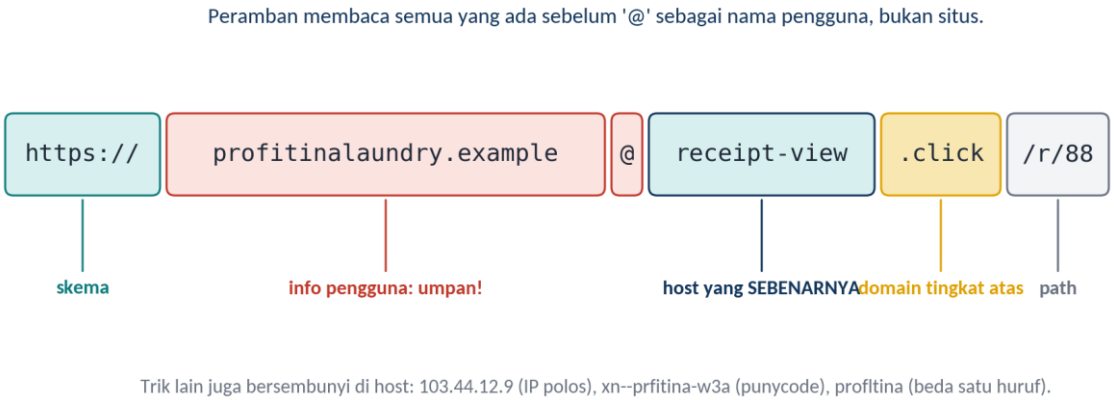
5.3 Anatomi Tautan, dan Tempat Trik Bersembunyi

5.3.1 Bagian-bagian URL

Alamat web, atau URL, memiliki struktur yang didefinisikan dalam RFC 3986 [4]. Untuk keperluan kita, lima bagian penting:

- **skema** (*scheme*), misalnya `https`, yang menyatakan cara mengambil sumber daya;
- **bagian situs**, yaitu semua yang berada di antara `//` dan `/`, `?`, atau `#` berikutnya, yang boleh memuat *informasi pengguna* yang diakhiri `@`, lalu **host**, lalu `:port` opsional;
- **host**, nama mesin, yang tersusun atas *label* yang dipisahkan titik, dan label terakhirnya adalah **domain tingkat atas** (*top-level domain*, TLD) seperti `com` atau `id`;
- **path**, yang menamai sumber daya di mesin itu;
- secara opsional *query* sesudah `?` dan *fragment* sesudah `#`.

Fakta tunggal yang paling penting untuk keamanan tautan adalah bahwa **hanya host yang menentukan server organisasi mana yang Anda tuju**. Semua yang lain, termasuk kata-kata yang mirip nama merek, dapat dipilih bebas oleh siapa pun yang menyiapkan halamannya. **Gambar 5.2** memperlihatkan tautan yang dibuat untuk memanfaatkan persis hal itu.



Gambar 5.2 Tautan yang menyembunyikan host sebenarnya. Semua yang ada sebelum `@` adalah informasi pengguna, sekadar umpan; host-nya adalah `receipt-view.click`. Baris bawah menyebut tiga tempat lain trik bersembunyi di dalam host.

Pembaca yang melihat `https://profitinalaundry.example@...` membaca "Profitina Laundry". Peramban membaca sebuah nama pengguna, `profitinalaundry.example`, diikuti host yang sebenarnya, `receipt-view.click`. Nama laundry itu hanya hiasan.

5.3.2 Trik-trik yang biasa

Tabel 5.1 mendaftar trik yang paling sering dipakai tautan *phishing*, masing-masing dengan contoh yang ditulis untuk buku ini. Tidak satu pun yang baru; semuanya bahan klasik setiap mata ajar keamanan, dan mengenalinya adalah langkah pertama untuk membangun penyaring.

Tabel 5.1 Trik yang umum dalam tautan berbahaya, dengan contoh rekaan.

Trik	Contoh	Mengapa berhasil
Alamat IP polos sebagai host	`http://103.44.12.9/pay`	Tidak ada nama untuk diperiksa; tidak ada yang mengingat angka
Informasi pengguna sebelum `@`	`https://brand.example@evil.click/`	Merek tampil lebih dulu; host sebenarnya datang sesudah `@`
Host punycode (terinternasionalisasi)	`https://xn--prfitina-w3a.com/`	Tampil sebagai kembaran dengan huruf beraksen
Beda satu huruf dari merek	`https://profltina.com/`	`l` untuk `i` sulit dilihat di banyak fon
Nama merek di domain milik orang lain	`https://profitina-support.xyz/`	Merek hanya sebuah label; domainnya bukan milik merek
Subdomain bertumpuk	`https://profitina.com.account.secure-login.buzz/`	Awalnya tampak benar; domain sebenarnya di ujung
Pemendek URL	`bit.ly/pl-prize2026`	Menyembunyikan tujuan sepenuhnya
Tanpa HTTPS	`http://voucher-laundry.icu`	Lalu lintas dapat dibaca atau diubah di perjalanan
TLD murah yang sering disalahgunakan	`.xyz`, `.top`, `.click`	Biaya rendah memudahkan domain sekali pakai
Desakan dalam teks	“verify within 24 hours”, “claim your prize”	Tekanan membuat orang berhenti memeriksa

Dua baris perlu penjelasan.

Punycode. Nama domain semula terbatas pada huruf a–z, angka, dan tanda hubung. Agar nama dalam aksara lain dapat dipakai, nama domain terinternasionalisasi diubah ke abjad terbatas itu dengan pengodean bernama Punycode, dan hasilnya diawali `xn--` [5]. Nama `prófitina.com`, dengan *o* beraksen, menjadi `xn--prfitina-w3a.com`. Peramban dapat menampilkan versi beraksennya, yang sekilas tidak dapat dibedakan dari nama asli. Konsorsium Unicode memelihara satu standar utuh tentang karakter yang *membingungkan* (*confusable*) semacam itu, Unicode Technical Standard #39, yang versi 18.0.0-nya terbit pada 27 Agustus 2026 [6]. Komunitas usaha

Indonesia jarang memerlukan nama host beraksen, sehingga memperlakukan `xn--` sebagai tanda bahaya yang kuat hampir tidak ada ruginya.

TLD murah. Tidak ada domain tingkat atas yang jahat dengan sendirinya, dan usaha yang sah memakai semuanya. Namun beberapa akhiran dijual begitu murah sehingga digemari untuk domain sekali pakai. Karena itu, aturan yang melihat TLD harus *lunak*: petunjuk yang digabungkan dengan petunjuk lain, tidak pernah menjadi putusan sendirian. Lab akan memperlihatkan dengan tepat apa yang terjadi ketika sebuah percetakan lokal yang sah memakai akhiran semacam itu.

5.3.3 Sinyal keras dan sinyal lunak

Trik-trik itu terbagi menjadi dua kelompok, dan perbedaannya menentukan seluruh rancangan penyaring.

> **DEFINISI.** *Sinyal keras* adalah sifat tautan yang hampir tidak pernah dimiliki tautan sah di komunitas, sehingga satu kemunculan sudah cukup untuk memblokir. *Sinyal lunak* adalah sifat yang mencurigakan tetapi juga muncul pada tautan sah, sehingga hanya boleh dihitung bersama sinyal lunak lainnya.

Alamat IP polos, informasi pengguna sebelum `@`, host punycode, atau host yang beda satu huruf dari merek yang dilindungi adalah sinyal keras di komunitas seperti milik Profitina: anggota yang jujur hampir tidak pernah mengirimnya. Pemendek URL, HTTP biasa, TLD murah, subdomain dalam, atau kata-kata mendesak adalah sinyal lunak: semuanya muncul di kiriman jujur setiap hari (“kesempatan terakhir ikut kelompok belajar malam ini!”). Penyaring yang memblokir pada sinyal lunak apa pun akan memblokir anggota jujur; penyaring yang mengabaikan sinyal lunak akan meloloskan penipuan yang berhati-hati menghindari sinyal keras. Lab di Subbab 5.8 membuat ketegangan itu dapat diukur.

5.3.4 Lima detik untuk memeriksa

Perangkat lunak adalah satu lini pertahanan; pembaca adalah lini lainnya. Setiap aturan di subbab ini juga dapat diterapkan seseorang dalam beberapa detik, dan komunitas lebih aman bila anggotanya tahu caranya. Sebelum mengeklik tautan di umpan mana pun, bukan hanya milik Profitina, ada lima pertanyaan yang patut diajukan secara berurutan.

1. *Di mana host berakhir?* Temukan `/` pertama sesudah `//`, lihat tepat sebelumnya, dan baca dua label terakhir. Itulah, bukan bagian awalnya, server milik siapa yang akan Anda tuju.
2. *Adakah `@` sebelum titik itu?* Jika ada, semua yang sebelumnya hanyalah hiasan.
3. *Apakah berupa angka atau diawali `xn--`?* Usaha yang jujur jarang mengirim keduanya.
4. *Apakah tautannya dipendekkan?* Maka Anda sama sekali tidak dapat melihat tujuannya; tanyakan kepada pengirim, atau buka hanya bila Anda sepenuhnya memercayai pengirimnya.
5. *Apakah pesan ini memburu-buru saya?* “Dalam 24 jam”, “kesempatan terakhir”, “akun Anda diblokir”: tekanan adalah alat paling andal bagi penipu, karena menghentikan persis jenis pemeriksaan ini.

Pertanyaan-pertanyaan ini tidak menggantikan penyaring, dan penyaring tidak menggantikannya. Penyaring melihat setiap tautan tetapi tidak memahami konteks; pembaca memahami konteks tetapi bisa lelah. Bersama-sama keduanya menangkap lebih banyak daripada masing-masing sendirian, dan itulah argumen yang sama, dalam skala manusia, dengan yang diajukan Subbab 5.4 untuk lapisan-lapisan perangkat lunak.

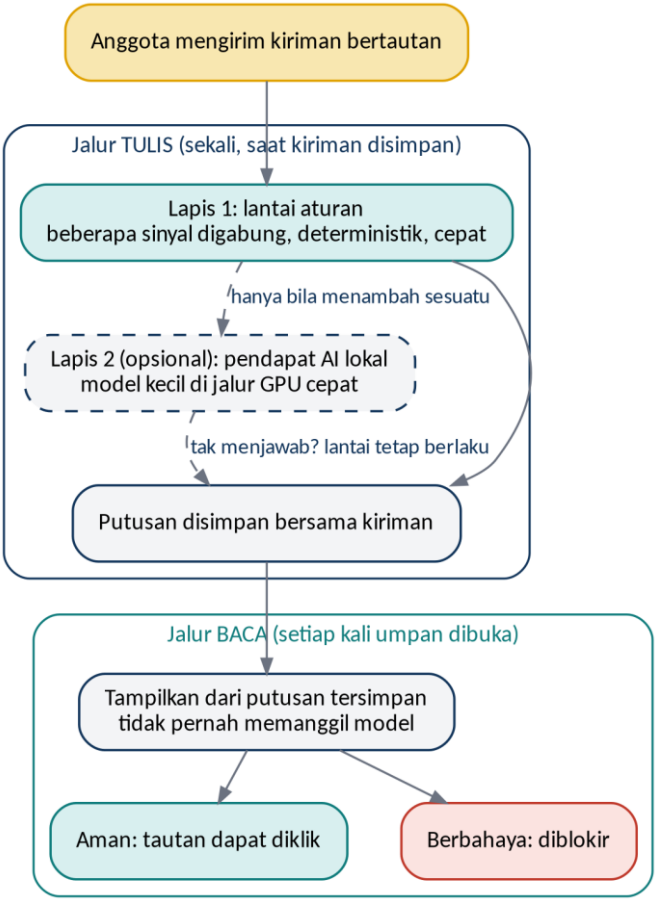
5.4 Keamanan Tautan sebagai Lapisan

5.4.1 Dua lapisan

Profitina menyaring setiap tautan yang dikirim di komunitas. Penyaringannya memiliki dua lapisan [7], [8]:

1. **Lantai aturan** (*rule floor*). Sekumpulan aturan yang menggabungkan beberapa sinyal dari jenis-jenis di atas menjadi sebuah putusan. Lantai ini deterministik (tautan yang sama selalu mendapat putusan yang sama), cepat (tidak memerlukan jaringan maupun model), dan dapat dijelaskan (setiap putusan disertai aturan-aturan yang terpicu).
2. **Pendapat AI lokal yang opsional.** Di atas lantai itu, model bahasa kecil yang berjalan secara lokal di jalur GPU cepat platform dapat memberi pendapat tentang sebuah tautan dan teks di sekitarnya. Model itu menangkap jenis niat yang sulit ditangkap aturan, seperti pesan ramah yang sebenarnya meminta sebuah kode. Model berjalan di mesin Profitina sendiri, seperti semua AI-nya, sehingga pesan tidak pernah meninggalkan server.

Tautan yang aman ditampilkan sebagai tautan yang dapat diklik; tautan berbahaya diblokir [7]. **Gambar 5.3** menggambar kedua lapisan itu dan, sama pentingnya, *kapan* keduanya berjalan.



Gambar 5.3 Keamanan tautan sebagai lapisan. Kedua lapisan berjalan sekali, di jalur tulis, dan putusannya disimpan bersama kiriman; menampilkan umpan hanya membaca putusan tersimpan dan tidak pernah memanggil model.

5.4.2 Mengapa lantai lebih dulu

Menyerahkan seluruh pekerjaan kepada model memang menggoda. Empat alasan menentangnya.

- **Ketersediaan.** Model bisa sibuk, sedang dinyalakan ulang, atau lebih lambat daripada biasanya. Pemeriksaan keamanan yang bergantung padanya akan menghambat pengiriman atau diam-diam meloloskan tautan. Lantai selalu menjawab. Jika model tidak menjawab tepat waktu, putusan lantai yang berlaku.
- **Keterjelasan.** Ketika sebuah kiriman ditolak, anggota berhak mendapat alasan. "Host tautan ini berupa alamat IP" adalah alasan; "kata modelnya begitu" bukan.
- **Keterujian.** Lantai yang deterministik dapat diuji dengan himpunan contoh yang tetap, dan perubahan pada sebuah aturan dapat diukur sebelum dirilis. Tolok ukur publik Profitina di Subbab 5.5.3 adalah pengukuran semacam itu.
- **Biaya.** Aturan berbiaya mikrodetik. Panggilan model, bahkan pada model kecil dan GPU cepat, berbiaya berlipat-lipat lebih besar.

Karena itu model adalah *tambahan*: model boleh memperhatikan apa yang terlewat oleh aturan, tetapi platform tidak bergantung pada kehadirannya.

5.4.3 Tidak pernah di jalur baca

Bab 4 menceritakan rilis September 2026, ketika sebuah halaman umpan ternyata memanggil model saat menampilkan, enam panggilan pada satu halaman uji, dan memerlukan 868 ms untuk tampil. Pelajaran itu berlaku dengan kekuatan khusus bagi keamanan tautan. Tautan ditulis sekali dan mungkin dibaca ribuan kali. Jika penyaringan berjalan ketika kiriman *ditulis*, penyaringan berjalan sekali, selagi satu anggota menunggu kirimannya sendiri; putusannya disimpan bersama kiriman. Jika penyaringan berjalan ketika kiriman *dibaca*, penyaringan berjalan ribuan kali, selagi semua orang menunggu. Sejak rilis itu, pendapat model hanya diminta di jalur tulis, dan jalur baca hanya membaca putusan yang tersimpan [8].

> **WAWASAN KUNCI.** Putuskan sekali, ketika data masuk ke sistem; baca putusannya berkali-kali. Aturan yang sama inilah yang membuat basis data menyimpan kolom hasil hitungan dan server web menyimpan halaman di *cache*, dan inilah alasan pemeriksaan keamanan dapat menyeluruh tanpa membuat umpan lambat.

5.4.4 Lapisan kedua yang lebih senyap: tampilkan teks sebagai teks

Menyaring tautan adalah separuh dari menampilkan isi pengguna dengan aman. Separuh lainnya adalah cara halaman menyisipkan isi itu. Halaman lab Bab 4 menyisipkan setiap kiriman dengan `textContent`, tidak pernah dengan `innerHTML`, sehingga kiriman yang berisi `<script>` menampilkan karakter-karakter itu alih-alih menjalankannya. Ini bukan penyaring yang dapat dikelabui; ini aturan tentang bagaimana peramban diminta memperlakukan teks, dan aturan ini berlaku untuk setiap kiriman, entah ada yang disaring atau tidak.

5.5 Mengukur Penyaring: Matriks Kebingungan

5.5.1 Empat jenis hasil

Penyaring melihat sebuah pesan dan menyatakan *tandai* (blokir) atau *loloskan*. Pesan itu sesungguhnya *penipuan* atau *normal*. Dari situ ada empat kombinasi, dan setiap evaluasi atas setiap pengklasifikasi dimulai dengan menghitungnya [9], [10].

> **DEFINISI.** Sebut penipuan sebagai *positif*. Maka **positif benar** (*true positive*, TP) adalah penipuan yang ditandai; **negatif palsu** (*false negative*, FN) adalah penipuan yang diloloskan; **positif palsu** (*false positive*, FP) adalah pesan normal yang ditandai; dan **negatif benar** (*true*

> *negative*, TN) adalah pesan normal yang diloloskan. Disusun dalam tabel 2 × 2 dengan kebenaran sebagai baris dan putusan sebagai kolom, keempat hitungan ini membentuk *matriks kebingungan*.

Contoh mungil membuat keempat hitungan itu konkret. Misalkan sebuah penyaring melihat sepuluh pesan: empat penipuan dan enam pesan normal. Penyaring itu menandai tiga penipuan dan satu pesan normal. Maka $TP = 3$ (penipuan yang ditandai), $FN = 1$ (penipuan yang terlewat), $FP = 1$ (pesan normal yang keliru ditandai), dan $TN = 5$ (pesan normal yang diloloskan). Keempat hitungan selalu berjumlah sama dengan banyaknya pesan, $3 + 1 + 1 + 5 = 10$, dan kedua barisnya berjumlah sama dengan banyaknya penipuan dan pesan normal, $3 + 1 = 4$ dan $1 + 5 = 6$. Memeriksa jumlah-jumlah itu adalah cara tercepat untuk menangkap salah hitung dalam evaluasi apa pun, termasuk evaluasi Anda sendiri. Kebiasaan kecil ini patut dipelihara sejak awal, karena kesalahan penghitungan dalam evaluasi jauh lebih sering terjadi daripada yang diakui orang, dan satu angka yang tertukar antara baris dan kolom sudah cukup untuk membalik kesimpulan tentang sebuah penyaring.

Kedua jenis kesalahan memiliki biaya yang sangat berbeda. Negatif palsu membiarkan penipuan sampai ke anggota. Positif palsu memblokir anggota jujur, yang kemudian seolah-olah diberi tahu bahwa platform menganggapnya penipu. Penyaring yang baik menekan keduanya, dan evaluasi yang baik melaporkan keduanya secara terpisah.

5.5.2 Lima ukuran dari empat hitungan

Dari keempat hitungan itu kita menghitung beberapa ukuran baku. **Presisi** bertanya: dari semua yang ditandai, berapa yang benar-benar penipuan?

$$\text{presisi} = \frac{TP}{TP + FP} \tag{5.1}$$

Recall, yang juga disebut sensitivitas atau laju positif benar, bertanya: dari semua penipuan, berapa yang ditandai?

$$\text{recall} = \frac{TP}{TP + FN} \tag{5.2}$$

Presisi dan *recall* saling tarik ke arah berlawanan: tandai segalanya dan *recall* sempurna sementara presisi runtuh; hampir tidak menandai apa pun dan presisi mungkin sempurna sementara *recall* runtuh. **Skor F1** menggabungkan keduanya menjadi satu angka, yaitu rata-rata harmoniknya:

$$F_1 = \frac{2 \cdot \text{presisi} \cdot \text{recall}}{\text{presisi} + \text{recall}} = \frac{2\,TP}{2\,TP + FP + FN} \tag{5.3}$$

Bentuk kedua diperoleh dari bentuk pertama dengan menyulihkan Persamaan (5.1) dan Persamaan (5.2) lalu mengalikan pembilang dan penyebut dengan $(TP + FP)(TP + FN)/TP$. Rata-rata harmonik menghukum ketimpangan: penyaring dengan presisi 1,0 dan *recall* 0,1 memiliki rata-rata aritmetika 0,55 tetapi F_1 hanya $2 \times 0.1/1.1 \approx 0.18$.

Akurasi adalah porsi seluruh putusan yang benar:

$$\text{akurasi} = \frac{TP + TN}{TP + FN + FP + TN} \tag{5.4}$$

dan **laju positif palsu** (*false positive rate*, FPR) adalah porsi pesan normal yang keliru ditandai:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{5.5}$$

Akurasi adalah ukuran yang paling dulu dikutip orang dan paling tidak layak dipercaya. Jika 99 dari setiap 100 pesan normal, "penyaring" yang meloloskan segalanya memiliki akurasi 99 % dan tidak menangkap apa pun. Bagi komunitas, dua angka yang penting adalah *recall* (berapa penipuan yang lolos) dan FPR (berapa anggota jujur yang terblokir).

5.5.3 Tolok ukur publik Profitina

Profitina menerbitkan satu hasil tolok ukur untuk lantai aturannya: pada himpunan 28 teks penipuan yang realistis dan 30 teks normal, lantai itu menangkap 27 dari 28 penipuan dengan 0 positif palsu pada 30 teks normal [7], [8]. Separuh kiri **Gambar 5.4** menampilkan hasil itu sebagai matriks kebingungan: $TP = 27, FN = 1, FP = 0, TN = 30$.

Gambar 5.4 Dua matriks kebingungan. Kiri: tolok ukur Profitina yang diterbitkan untuk lantai aturannya (27 dari 28 penipuan tertangkap, 0 positif palsu pada 30 teks normal). Kanan: penyaring lab Subbab 5.8 pada 60 sampel buku ini dengan $k = 2$.

Mari kerjakan rumusnya, selangkah demi selangkah:

- presisi menurut Persamaan (5.1): $27/(27 + 0) = 1.000$;
- *recall* menurut Persamaan (5.2): $27/(27 + 1) = 27/28 \approx 0.964$;
- F_1 menurut Persamaan (5.3): $2 \times 27/(54 + 0 + 1) = 54/55 \approx 0.982$;
- akurasi menurut Persamaan (5.4): $(27 + 30)/58 = 57/58 \approx 0.983$;
- FPR menurut Persamaan (5.5): $0/30 = 0$.

Angka-angka ini baik. Dua subbagian berikut membahas cara membacanya dengan benar.

5.5.4 Apa yang dibuktikan nol dari tiga puluh

Nol positif palsu pada 30 teks normal *tidak* membuktikan bahwa laju positif palsunya nol. Yang dibuktikannya adalah bahwa laju itu mungkin kecil. Seberapa kecil? Misalkan laju sebenarnya p. Peluang tidak melihat satu pun positif palsu pada n teks normal yang saling bebas adalah $(1 - p)^n$. Nilai p terbesar yang peluang itu masih sekurang-kurangnya 5 % adalah batas atas kepercayaan 95 %, yang diperoleh dengan menyelesaikan $(1 - p)^n = 0.05$:

$$p_{\text{atas}} = 1 - 0.05^{1/n} \approx \frac{3}{n}. \tag{5.6}$$

Hampiran di sebelah kanan adalah *aturan tiga* (*rule of three*) yang klasik [11]: hampiran itu mengikuti dari $\ln 0.05 \approx -3$ dan $\ln(1-p) \approx -p$ untuk p kecil. Untuk $n = 30$ batas eksaknya $1 - 0.05^{1/30} = 1 - e^{-0.0999} \approx 0.095$, dan aturan tiga memberikan $3/30 = 0.10$. Dalam kata-kata: setelah 0 positif palsu pada 30 teks normal, kita dapat menyatakan dengan kepercayaan 95 % bahwa laju positif palsunya di bawah sekitar 10 %. Untuk menekan batas itu ke bawah 1 %, Persamaan (5.6) menyatakan kita memerlukan sekitar 300 teks normal tanpa satu pun positif palsu. Tolok ukur kecil jujur tentang arah, bukan tentang ketelitian.

5.5.5 Mengapa laju positif palsu lebih penting daripada kelihatannya

Tolok ukur biasanya *seimbang*: kira-kira sama banyak penipuan dan teks normal. Umpan sungguhan tidak demikian. Misalkan sebagian π dari semua pesan yang memuat tautan adalah penipuan, yang disebut *prevalensi*. Maka di antara semua pesan yang ditandai, bagian yang sungguh penipuan, yaitu presisi yang benar-benar dialami anggota, adalah

$$\text{PPV} = \frac{\text{recall} \cdot \pi}{\text{recall} \cdot \pi + \text{FPR} \cdot (1 - \pi)}, \tag{5.7}$$

yang mengikuti dari aturan Bayes: dari M pesan, $\text{recall} \cdot \pi M$ adalah penipuan yang ditandai dan $\text{FPR} \cdot (1-\pi)M$ adalah pesan normal yang ditandai. PPV adalah singkatan *positive predictive value* (nilai prediktif positif).

Ambil penyaring hipotetis dengan *recall* 0,833 dan FPR 0,100 (kebetulan inilah angka penyaring lab di Subbab 5.8) dan umpan tempat 2 % pesan bertautan adalah penipuan. Menurut Persamaan (5.7),

$$\text{PPV} = \frac{0.833 \times 0.02}{0.833 \times 0.02 + 0.100 \times 0.98} = \frac{0.01666}{0.01666 + 0.098} \approx 0.145.$$

Hanya sekitar satu dari tujuh pesan yang ditandai yang benar-benar penipuan; enam lainnya anggota jujur yang terblokir. Pada tolok ukur yang seimbang, penyaring yang sama memiliki presisi 0,893. Perbedaannya sepenuhnya disebabkan prevalensi, dan itulah sebabnya platform harus menekan laju positif palsunya sangat rendah sebelum boleh memblokir secara otomatis, dan mengapa mengukur pada lebih banyak teks normal sepadan dengan usahanya.

> **JEBAKAN.** Jangan pernah membandingkan angka dua penyaring yang diukur pada himpunan sampel berbeda. Penyaring yang meraih 0,96 pada sampel mudah bisa lebih buruk daripada yang meraih 0,83 pada sampel sulit. Angka lab dan tolok ukur Profitina pada **Gambar 5.4** disandingkan untuk memperlihatkan dua matriks kebingungan, bukan untuk memeringkat dua penyaring.

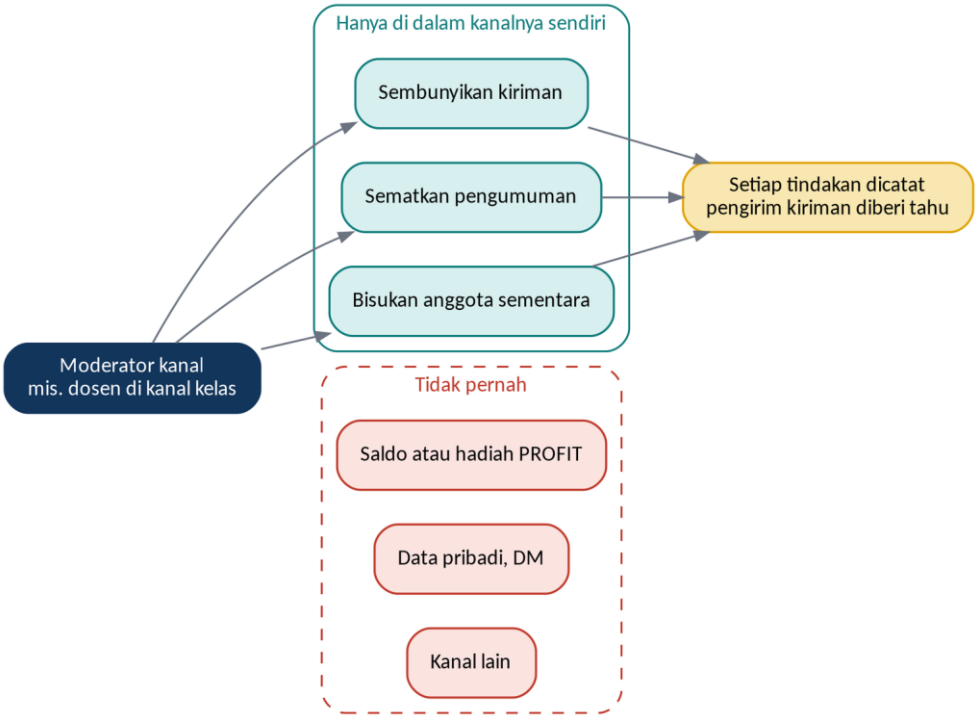
5.6 Moderator: Pertimbangan Manusia, Wewenang Sempit

5.6.1 Yang tidak dapat dilakukan perangkat lunak

Penyaring tautan menangkap tautan. Penyaring tidak memperhatikan bahwa seorang anggota berulang kali mengejek teman sekelasnya, bahwa sebuah utas telah bergeser menjadi tempat berjualan, atau bahwa "staf" yang meminta kode OTP sama sekali tidak menyertakan tautan dalam pesannya. Ini pertimbangan tentang orang dan konteks, dan hakim terbaiknya adalah orang yang mengenal ruang itu.

5.6.2 Moderator bercakupan kanal

Sejak 13 Agustus 2026 Profitina memiliki *moderator bercakupan kanal*, dibangun agar sebuah kelas universitas dapat menjalankan kanalnya sendiri [7]. Seorang moderator ditunjuk untuk satu kanal, dan hanya di dalam kanal itu ia boleh melakukan tiga hal: **menyembunyikan** kiriman, **menyematkan** pengumuman, dan **membisukan** anggota untuk waktu terbatas. **Gambar 5.5** memperlihatkan wewenang itu dan, sama jelasnya, apa yang berada di luarnya.



Gambar 5.5 Apa yang dapat dan tidak dapat dilakukan moderator kanal. Tiga wewenang di dalam satu kanal, setiap tindakan dicatat dan pengirim kiriman diberi tahu; tanpa akses ke PROFIT, data pribadi, pesan langsung, atau kanal lain.

Batas-batas itu disengaja, dan masing-masing menjawab satu risiko.

- **Hanya satu kanal.** Dosen yang memoderasi kanal kelas tidak memiliki wewenang di tempat lain. Moderator yang dapat bertindak di seluruh platform akan menjadi administrator kedua tanpa akuntabilitas seorang administrator.
- **Tanpa PROFIT, tanpa data pribadi.** Moderator tidak pernah menyentuh saldo atau hadiah dan tidak pernah melihat informasi pribadi seperti alamat surel atau DM. Moderasi adalah soal apa yang dikatakan di ruang publik, bukan soal siapa seseorang atau apa yang dimilikinya.
- **Menyembunyikan, bukan menghapus.** Wewenang moderator adalah mengeluarkan kiriman dari pandangan, langkah yang lebih lunak daripada memusnahkannya.

- **Membisukan sementara, bukan melarang.** Pembisuan sementara mendinginkan suasana; pembisuan tidak mengakhiri keanggotaan seseorang.
- **Semuanya dicatat, dan pengirim diberi tahu.** Setiap tindakan moderasi direkam, dan orang yang kirimannya disembunyikan atau yang dibisukan diberi notifikasi.

Aturan-aturan ini selaras dengan *Santa Clara Principles on Transparency and Accountability in Content Moderation*, yang disusun organisasi masyarakat sipil dan para peneliti, yang versi keduanya menuntut aturan yang jelas, pemberitahuan kepada orang yang terdampak, dan kemungkinan banding [12]. Platform kecil tidak dapat menjalankan departemen banding yang besar, tetapi dapat membuat setiap tindakan terlihat oleh orang yang terdampak dan dapat dipulihkan sejak rancangannya.

5.6.3 Pelaporan, dan salinan beku

Setiap anggota dapat melaporkan kiriman, dan setiap penerima DM dapat melaporkan pesan. Seperti dijelaskan Bab 4, administrator tidak dapat membaca DM; laporan menciptakan *salinan beku* dari pesan yang dilaporkan, satu-satunya bagian percakapan privat yang pernah dapat dilihat peninjau, disimpan persis seperti adanya agar tidak dapat disunting sesudahnya. Pemblokiran berlaku seketika dan tidak memerlukan persetujuan siapa pun.

> **CONTOH.** Di sebuah kanal kelas, seorang anggota mengirim "Jawaban tugas 2, DM saya untuk berkasnya" tanpa tautan. Penyaring tautan tidak punya apa pun untuk disaring. Dua teman sekelas melaporkan kiriman itu. Dosen, yang memoderasi kanal, menyembunyikannya dan menyematkan pengingat tentang aturan kelas. Pengirim diberi tahu bahwa kirimannya disembunyikan, dan tindakan itu muncul di log moderasi. Saldo PROFIT siapa pun tidak disentuh, dan tidak ada yang berubah di luar kanal itu.

5.6.4 Aturan yang dapat dimoderasi

Moderator hanya dapat bersikap adil bila aturan yang ditegakkannya ditulis sedemikian rupa sehingga orang yang wajar dapat menerapkannya. Kanal-kanal kelas memberi pelajaran sederhana di sini: aturan yang disematkan di sebuah kanal paling baik bekerja bila setiap aturan menyebut sebuah *perilaku*, bukan sikap, dan menyatakan apa yang terjadi bila aturan itu dilanggar. "Saling menghormati" adalah nilai yang baik tetapi aturan yang buruk, karena dua orang akan berbeda pendapat tentang apakah sebuah lelucon sudah kelewatan. "Dilarang mengirim jawaban tugas sebelum tenggat; kiriman semacam itu disembunyikan" dapat diterapkan dengan cara yang sama oleh moderator mana pun, dan seorang anggota dapat memperkirakan akibatnya sebelum mengirim.

Aturan kanal yang baik memiliki tiga bagian. *Perilaku* yang dilarang atau diminta, dinyatakan sedemikian rupa sehingga dapat dikenali dalam satu kiriman. *Tindakan* yang akan diambil moderator, dipilih dari tiga wewenang di Subbab 5.6.2. Dan *alasan*, dalam satu kalimat pendek, karena anggota jauh lebih sukarela mengikuti aturan yang mereka pahami daripada aturan yang sekadar mereka patuhi. Daftar pendek aturan semacam itu, disematkan di bagian atas kanal, berbuat lebih banyak bagi sebuah ruang daripada perangkat lunak sebanyak apa pun, dan membuat setiap tindakan moderasi mudah dijelaskan dalam notifikasi yang diterima pengirim.

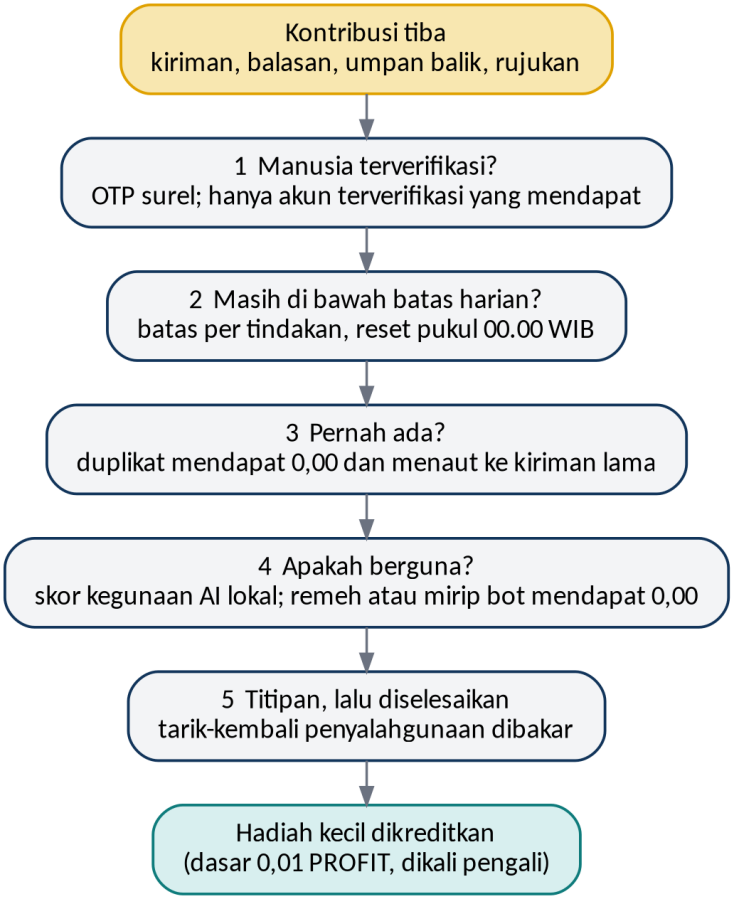
Disiplin yang sama berlaku bagi platform itu sendiri. Aturan Profitina untuk moderatornya, yaitu satu kanal, tiga wewenang, tanpa PROFIT, tanpa data pribadi, setiap tindakan dicatat, ditulis persis dengan gaya ini: masing-masing menyebut apa yang boleh atau tidak boleh dilakukan, dan masing-masing memiliki alasan yang dapat dinyatakan dalam satu kalimat.

5.7 Main Adil: Menjaga Hadiah untuk Manusia

5.7.1 Hadiah mengundang otomatisasi

Setiap ukuran yang diberi hadiah menjadi sasaran. Pengamatan ekonom Charles Goodhart, dalam rumusan umum yang dipopulerkan Marilyn Strathern, menyatakan bahwa "ketika sebuah ukuran menjadi sasaran, ukuran itu tidak lagi menjadi ukuran yang baik" [13]. Bayar untuk kiriman, dan Anda akan mendapat kiriman, entah ada yang menginginkannya atau tidak. Dengan lebih dari separuh lalu lintas web sudah otomatis [3], platform yang membayar kontribusi harus berasumsi bahwa sebagian "anggotanya" adalah program.

Jawaban Profitina adalah seperangkat lapisan main adil, diuraikan dalam *whitepaper* publiknya [7] dan digambar pada tingkat kotak di **Gambar 5.6**. Urutan pada gambar adalah untuk keperluan belajar; yang penting, setiap kontribusi melewati semuanya.



Gambar 5.6 Main adil pada tingkat kotak. Sebuah kontribusi memperoleh hadiah kecil hanya bila berasal dari manusia terverifikasi, masih di bawah batas harian, bukan duplikat, dan dinilai berguna; penyalahgunaan yang ditemukan belakangan ditarik kembali dan dibakar.

5.7.2 Lapisan-lapisannya

Hanya manusia terverifikasi. Pendaftaran memerlukan alamat surel dan kata sandi, dan surel diverifikasi dengan kode sekali pakai (OTP), lapisan anti-bot yang pertama. Hanya akun yang surelnya terverifikasi yang memperoleh apa pun; hadiah rujukan, 0,25 PROFIT untuk masing-masing pihak, baru dibayarkan setelah orang yang dirujuk memverifikasi surelnya, dan hanya sekali per pasangan, seumur hidup. Program rujukan hanya memiliki satu tingkat, karena skema rujukan bertingkat, dalam kata-kata *whitepaper*, "secara struktur mirip skema piramida" [7].

Hadiah kecil, batas harian. Hadiahnya kecil (satuan dasarnya 0,01 PROFIT) dan setiap jenis tindakan memiliki batas harian yang berganti pukul 00.00 WIB. Skrip yang mengirim seribu kali tidak memperoleh lebih banyak daripada orang yang mengirim beberapa kali.

Duplikat tidak memperoleh apa-apa. Kiriman yang mengulang isi sebelumnya memperoleh 0,00 dan menaut ke kiriman sebelumnya, sehingga penyalinan terlihat sekaligus tidak dihargai.

Kegunaan dinilai. Skor kegunaan dari AI lokal mengalikan setiap hadiah; kontribusi yang remeh atau mirip bot memperoleh 0,00. Modelnya berjalan secara lokal, seperti semua AI Profitina.

Titipan, tarik-kembali, dan pembakaran. Hadiah ditahan sebelum diselesaikan; jika penyalahgunaan ditemukan, hadiah ditarik kembali, dan PROFIT yang ditarik kembali dibakar alih-alih dibagikan ulang, salah satu dari empat mekanisme pembakaran koin yang dijelaskan di Bab 12.

Suntingan dihitung ulang. Seperti dijelaskan Bab 4, menyunting kiriman dalam jendela lima belas menitnya menghitung ulang hadiahnya sepenuhnya, sehingga jendela sunting tidak dapat dipakai untuk meningkatkan kiriman setelah dinilai.

Peringkat tidak dapat dibeli. PROFIT yang dibelanjakan untuk layanan tidak pernah menaikkan peringkat anggota; PROFIT yang dibeli atau dihibahkan tidak dapat ditransfer ke orang lain, hanya PROFIT yang diperoleh secara organik; dan hibah diskresioner administrator dipublikasikan serta dikecualikan dari setiap papan peringkat [7].

> **WAWASAN KUNCI.** Tidak satu pun lapisan ini perlu sempurna, karena semuanya saling mengalikan. Bot yang lolos verifikasi surel tetap membentur batas harian; bot yang tetap di bawah batas tetap menghasilkan duplikat atau teks remeh; bot yang menulis teks beragam tetap mendapati hadiahnya ditahan dalam titipan, tempat penyalahgunaan dapat ditarik kembali. Setiap lapisan mengubah keuntungan besar dari otomatisasi menjadi lebih kecil, dan bersama-sama semuanya membuat "ternak" hadiah tidak sepadan dengan usahanya.

5.7.3 Mengapa lapisan saling mengalikan: ilustrasi terhitung

Klaim bahwa lapisan saling *mengalikan* dapat dibuat tepat dengan contoh kecil yang sepenuhnya rekaan; tidak satu pun angkanya milik Profitina. Misalkan sebuah skrip penernak, tanpa pertahanan sama sekali, akan memperoleh sejumlah G per hari. Misalkan sekarang setiap lapisan, bila bertindak sendiri, hanya meloloskan sebagian dari yang sampai kepadanya: verifikasi meloloskan sebagian v dari akun yang coba dibuat skrip, batas harian membiarkan setiap akun yang bertahan menyimpan sebagian c dari yang semestinya diperolehnya, deteksi duplikat dan skor kegunaan bersama-sama membiarkan sebagian u dari kontribusi yang tersisa memperoleh sesuatu, dan tarik-kembali menyisakan sebagian r dari itu di tangan skrip. Jika lapisan-lapisan itu bertindak saling bebas, keuntungan harapan skrip adalah

$$G_{\text{tersisa}} = G \cdot v \cdot c \cdot u \cdot r. \tag{5.8}$$

Dengan nilai ilustratif $v = 0.5$, $c = 0.1$, $u = 0.2$, dan $r = 0.5$, yang masing-masing sendirian tampak seperti penyaring lemah (separuh akun bertahan, separuh hadiah tersisa), Persamaan (5.8) memberikan $0.5 \times 0.1 \times 0.2 \times 0.5 = 0.005$: skrip hanya menyimpan setengah persen dari yang semestinya diperolehnya. Sementara itu biayanya, yakni kotak surat untuk diverifikasi, teks beragam untuk dibangkitkan, dan waktu menunggu titipan, semuanya lebih tinggi daripada sebelumnya. Otomatisasi berhenti sepadan jauh sebelum satu lapisan pun sempurna. Aritmetika yang sama menjelaskan mengapa platform dapat menjaga setiap lapisan

tetap *lunak* terhadap anggota jujur: tidak ada satu lapisan pun yang harus keras, karena perlindungan datang dari hasil kali semuanya.

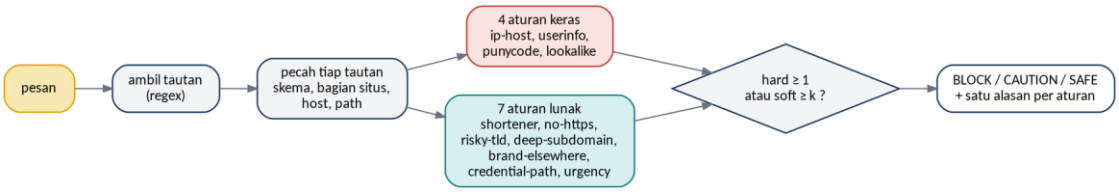
5.7.4 Yang sengaja tidak dipublikasikan

Lapisan-lapisannya publik; pengaturannya tidak. Batas pasti per tindakan, ambang kegunaan, tingkat kemiripan yang membuat dua kiriman dianggap duplikat, dan waktu titipan disetel dari pengamatan dan diubah ketika penyalahgunaan berubah. Memublikasikannya akan mengubah masing-masing menjadi sasaran, persis dalam pengertian Goodhart. Yang *dipublikasikan* adalah hasil yang penting bagi anggota: hadiah kecil, untuk manusia terverifikasi, untuk kontribusi yang berguna, dengan batas harian yang berganti pukul 00.00 WIB.

5.8 Lab G3: Penyaring Tautan Berbasis Aturan Milik Kita

5.8.1 Apa yang akan kita bangun

Lab ini membangun penyaring tautan untuk komunitas Profitina Laundry dalam Python dan Java, dengan perilaku dan keluaran yang identik, serta perangkat evaluasi yang mengukurnya pada himpunan sampel berlabel. Rancangannya, yang digambar pada **Gambar 5.7**, memiliki empat langkah: mengambil tautan dari pesan, memecah setiap tautan menjadi bagian-bagiannya, menerapkan setiap aturan, dan memutuskan.



Gambar 5.7 Penyaring lab. Tautan diambil dan dipecah, sebelas aturan buku teks diterapkan, dan putusannya BLOCK bila ada aturan keras yang terpicu atau sekurang-kurangnya k aturan lunak terpicu.

Aturannya adalah trik-trik buku teks pada **Tabel 5.1**, diubah menjadi kode dengan daftar-daftar yang memang dimaksudkan untuk Anda ubah: merek yang kita lindungi, domain resminya, pemendek URL, TLD yang kita anggap sering disalahgunakan, frasa desakan, dan kata-kata yang mengisyaratkan permintaan kredensial. **Tabel 5.2** mendaftar kesebelasnya.

Tabel 5.2 Kesebelas aturan penyaring lab. Daftar-daftar di baliknya adalah contoh milik buku ini dan dimaksudkan untuk disunting.

Aturan	Jenis	Terpicu bila
`ip-host`	keras	Host berupa alamat IPv4
`userinfo`	keras	Bagian situs memuat `@`
`punycode`	keras	Sebuah label host diawali `xn--`
`lookalike`	keras	Sebuah label host atau bagian yang dipisahkan tanda hubung berjarak tepat satu suntingan dari merek yang dilindungi, pada host yang tidak resmi
`shortener`	lunak	Host ada di daftar pemendek URL
`no-https`	lunak	Skemanya `http`
`risky-tld`	lunak	TLD ada di daftar yang sering disalahgunakan
`deep-subdomain`	lunak	Host memiliki empat label atau lebih
`brand-elsewhere`	lunak	Merek yang dilindungi muncul sebagai label atau bagian pada host yang tidak resmi
`credential-path`	lunak	Path memuat kata seperti `login`, `verify`, atau `otp`, pada host yang tidak resmi
`urgency`	lunak	Pesan memuat frasa desakan

5.8.2 Aturan keputusan

Misalkan h banyaknya aturan keras dan s banyaknya aturan lunak yang terpicu untuk sebuah tautan, dan k ambang yang Anda pilih. Putusan untuk tautan itu adalah

$$\text{putusan} = \begin{cases} \text{BLOCK} & \text{jika } h \geq 1 \text{ atau } s \geq k, \\ \text{CAUTION} & \text{jika } h = 0 \text{ dan } 1 \leq s < k, \\ \text{SAFE} & \text{jika } h = 0 \text{ dan } s = 0. \end{cases} \tag{5.9}$$

Pesan dengan beberapa tautan mendapat putusan terburuk di antara tautan-tautannya, dan pesan tanpa tautan sama sekali bernilai SAFE, karena penyaring tautan tidak punya apa pun untuk dikatakan tentangnya. Untuk evaluasi, BLOCK dihitung sebagai *ditandai*, sedangkan CAUTION dan SAFE dihitung sebagai *diloloskan*: tautan CAUTION tetap ditampilkan, mungkin dengan peringatan.

Ambang k adalah satu-satunya kenop penyetelan lab. Dengan $k = 1$ sinyal lunak apa pun memblokir; dengan $k = 5$ hampir tidak ada sinyal lunak yang pernah memblokir. Subbab 5.8.7 mengukur biaya setiap pilihan.

5.8.3 Seberapa jauh “beda satu huruf”?

Aturan `lookalike` memerlukan pengertian yang tepat tentang “beda satu huruf”. Pengertian bakunya adalah *jarak suntingan* (*edit distance*) yang diperkenalkan Levenshtein pada 1966 [14]: banyaknya penyisipan, penghapusan, dan penggantian satu karakter yang paling sedikit untuk mengubah satu string menjadi string lain. Tuliskan $D(i, j)$ untuk jarak antara i karakter pertama string a dan j karakter pertama string b. Maka $D(i, 0) = i$ dan $D(0, j) = j$ (membangun dari kosong atau menghapus sampai kosong), dan untuk $i, j \geq 1$

$$D(i,j) = \min(D(i-1,j) + 1,\ D(i,j-1) + 1,\ D(i-1,j-1) + [a_i \neq b_j]), \tag{5.10}$$

dengan $[a_i \neq b_j]$ bernilai 1 bila karakter ke-i dari a berbeda dari karakter ke-j dari b dan 0 bila tidak. Ketiga suku itu, berurutan, adalah menghapus a_i, menyisipkan b_j, dan mempertahankan atau mengganti. Mengisi tabel baris demi baris adalah algoritma pemrograman dinamis klasik Wagner dan Fischer [15]; waktunya $O(|a|\,|b|)$ dan, dengan hanya menyimpan dua baris, memorinya $O(|b|)$.

Tabel 5.3 mengisi tabel untuk $a = $ `pay` dan $b = $ `pya`, dua string yang berbeda karena dua huruf bertetangga bertukar tempat.

Tabel 5.3 Tabel jarak suntingan untuk `pay` dan `pya`, diisi baris demi baris dengan Persamaan (5.10); jawabannya ada di sel kanan bawah.

	“”	p	y	a
“”	0	1	2	3
p	1	0	1	2
a	2	1	1	1
y	3	2	1	2

Ambil satu sel untuk melihat rekurensinya bekerja: baris `a`, kolom `a`. Karakternya sama, jadi suku diagonalnya $D(1,2) + 0 = 1 + 0 = 1$, sedangkan menghapus memberikan $D(1,3) + 1 = 3$ dan menyisipkan memberikan $D(2,2) + 1 = 2$; minimumnya 1. Sel kanan bawah menyatakan jaraknya 2: pertukaran dua tetangga berbiaya dua suntingan menurut definisi Levenshtein. (Varian Damerau menghitung pertukaran sebagai satu suntingan; ini latihan pengembangan yang baik.) Untuk `profltina` dan `profitina` jaraknya 1, satu penggantian, dan untuk `profitina-laundry` dan `profitinalaundry` jaraknya juga 1, satu penyisipan tanda hubung.

5.8.4 Penyaring Python

Kode 5.1 adalah seluruh penyaringnya. Daftar-daftar di bagian atas adalah data aturan; ubah daftarnya dan perilakunya pun berubah.

Kode 5.1 `link_screener.py`: penyaring tautan berbasis aturan dengan sebelas aturan buku teks dan aturan keputusan Persamaan (5.9).

```
"""A small rule-based link screener, written for teaching.

The rules and lists below are this book's own textbook examples. Change them,
measure again with evaluate.py, and see what happens.

  python link_screener.py "message text with a link"  [--k 2]
"""
import re
import sys

BRANDS = ["profitina", "profitinalaundry", "dompetkita"]       # names we protect
OFFICIAL = ["profitina.com", "profitinalaundry.example", "dompetkita.example"]
SHORTENERS = ["bit.ly", "s.id", "tinyurl.com", "t.ly", "cutt.ly"]
RISKY_TLDS = ["xyz", "top", "click", "icu", "buzz", "cfd", "rest"]
URGENCY = ["urgent", "immediately", "24 hours", "account suspended",
           "account is blocked", "verify your account", "claim", "prize",
           "winner", "last chance", "limited time", "act now",
           "segera", "hadiah", "verifikasi"]
CREDENTIAL_WORDS = ["login", "verify", "secure", "wallet", "otp", "password"]

URL = re.compile(r"https?://[^\s<>\"']+"
                 r"|\b(?:www\.)?[a-z0-9-]+(?:\.[a-z0-9-]+)*\.[a-z]{2,}"
                 r"(?:/[^\s<>\"']*)?", re.IGNORECASE)
IPV4 = re.compile(r"^\d{1,3}(?:\.\d{1,3}){3}$")


def levenshtein(a, b):
    """Fewest one-character insertions, deletions or substitutions a -> b."""
    prev = list(range(len(b) + 1))
    for i in range(1, len(a) + 1):
        cur = [i] + [0] * len(b)
        for j in range(1, len(b) + 1):
            cost = 0 if a[i - 1] == b[j - 1] else 1
            cur[j] = min(prev[j] + 1, cur[j - 1] + 1, prev[j - 1] + cost)
        prev = cur
    return prev[len(b)]


def extract_links(text):
    return [m.group(0).rstrip(".,;:!?)'\"") for m in URL.finditer(text)]
```

Kode 5.1 (Lanjutan) *link_screener.py: penyaring tautan berbasis aturan dengan sebelas aturan buku teks dan aturan keputusan Persamaan (5.9).*

```
def split_link(link):
    """Return (scheme, authority, host, path) using plain string steps."""
    lower = link.lower()
    scheme = ""
    rest = link
    for s in ("https", "http"):
        if lower.startswith(s + "://"):
            scheme, rest = s, link[len(s) + 3:]
            break
    cut = len(rest)
    for ch in "/?#":
        pos = rest.find(ch)
        if pos != -1 and pos < cut:
            cut = pos
    authority, path = rest[:cut], rest[cut:]
    host = authority[authority.rfind("@") + 1:].split(":")[0].lower()
    return scheme, authority, host, path.lower()


def is_official(host):
    return any(host == d or host.endswith("." + d) for d in OFFICIAL)


def check_link(link, text):
    """Apply every rule to one link; return a list of (kind, name, why)."""
    scheme, authority, host, path = split_link(link)
    labels = host.split(".")
    tokens = set(labels[:-1])
    for label in labels[:-1]:
        tokens.update(label.split("-"))
    official = is_official(host)
    fired = []
    # Hard rules: one is enough to block.
    if IPV4.match(host):
        fired.append(("hard", "ip-host", "the host is a bare IP address"))
    if "@" in authority:
        fired.append(("hard", "userinfo",
                      "text before '@' hides the real host " + host))
    if any(label.startswith("xn--") for label in labels):
        fired.append(("hard", "punycode",
                      "the host uses punycode (look-alike letters)"))
    if not official:
        for tok in sorted(tokens):
            near = [b for b in BRANDS if levenshtein(tok, b) == 1]
            if near:
                fired.append(("hard", "lookalike",
```

Kode 5.1 (Lanjutan) *link_screener.py: penyaring tautan berbasis aturan dengan sebelas aturan buku teks dan aturan keputusan Persamaan (5.9).*

```
                                "'%s' is one letter away from '%s'"
                                % (tok, near[0])))
                    break
        # Soft rules: block when k or more of them fire.
        if host in SHORTENERS:
            fired.append(("soft", "shortener", "a short link hides the target"))
        if scheme == "http":
            fired.append(("soft", "no-https", "the link does not use HTTPS"))
        if labels[-1] in RISKY_TLDS:
            fired.append(("soft", "risky-tld",
                          "." + labels[-1] + " is on our list of often-abused endings"))
        if len(labels) >= 4:
            fired.append(("soft", "deep-subdomain",
                          "%d labels in the host name" % len(labels)))
        if not official and any(b in tokens for b in BRANDS):
            fired.append(("soft", "brand-elsewhere",
                          "a brand name on a domain that is not the brand's own"))
        if not official and any(w in path for w in CREDENTIAL_WORDS):
            fired.append(("soft", "credential-path",
                          "the path asks for a login, a code or a wallet"))
        lowered = text.lower()
        if any(w in lowered for w in URGENCY):
            fired.append(("soft", "urgency", "the message pushes you to act fast"))
        return host, fired


def decide(fired, k):
    """BLOCK if any hard rule, or at least k soft rules; else CAUTION/SAFE."""
    hard = sum(1 for kind, _, _ in fired if kind == "hard")
    soft = len(fired) - hard
    if hard >= 1 or soft >= k:
        return "BLOCK", hard, soft
    return ("CAUTION" if soft else "SAFE"), hard, soft


def screen(text, k=2):
    """Screen a whole message; the worst link decides. No link means SAFE."""
    order = {"SAFE": 0, "CAUTION": 1, "BLOCK": 2}
    worst, report = "SAFE", []
    for link in extract_links(text):
        host, fired = check_link(link, text)
        verdict, hard, soft = decide(fired, k)
        report.append((link, host, fired, verdict, hard, soft))
        if order[verdict] > order[worst]:
            worst = verdict
    return worst, report
```

Kode 5.1 (Lanjutan) *link_screener.py: penyaring tautan berbasis aturan dengan sebelas aturan buku teks dan aturan keputusan Persamaan (5.9).*

```
def main():
    args = sys.argv[1:]
    k = 2
    if "--k" in args:
        k = int(args[args.index("--k") + 1])
        del args[args.index("--k"):args.index("--k") + 2]
    text = " ".join(args)
    verdict, report = screen(text, k)
    if not report:
        print("no link found")
    for link, host, fired, v, hard, soft in report:
        print("link:", link)
        print("  host:", host)
        for kind, name, why in fired:
            print("  [%s] %s: %s" % (kind, name, why))
        print("  hard %d, soft %d (k = %d)  ->  %s" % (hard, soft, k, v))
    print("verdict:", verdict)


if __name__ == "__main__":
    main()
```

Beberapa hal patut diperhatikan.

- **Mengambil tautan.** Ekspresi reguler `URL` punya dua alternatif: tautan lengkap yang diawali `http://` atau `https://`, atau domain polos seperti `www.profitina-bonus.top` atau `s.id/laundry-kirim`, karena penipu sering menghilangkan skemanya. Tanda baca di ujung dibuang, agar tautan di akhir kalimat tidak ikut membawa titiknya.
- **Memecah tanpa pustaka.** `split_link` menemukan skema, memotong bagian situs pada `/`, `?`, atau `#` pertama, dan mengambil host sebagai bagian sesudah `@` *terakhir* dan sebelum `:` mana pun. Semua itu dikerjakan dengan operasi string biasa, bukan `urllib.parse`, agar versi Java dapat melakukan langkah yang persis sama dan menghasilkan jawaban yang persis sama.
- **Host resmi.** `is_official` menerima host yang sama dengan sebuah domain resmi atau berakhir dengan titik diikuti domain itu, sehingga `app.profitina.com` resmi sedangkan `profitina.com.account.secure-login.buzz` tidak. Tiga aturan tentang merek dan kredensial dilewati untuk host resmi.
- **Token.** Untuk aturan merek, host dipecah menjadi label-labelnya (tanpa TLD) dan bagian-bagian setiap label yang dipisahkan tanda hubung, sehingga `secure-dompetkita.co` menghasilkan token `dompetkita`. Pengurutan token membuat keluaran deterministik di kedua bahasa.
- **Penjelasan.** Setiap aturan yang terpicu mengembalikan alasan singkat. Penyaring tidak pernah hanya berkata “diblokir”.

Berapa banyak kerja yang dilakukan penyaring? Untuk pesan sepanjang m dengan L tautan, pengambilan tautan adalah satu kali lintasan atas teks, $O(m)$. Untuk setiap tautan, aturan-aturannya berupa pindaian sederhana atas tautan dan pesan, kecuali `lookalike`, yang menghitung jarak suntingan antara setiap token host dan setiap merek dari B merek yang dilindungi. Token dan merek sama-sama pendek, paling banyak beberapa puluh karakter, sehingga setiap jarak berbiaya paling banyak beberapa ribu langkah menurut batas $O(|a|\,|b|)$ di Subbab 5.8.3. Secara keseluruhan kerjanya $O(m + L \cdot t \cdot B \cdot \ell^2)$, dengan t banyaknya token per host dan ℓ panjang token maksimum: mikrodetik untuk pesan nyata mana pun. Itulah argumen biaya Subbab 5.4.2 dalam angka.

5.8.5 Penyaring Java

Kode 5.2 adalah penyaring yang sama dalam Java. *Record* menyimpan satu aturan yang terpicu, laporan untuk satu tautan, dan hasil untuk satu pesan; `TreeSet` menjaga token tetap terurut seperti `sorted` di Python; dan `break outer` berlabel berhenti setelah *lookalike* pertama, seperti `break` di Python.

Kode 5.2 `LinkScreener.java`: penyaring yang sama dalam Java, menghasilkan keluaran yang identik bita demi bita.

```
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.Set;
import java.util.TreeSet;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** A small rule-based link screener, written for teaching.
 *  The rules and lists are this book's own textbook examples.
 *    java -cp out LinkScreener "message text with a link" [--k 2] */
public class LinkScreener {
    static final List<String> BRANDS =
            List.of("profitina", "profitinalaundry", "dompetkita");
    static final List<String> OFFICIAL = List.of("profitina.com",
            "profitinalaundry.example", "dompetkita.example");
    static final List<String> SHORTENERS =
            List.of("bit.ly", "s.id", "tinyurl.com", "t.ly", "cutt.ly");
    static final List<String> RISKY_TLDS =
            List.of("xyz", "top", "click", "icu", "buzz", "cfd", "rest");
    static final List<String> URGENCY = List.of("urgent", "immediately",
            "24 hours", "account suspended", "account is blocked",
            "verify your account", "claim", "prize", "winner", "last chance",
            "limited time", "act now", "segera", "hadiah", "verifikasi");
    static final List<String> CREDENTIAL_WORDS =
            List.of("login", "verify", "secure", "wallet", "otp", "password");

    static final Pattern URL = Pattern.compile("https?://[^\\s<>\"']+"
            + "|\\b(?:www\\.)?[a-z0-9-]+(?:\\.[a-z0-9-]+)*\\.[a-z]{2,}"
            + "(?:/[^\\s<>\"']*)?", Pattern.CASE_INSENSITIVE);
    static final Pattern IPV4 = Pattern.compile("^\\d{1,3}(?:\\.\\d{1,3}){3}$");

    /** One rule that fired: hard or soft, its name and a reason. */
    record Hit(String kind, String name, String why) { }

    /** The result for one link. */
    record LinkReport(String link, String host, List<Hit> fired,
                      String verdict, int hard, int soft) { }

    /** The result for one message: the worst link decides. */
    record Result(String verdict, List<LinkReport> links) { }
```

Kode 5.2 (Lanjutan) *LinkScreener.java: penyaring yang sama dalam Java, menghasilkan keluaran yang identik bita demi bita.*

```
    /** Fewest one-character insertions, deletions or substitutions a -> b. */
    static int levenshtein(String a, String b) {
        int[] prev = new int[b.length() + 1];
        for (int j = 0; j <= b.length(); j++) prev[j] = j;
        for (int i = 1; i <= a.length(); i++) {
            int[] cur = new int[b.length() + 1];
            cur[0] = i;
            for (int j = 1; j <= b.length(); j++) {
                int cost = a.charAt(i - 1) == b.charAt(j - 1) ? 0 : 1;
                cur[j] = Math.min(Math.min(prev[j] + 1, cur[j - 1] + 1),
                                  prev[j - 1] + cost);
            }
            prev = cur;
        }
        return prev[b.length()];
    }

    static List<String> extractLinks(String text) {
        List<String> links = new ArrayList<>();
        Matcher m = URL.matcher(text);
        while (m.find()) {
            String link = m.group();
            int end = link.length();
            while (end > 0 && ".,;:!?)'\"".indexOf(link.charAt(end - 1)) >= 0) {
                end--;                                  // drop trailing punctuation
            }
            links.add(link.substring(0, end));
        }
        return links;
    }

    static boolean isOfficial(String host) {
        for (String d : OFFICIAL) {
            if (host.equals(d) || host.endsWith("." + d)) return true;
        }
        return false;
    }
```

Kode 5.2 (Lanjutan) *LinkScreener.java: penyaring yang sama dalam Java, menghasilkan keluaran yang identik bita demi bita.*

```
    /** Apply every rule to one link. */
    static List<Hit> checkLink(String link, String text, String[] hostOut) {
        String lower = link.toLowerCase(Locale.ROOT);
        String scheme = "", rest = link;
        for (String s : new String[] {"https", "http"}) {
            if (lower.startsWith(s + "://")) {
                scheme = s;
                rest = link.substring(s.length() + 3);
                break;
            }
        }
        int cut = rest.length();
        for (char ch : "/?#".toCharArray()) {
            int pos = rest.indexOf(ch);
            if (pos != -1 && pos < cut) cut = pos;
        }
        String authority = rest.substring(0, cut);
        String path = rest.substring(cut).toLowerCase(Locale.ROOT);
        String host = authority.substring(authority.lastIndexOf('@') + 1)
                .split(":")[0].toLowerCase(Locale.ROOT);
        hostOut[0] = host;
        String[] labels = host.split("\\.", -1);
        Set<String> tokens = new TreeSet<>();          // sorted, like Python
        for (int i = 0; i < labels.length - 1; i++) {
            tokens.add(labels[i]);
            for (String part : labels[i].split("-", -1)) tokens.add(part);
        }
        boolean official = isOfficial(host);
        String tld = labels[labels.length - 1];
        List<Hit> fired = new ArrayList<>();
        // Hard rules: one is enough to block.
        if (IPV4.matcher(host).matches())
            fired.add(new Hit("hard", "ip-host", "the host is a bare IP address"));
        if (authority.contains("@"))
            fired.add(new Hit("hard", "userinfo",
                    "text before '@' hides the real host " + host));
        for (String label : labels) {
            if (label.startsWith("xn--")) {
                fired.add(new Hit("hard", "punycode",
                        "the host uses punycode (look-alike letters)"));
                break;
            }
        }
        if (!official) {
            outer:
            for (String tok : tokens) {
```

Kode 5.2 (Lanjutan) *LinkScreener.java: penyaring yang sama dalam Java, menghasilkan keluaran yang identik bita demi bita.*

```
                for (String b : BRANDS) {
                    if (levenshtein(tok, b) == 1) {
                        fired.add(new Hit("hard", "lookalike", "'" + tok
                                + "' is one letter away from '" + b + "'"));
                        break outer;
                    }
                }
            }
        }
        // Soft rules: block when k or more of them fire.
        if (SHORTENERS.contains(host))
            fired.add(new Hit("soft", "shortener",
                    "a short link hides the target"));
        if (scheme.equals("http"))
            fired.add(new Hit("soft", "no-https", "the link does not use HTTPS"));
        if (RISKY_TLDS.contains(tld))
            fired.add(new Hit("soft", "risky-tld",
                    "." + tld + " is on our list of often-abused endings"));
        if (labels.length >= 4)
            fired.add(new Hit("soft", "deep-subdomain",
                    labels.length + " labels in the host name"));
        if (!official && BRANDS.stream().anyMatch(tokens::contains))
            fired.add(new Hit("soft", "brand-elsewhere",
                    "a brand name on a domain that is not the brand's own"));
        if (!official && CREDENTIAL_WORDS.stream().anyMatch(path::contains))
            fired.add(new Hit("soft", "credential-path",
                    "the path asks for a login, a code or a wallet"));
        String lowered = text.toLowerCase(Locale.ROOT);
        if (URGENCY.stream().anyMatch(lowered::contains))
            fired.add(new Hit("soft", "urgency",
                    "the message pushes you to act fast"));
        return fired;
    }

    /** Screen a whole message; the worst link decides. No link means SAFE. */
    static Result screen(String text, int k) {
        List<String> order = List.of("SAFE", "CAUTION", "BLOCK");
        String worst = "SAFE";
        List<LinkReport> report = new ArrayList<>();
        for (String link : extractLinks(text)) {
            String[] host = new String[1];
            List<Hit> fired = checkLink(link, text, host);
            int hard = (int) fired.stream()
                    .filter(h -> h.kind().equals("hard")).count();
            int soft = fired.size() - hard;
            String verdict = hard >= 1 || soft >= k ? "BLOCK"
```

Kode 5.2 (Lanjutan) *LinkScreener.java: penyaring yang sama dalam Java, menghasilkan keluaran yang identik bita demi bita.*

```
                            : soft > 0 ? "CAUTION" : "SAFE";
            report.add(new LinkReport(link, host[0], fired, verdict, hard, soft));
            if (order.indexOf(verdict) > order.indexOf(worst)) worst = verdict;
        }
        return new Result(worst, report);
    }

    public static void main(String[] args) {
        int k = 2;
        StringBuilder text = new StringBuilder();
        for (int i = 0; i < args.length; i++) {
            if (args[i].equals("--k") && i + 1 < args.length) {
                k = Integer.parseInt(args[++i]);
            } else {
                if (text.length() > 0) text.append(' ');
                text.append(args[i]);
            }
        }
        Result r = screen(text.toString(), k);
        if (r.links().isEmpty()) System.out.println("no link found");
        for (LinkReport lr : r.links()) {
            System.out.println("link: " + lr.link());
            System.out.println("  host: " + lr.host());
            for (Hit h : lr.fired()) {
                System.out.println("  [" + h.kind() + "] " + h.name() + ": "
                        + h.why());
            }
            System.out.printf("  hard %d, soft %d (k = %d)  ->  %s%n",
                    lr.hard(), lr.soft(), k, lr.verdict());
        }
        System.out.println("verdict: " + r.verdict());
    }
}
```

5.8.6 Mencobanya pada pesan tunggal

Jalankan penyaring pada satu pesan dari baris perintah. Contoh pertama adalah penipuan klasik:

```
python link_screener.py "URGENT: your Profitina account is blocked. Verify within 24
hours at http://profitina-support.xyz/login"
```

```
link: http://profitina-support.xyz/login
  host: profitina-support.xyz
  [soft] no-https: the link does not use HTTPS
  [soft] risky-tld: .xyz is on our list of often-abused endings
  [soft] brand-elsewhere: a brand name on a domain that is not the brand's own
  [soft] credential-path: the path asks for a login, a code or a wallet
  [soft] urgency: the message pushes you to act fast
  hard 0, soft 5 (k = 2)  ->  BLOCK
verdict: BLOCK
```

Tidak ada satu aturan pun di sini yang memberatkan secara mutlak, dan justru karena itu contoh ini mendidik: lima aturan lunak terpicu, jauh di atas $k = 2$. Tautan dari **Gambar 5.2** tertangkap oleh aturan keras:

```
python link_screener.py "Check your laundry receipt:
https://profitinalaundry.example@receipt-view.click/r/88"
```

```
link: https://profitinalaundry.example@receipt-view.click/r/88
  host: receipt-view.click
  [hard] userinfo: text before '@' hides the real host receipt-view.click
  [soft] risky-tld: .click is on our list of often-abused endings
  hard 1, soft 1 (k = 2)  ->  BLOCK
verdict: BLOCK
```

Kembaran beda satu huruf tertangkap oleh jarak suntingan Persamaan (5.10):

```
python link_screener.py "Free 50 PROFIT for early members! Connect your wallet:
https://profltina.com/airdrop"
```

```
link: https://profltina.com/airdrop
  host: profltina.com
  [hard] lookalike: 'profltina' is one letter away from 'profitina'
  hard 1, soft 0 (k = 2)  ->  BLOCK
verdict: BLOCK
```

Tautan resmi lolos tanpa ada yang dilaporkan, dan host yang sah tetapi tidak lazim mendapat CAUTION, bukan blokir:

```
python link_screener.py "Scan your site again after adding FAQ schema:
https://app.profitina.com/#/pindai"
python link_screener.py "Read the APJII survey: https://survey.apjii.or.id/"
```

```
link: https://app.profitina.com/#/pindai
  host: app.profitina.com
  hard 0, soft 0 (k = 2)  ->  SAFE
verdict: SAFE
link: https://survey.apjii.or.id/
  host: survey.apjii.or.id
  [soft] deep-subdomain: 4 labels in the host name
  hard 0, soft 1 (k = 2)  ->  CAUTION
verdict: CAUTION
```

Versi Java memberikan keluaran yang sama: di folder `java`, kompilasi dengan `javac -d out *.java` lalu jalankan `java -cp out LinkScreener "..."`.

5.8.7 Himpunan sampel dan evaluasinya

Penyaring hanya dapat dipercaya sejauh evaluasinya. Folder kode memuat `data/samples.tsv`, berisi 60 pesan yang ditulis untuk buku ini tentang komunitas Profitina Laundry, masing-masing berlabel `scam` atau `normal` dan dipisahkan dari labelnya oleh tab. Setiap nama dan tautan di dalamnya rekaan, dan himpunan ini sengaja memuat kasus sulit di kedua sisi. Pesan-pesannya ditulis dalam bahasa Inggris agar kode dan keluarannya sama persis di kedua edisi buku ini; daftar desakan dalam kode juga memuat beberapa kata kunci Indonesia seperti *segera*, *hadiah*, dan *verifikasi*, karena penipuan di Indonesia biasanya ditulis dalam bahasa Indonesia. **Tabel 5.4** menampilkan beberapa baris.

Tabel 5.4 Beberapa dari 60 sampel berlabel (30 penipuan, 30 normal) dalam `data/samples.tsv`.

#	Label	Pesan (dipersingkat)
1	scam	URGENT: your Profitina account is blocked. Verify within 24 hours at http://profitina-support.xyz/login
9	scam	Your parcel of clean clothes is waiting. Confirm the address: s.id/laundry-kirim
10	scam	Hi, this is the Profitina admin. Please send me the OTP code we just texted you.
26	scam	Confirm your email to keep earning PROFIT: https://profitina.com-verify.email/confirm
33	normal	Scan your site again after adding FAQ schema: https://app.profitina.com/#/pindai
42	normal	Last chance to join the OOP study group tonight: http://kelas.example/oop
50	normal	Local printer shop for our flyers: http://cetak-murah.xyz
51	normal	Read the APJII survey: https://survey.apjii.or.id/

Kode 5.3 membaca sampel, menyaring masing-masing, menghitung keempat hasil, mencetak ukuran-ukuran Subbab 5.5 beserta aritmetikanya, mendaftar setiap kesalahan beserta aturan yang terpicu, dan akhirnya mengulangi penghitungan untuk $k = 1$ sampai 5. **Kode 5.4** adalah versi Java-nya.

Kode 5.3 `evaluate.py`: matriks kebingungan, lima ukuran, daftar kesalahan, dan penyapuan atas k.

```
"""Measure the screener on our own labeled samples.

  python evaluate.py [--k 2] [--data ../data/samples.tsv]
"""
import sys
from pathlib import Path

from link_screener import screen

DATA = Path(__file__).resolve().parent.parent / "data" / "samples.tsv"


def load(path):
    samples = []
    for line in path.read_text(encoding="utf-8").splitlines():
        if line and not line.startswith("#"):
            label, text = line.split("\t", 1)
            samples.append((label, text))
    return samples


def ratio(num, den):
    return "n/a" if den == 0 else "%.3f" % (num / den)


def confusion(samples, k):
    """Count TP, FN, FP, TN; 'positive' means the screener says BLOCK."""
    tp = fn = fp = tn = 0
    mistakes = []
    for number, (label, text) in enumerate(samples, start=1):
        verdict, report = screen(text, k)
        blocked = verdict == "BLOCK"
        if label == "scam" and blocked:
            tp += 1
        elif label == "scam":
            fn += 1
            mistakes.append(("FN", number, verdict, report, text))
        elif blocked:
            fp += 1
            mistakes.append(("FP", number, verdict, report, text))
        else:
            tn += 1
    return tp, fn, fp, tn, mistakes
```

Kode 5.3 (Lanjutan) *evaluate.py: matriks kebingungan, lima ukuran, daftar kesalahan, dan penyapuan atas k.*

```
def main():
    args = sys.argv[1:]
    k = int(args[args.index("--k") + 1]) if "--k" in args else 2
    path = Path(args[args.index("--data") + 1]) if "--data" in args else DATA
    samples = load(path)
    scams = sum(1 for label, _ in samples if label == "scam")
    tp, fn, fp, tn, mistakes = confusion(samples, k)
    print("samples: %d (%d scam, %d normal), k = %d"
          % (len(samples), scams, len(samples) - scams, k))
    print()
    print("                  predicted BLOCK   predicted not BLOCK")
    print("actual scam       TP = %-12d  FN = %d" % (tp, fn))
    print("actual normal     FP = %-12d  TN = %d" % (fp, tn))
    print()
    print("precision = TP / (TP + FP) = %d / %d = %s"
          % (tp, tp + fp, ratio(tp, tp + fp)))
    print("recall    = TP / (TP + FN) = %d / %d = %s"
          % (tp, tp + fn, ratio(tp, tp + fn)))
    print("F1        = 2TP / (2TP + FP + FN) = %d / %d = %s"
          % (2 * tp, 2 * tp + fp + fn, ratio(2 * tp, 2 * tp + fp + fn)))
    print("accuracy  = (TP + TN) / all = %d / %d = %s"
          % (tp + tn, len(samples), ratio(tp + tn, len(samples))))
    print("FPR       = FP / (FP + TN) = %d / %d = %s"
          % (fp, fp + tn, ratio(fp, fp + tn)))
    print()
    print("mistakes:")
    for kind, number, verdict, report, text in mistakes:
        rules = sorted({name for _, _, fired, _, _, _ in report
                        for _, name, _ in fired})
        print("  %s #%02d %-7s rules: %s" % (kind, number, verdict,
                                             ", ".join(rules) or "(no link)"))
        print("         " + (text if len(text) <= 70 else text[:67] + "..."))
    print()
    print("k   TP  FN  FP  TN  precision  recall  F1")
    for kk in range(1, 6):
        a, b, c, d, _ = confusion(samples, kk)
        print("%d  %3d %3d %3d %3d  %9s  %6s  %s"
              % (kk, a, b, c, d, ratio(a, a + c), ratio(a, a + b),
                 ratio(2 * a, 2 * a + c + b)))


if __name__ == "__main__":
    main()
```

Kode 5.4 `Evaluate.java`: evaluasi yang sama dalam Java.

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.TreeSet;

/** Measure the screener on our own labeled samples.
 *    java -cp out Evaluate [--k 2] [--data ../data/samples.tsv] */
public class Evaluate {
    record Sample(String label, String text) { }

    record Mistake(String kind, int number, LinkScreener.Result result,
                   String text) { }

    record Counts(int tp, int fn, int fp, int tn, List<Mistake> mistakes) { }

    static List<Sample> load(Path path) throws IOException {
        List<Sample> samples = new ArrayList<>();
        for (String line : Files.readAllLines(path, StandardCharsets.UTF_8)) {
            if (!line.isEmpty() && !line.startsWith("#")) {
                int tab = line.indexOf('\t');
                samples.add(new Sample(line.substring(0, tab),
                                       line.substring(tab + 1)));
            }
        }
        return samples;
    }

    static String ratio(int num, int den) {
        if (den == 0) return "n/a";
        return String.format(Locale.ROOT, "%.3f", (double) num / den);
    }
```

Kode 5.4 (Lanjutan) *Evaluate.java: evaluasi yang sama dalam Java.*

```
    /** Count TP, FN, FP, TN; "positive" means the screener says BLOCK. */
    static Counts confusion(List<Sample> samples, int k) {
        int tp = 0, fn = 0, fp = 0, tn = 0;
        List<Mistake> mistakes = new ArrayList<>();
        for (int i = 0; i < samples.size(); i++) {
            Sample s = samples.get(i);
            LinkScreener.Result r = LinkScreener.screen(s.text(), k);
            boolean blocked = r.verdict().equals("BLOCK");
            if (s.label().equals("scam") && blocked) {
                tp++;
            } else if (s.label().equals("scam")) {
                fn++;
                mistakes.add(new Mistake("FN", i + 1, r, s.text()));
            } else if (blocked) {
                fp++;
                mistakes.add(new Mistake("FP", i + 1, r, s.text()));
            } else {
                tn++;
            }
        }
        return new Counts(tp, fn, fp, tn, mistakes);
    }

    public static void main(String[] args) throws IOException {
        int k = 2;
        Path path = Files.exists(Path.of("data/samples.tsv"))
                ? Path.of("data/samples.tsv") : Path.of("../data/samples.tsv");
        for (int i = 0; i + 1 < args.length; i += 2) {
            if (args[i].equals("--k")) k = Integer.parseInt(args[i + 1]);
            if (args[i].equals("--data")) path = Path.of(args[i + 1]);
        }
        List<Sample> samples = load(path);
        int scams = (int) samples.stream()
                .filter(s -> s.label().equals("scam")).count();
        Counts c = confusion(samples, k);
        int tp = c.tp(), fn = c.fn(), fp = c.fp(), tn = c.tn(), n = samples.size();
        System.out.printf("samples: %d (%d scam, %d normal), k = %d%n",
                n, scams, n - scams, k);
        System.out.println();
        System.out.println("                 predicted BLOCK"
                + "   predicted not BLOCK");
        System.out.printf("actual scam      TP = %-12d  FN = %d%n", tp, fn);
        System.out.printf("actual normal    FP = %-12d  TN = %d%n", fp, tn);
        System.out.println();
        System.out.printf("precision = TP / (TP + FP) = %d / %d = %s%n",
                tp, tp + fp, ratio(tp, tp + fp));
```

Kode 5.4 (Lanjutan) *Evaluate.java: evaluasi yang sama dalam Java.*

```
        System.out.printf("recall    = TP / (TP + FN) = %d / %d = %s%n",
                tp, tp + fn, ratio(tp, tp + fn));
        System.out.printf("F1        = 2TP / (2TP + FP + FN) = %d / %d = %s%n",
                2 * tp, 2 * tp + fp + fn, ratio(2 * tp, 2 * tp + fp + fn));
        System.out.printf("accuracy  = (TP + TN) / all = %d / %d = %s%n",
                tp + tn, n, ratio(tp + tn, n));
        System.out.printf("FPR       = FP / (FP + TN) = %d / %d = %s%n",
                fp, fp + tn, ratio(fp, fp + tn));
        System.out.println();
        System.out.println("mistakes:");
        for (Mistake m : c.mistakes()) {
            TreeSet<String> rules = new TreeSet<>();
            for (LinkScreener.LinkReport lr : m.result().links()) {
                for (LinkScreener.Hit h : lr.fired()) rules.add(h.name());
            }
            String names = rules.isEmpty() ? "(no link)" : String.join(", ", rules);
            System.out.printf("  %s #%02d %-7s rules: %s%n",
                    m.kind(), m.number(), m.result().verdict(), names);
            String t = m.text();
            if (t.length() > 70) t = t.substring(0, 67) + "...";
            System.out.println("         " + t);
        }
        System.out.println();
        System.out.println("k   TP  FN  FP  TN  precision  recall  F1");
        for (int kk = 1; kk <= 5; kk++) {
            Counts d = confusion(samples, kk);
            System.out.printf("%d  %3d %3d %3d %3d  %9s  %6s  %s%n", kk,
                    d.tp(), d.fn(), d.fp(), d.tn(), ratio(d.tp(), d.tp() + d.fp()),
                    ratio(d.tp(), d.tp() + d.fn()),
                    ratio(2 * d.tp(), 2 * d.tp() + d.fp() + d.fn()));
        }
    }
}
```

Jalankan (`python evaluate.py` di folder `python`, atau `java -cp out Evaluate` di folder `java`; keduanya mencetak hal yang sama):

```
samples: 60 (30 scam, 30 normal), k = 2

                 predicted BLOCK   predicted not BLOCK
actual scam      TP = 25            FN = 5
actual normal    FP = 3             TN = 27

precision = TP / (TP + FP) = 25 / 28 = 0.893
recall    = TP / (TP + FN) = 25 / 30 = 0.833
F1        = 2TP / (2TP + FP + FN) = 50 / 58 = 0.862
accuracy  = (TP + TN) / all = 52 / 60 = 0.867
FPR       = FP / (FP + TN) = 3 / 30 = 0.100

mistakes:
  FN #09 CAUTION rules: shortener
         Your parcel of clean clothes is waiting. Confirm the address: s.id/...
  FN #10 SAFE    rules: (no link)
         Hi, this is the Profitina admin. Please send me the OTP code we jus...
  FN #11 SAFE    rules: (no link)
         Transfer 20 PROFIT to u9a8b7c6d and I will send back 200 tomorrow, ...
  FN #23 SAFE    rules: (no link)
         I am staff. Send me your password and I will move your channel to O...
  FN #26 CAUTION rules: brand-elsewhere
         Confirm your email to keep earning PROFIT: https://profitina.com-ve...
  FP #42 BLOCK   rules: no-https, urgency
         Last chance to join the OOP study group tonight: http://kelas.examp...
  FP #43 BLOCK   rules: shortener, urgency
         Claim your seat for the GEO workshop at s.id/workshop-geo
  FP #50 BLOCK   rules: no-https, risky-tld
         Local printer shop for our flyers: http://cetak-murah.xyz

k   TP  FN  FP  TN  precision  recall  F1
1   27   3   6  24      0.818   0.900  0.857
2   25   5   3  27      0.893   0.833  0.862
3   14  16   0  30      1.000   0.467  0.636
4    9  21   0  30      1.000   0.300  0.462
5    8  22   0  30      1.000   0.267  0.421
```

5.8.8 Membaca hasilnya

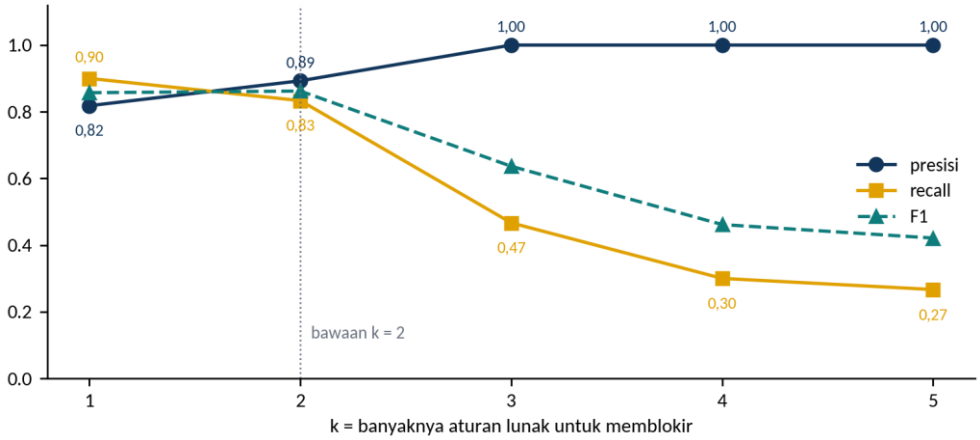
Matriks kebingungan. Dengan $k = 2$ penyaring menandai 25 dari 30 penipuan dan 3 dari 30 pesan normal: $TP = 25$, $FN = 5$, $FP = 3$, $TN = 27$, separuh kanan **Gambar 5.4**. Program mencetak sendiri aritmetika Persamaan (5.1) sampai Persamaan (5.5): presisi $25/28 = 0.893$, *recall* $25/30 = 0.833$, $F_1 = 50/58 = 0.862$, akurasi $52/60 = 0.867$, FPR $3/30 = 0.100$.

Negatif palsu. Tiga dari lima penipuan yang terlewat (#10, #11, dan #23) sama sekali tidak memuat tautan: permintaan OTP, janji "kirimi saya 20 PROFIT dan saya kembalikan 200", dan staf palsu yang meminta kata sandi. Tidak ada penyaring tautan yang dapat menangkapnya; itu tugas moderator dan sistem pelaporan (Subbab 5.6), dan persis itulah sebabnya komunitas memerlukan lebih dari satu pertahanan. Dua sisanya lebih menarik. Penipuan #9 hanya memakai pemendek URL, satu sinyal lunak, dan mendapat CAUTION. Penipuan #26 memakai host `profitina.com-verify.email`, yang diawali domain asli lalu dilanjutkan dengan `-verify.email`; pemecah token kita melihat label `profitina`, sehingga `brand-elsewhere` terpicu, tetapi tidak ada yang lain. Aturan yang memperhatikan domain resmi yang diikuti tanda hubung, atau `com` di dalam sebuah label, akan menangkapnya; itulah Soal 5.9.

Positif palsu. Ketiganya pesan jujur yang kebetulan memicu dua aturan lunak: undangan kelompok belajar dengan "last chance" dan tautan HTTP biasa (#42), undangan lokakarya dengan "claim your seat" dan pemendek URL (#43), serta percetakan lokal di TLD murah tanpa HTTPS (#50). Masing-masing mengingatkan bahwa sinyal lunak memang lunak. Perhatikan juga apa

yang *tidak* terjadi: #51, situs survei asosiasi internet nasional dengan empat label pada host-nya, hanya mendapat CAUTION.

Penyapuan atas k. Tabel terakhir, yang digambar pada **Gambar 5.8**, memperlihatkan tawar-menawar yang dihadapi setiap penyaring.



Gambar 5.8 Presisi, *recall*, dan F_1 penyaring lab ketika ambang lunak k bervariasi dari 1 sampai 5. k yang lebih rendah menangkap lebih banyak penipuan dan memblokir lebih banyak pesan jujur; mulai $k = 3$ tidak ada positif palsu, tetapi sebagian besar penipuan lolos.

Pada $k = 1$ *recall* naik menjadi 0,900 (27 dari 30: setiap penipuan yang bertautan tertangkap) tetapi laju positif palsu berlipat dua menjadi $6/30 = 0.200$. Pada $k = 3$ sama sekali tidak ada positif palsu, tetapi *recall* turun menjadi 0,467: lebih dari separuh penipuan lolos. F_1 terbaik, 0,862, ada pada $k = 2$, dan itulah sebabnya nilai itu menjadi bawaan. Namun penyapuan juga memperlihatkan mengapa F_1 bukan keseluruhan cerita: pada umpan sungguhan, tempat penipuan jarang, Persamaan (5.7) menyatakan bahwa FPR 0,100 akan membuat sebagian besar blokir keliru. Penyaring produksi memerlukan aturan yang jauh lebih baik, lebih banyak sampel normal, dan pendapat kedua, dan itulah persis yang disediakan rancangan berlapis Profitina.

Membandingkan dengan Profitina. Tolok ukur Profitina yang diterbitkan (27 dari 28 penipuan tertangkap, 0 dari 30 teks normal ditandai) lebih baik pada setiap ukuran daripada penyaring lab kita. Itu tidak perlu mengejutkan siapa pun: lantai aturan produksi yang disempurnakan pada lalu lintas sungguhan akan mengalahkan sebelas aturan buku teks yang ditulis untuk satu bab. Namun, seperti diingatkan di Subbab 5.5.5, kedua hasil berasal dari himpunan sampel yang berbeda, dan tidak satu pun himpunan itu besar. Tujuan menyandingkan keduanya adalah metodenya: empat hitungan yang sama, rumus yang sama, dan kejujuran yang sama tentang apa yang dapat dibuktikan 30 sampel normal.

5.8.9 Pandangan lain: titik-titik pada plot ROC

Penyapuan atas k juga dapat dibaca seperti biasanya literatur klasifikasi membacanya, yaitu sebagai titik-titik pada plot *receiver operating characteristic* (ROC) [10]. Setiap nilai k memberikan satu pasangan: laju positif palsu pada sumbu mendatar dan *recall*, yang dalam konteks ini disebut laju positif benar, pada sumbu tegak. Dari keluaran evaluasi, $k = 1$ memberikan titik (0.200,0.900), $k = 2$ memberikan (0.100,0.833), $k = 3$ memberikan (0.000,0.467), $k = 4$ memberikan (0.000,0.300), dan $k = 5$ memberikan (0.000,0.267). Penyaring sempurna akan berada di (0,1), pojok kiri atas; penyaring yang menandai secara acak berada pada diagonal dari (0,0) ke (1,1).

Dua hal terlihat pada kelima titik ini. Pertama, tidak ada nilai k yang mencapai pojok itu: menurunkan ambang hanya membeli *recall* dengan membayar positif palsu, yang merupakan tawar-menawar pada **Gambar 5.8** dalam bentuk lain. Kedua, titik-titik untuk $k = 3$, 4, dan 5 semuanya terletak pada sumbu tegak: begitu k sekurang-kurangnya 3, penyaring kita tidak pernah menandai pesan normal pada himpunan sampel ini, dan menaikkan k lebih jauh hanya membuat lebih banyak penipuan lolos. Tolok ukur Profitina yang diterbitkan, bila digambar sebagai titik pada sumbu yang sama, akan berada di $(0.000, 0.964)$, dekat dengan pojok itu. Pandangan ROC memperjelas satu pelajaran lagi: untuk menggeser penyaring ke arah pojok, menyetel ambang saja tidak cukup; aturan-aturannya sendiri harus menjadi lebih baik dalam memisahkan kedua kelas, dan itulah persis yang diminta soal-soal latihan di akhir bab ini.

5.8.10 Apa yang tidak dicakup lab

Penyaring kita tidak pernah mengunjungi tautan yang disaringnya, sehingga tidak dapat melihat ke mana pemendek URL mengarah, apa isi sebuah halaman, atau apakah sebuah domain baru didaftarkan kemarin. Penyaring ini tidak mengetahui daftar reputasi yang dipelihara pembuat peramban dan perusahaan keamanan. Daftar mereknya berisi tiga entri, dan himpunan sampelnya berisi 60 baris yang ditulis oleh satu tangan. Masing-masing adalah arah pertumbuhan sistem sungguhan. Gagasan berlapisnya, lantai yang cepat dan dapat dijelaskan dengan pendapat opsional di atasnya, semuanya diputuskan sekali di jalur tulis, tetap sama.

5.9 Profitina dalam Praktik

Kirim sebuah tautan. Di aplikasi yang berjalan, tautan dalam kiriman komunitas disaring ketika Anda mengirimnya. Tautan yang aman tampil sebagai tautan yang dapat diklik; tautan yang berbahaya diblokir [7].

Lihat kanal kelas. Kanal kelas memiliki moderator, sering kali dosennya, yang pengumuman sematannya muncul di bagian atas. Kiriman yang disembunyikan hilang dari pandangan; pengirimnya diberi tahu.

Laporkan sesuatu. Setiap kiriman dan setiap DM memiliki tindakan lapor. Laporan DM membekukan salinan satu pesan itu untuk ditinjau; tidak ada bagian lain percakapan yang menjadi terlihat.

Amati hadiahnya. Kiriman yang berguna memperoleh sedikit PROFIT setelah diselesaikan; duplikat memperoleh 0,00 dan menunjuk ke kiriman sebelumnya; tidak ada yang memperoleh apa pun sebelum surel Anda terverifikasi. Batasnya berganti pukul 00.00 WIB.

Yang berjalan dan yang masih rencana. Lantai aturan, pendapat AI lokal opsional di jalur cepat, moderator kanal, pelaporan dengan salinan beku, dan lapisan-lapisan main adil semuanya berjalan pada rilis September 2026 [7]. Pertanyaan tentang keamanan dapat dikirim ke `info@profitina.com`.

5.10 Kesalahpahaman yang Umum

"Model AI sebaiknya memutuskan apakah tautan aman." Model adalah pendapat kedua yang berguna, tetapi pemeriksaan keamanan harus menjawab bahkan ketika model sibuk, harus menjelaskan putusannya, dan harus dapat diuji. Itu tugas lantai aturan (Subbab 5.4.2).

"Pemeriksaan keamanan tempatnya di saat isi ditampilkan." Isi ditulis sekali dan dibaca berkali-kali; memeriksa di jalur baca melipatgandakan biaya sebanyak jumlah pembaca dan menaruhnya di depan setiap pembaca. Putuskan di jalur tulis dan simpan putusannya (Subbab 5.4.3).

"Tautan yang diawali nama merek adalah tautan milik merek itu." Hanya host yang menentukan. Pada `https://profitina.com.account.secure-login.buzz/verify` domain terdaftarnya adalah `secure-login.buzz` (Subbab 5.3.1).

"Akurasi menunjukkan seberapa baik penyaring." Pada umpan tempat 99 % pesan normal, penyaring yang meloloskan segalanya 99 % akurat. Laporkan *recall* dan laju positif palsu (Subbab 5.5.2).

"Nol positif palsu berarti penyaring tidak pernah memblokir orang jujur." Nol dari 30 membatasi lajunya di bawah sekitar 10 % dengan kepercayaan 95 %, menurut Persamaan (5.6); bukan membuatnya nol (Subbab 5.5.4).

"Presisi 0,89 pada tolok ukur berarti 89 % blokir sungguhan benar." Hanya jika lalu lintas sungguhan memiliki campuran yang sama dengan tolok ukur. Dengan penipuan yang jarang, Persamaan (5.7) dapat mengubah presisi nyata penyaring yang sama menjadi 0,15 (Subbab 5.5.5).

"Moderator adalah administrator kecil." Wewenangnya tiga, di dalam satu kanal, dicatat dan terlihat oleh orang yang terdampak; moderator tidak pernah menyentuh PROFIT atau data pribadi (Subbab 5.6.2).

"Memublikasikan ambang main adil akan lebih transparan." Itu akan menjadikan setiap ambang sasaran. Lapisan-lapisan dan tujuannya publik; pengaturannya disetel dan dijaga privat (Subbab 5.7.4).

5.11 Ringkasan Bab

- Komunitas menghadapi tiga ancaman: tautan berbahaya, penyalahgunaan oleh orang, dan penyalahgunaan oleh otomatisasi. Pada 2026 *phishing* mencapai 971.181 serangan dalam satu kuartal, Indonesia Anti-Scam Centre telah menerima 636.014 laporan, dan 53 % lalu lintas web bersifat otomatis (**Gambar 5.1**).
- Hanya host sebuah URL yang menentukan server siapa yang Anda tuju. Trik-trik klasik, yaitu host IP, informasi pengguna sebelum `@`, punycode, kembaran beda satu huruf, merek di domain lain, subdomain bertumpuk, pemendek URL, HTTP biasa, TLD murah, dan desakan, terbagi menjadi sinyal keras dan sinyal lunak (**Tabel 5.1**).
- Profitina menyaring tautan secara berlapis: lantai aturan deterministik yang menggabungkan beberapa sinyal, dengan pendapat AI lokal opsional di jalur GPU cepat di atasnya. Keduanya berjalan sekali di jalur tulis; jalur baca hanya membaca putusan yang tersimpan (**Gambar 5.3**).
- Penyaring diukur dengan matriks kebingungan serta presisi, *recall*, F_1, akurasi, dan laju positif palsu (Persamaan (5.1)–Persamaan (5.5)). Tolok ukur publik Profitina, 27 dari 28 penipuan tertangkap dengan 0 dari 30 positif palsu, memberikan presisi 1,000 dan *recall* 0,964; menurut aturan tiga (Persamaan (5.6)) laju positif palsunya di bawah sekitar 10 % dengan kepercayaan 95 %, dan ketika penipuan jarang, presisi yang dialami anggota sangat bergantung padanya (Persamaan (5.7)).

- Moderator kanal menyembunyikan, menyematkan, dan membisukan hanya di dalam satu kanal, dengan setiap tindakan dicatat dan pengirim diberi tahu, dan tidak pernah menyentuh PROFIT atau data pribadi. Laporan membekukan salinan pesan yang dilaporkan.
- Main adil menjaga hadiah untuk manusia melalui lapisan-lapisan yang saling mengalikan (Persamaan (5.8)): OTP surel, perolehan hanya setelah terverifikasi, hadiah kecil dengan batas harian, nol untuk duplikat, skor kegunaan AI lokal, titipan dengan tarik-kembali dan pembakaran, penghitungan ulang saat disunting, dan peringkat yang tidak dapat dibeli.
- Lab G3 membangun penyaring sebelas aturan dalam Python dan Java dengan aturan keputusan Persamaan (5.9) dan jarak suntingan Persamaan (5.10), lalu mengukurnya pada 60 sampel berlabel: pada $k = 2$, presisi 0,893, *recall* 0,833, F_1 0,862, FPR 0,100; penyapuan atas k memperlihatkan tawar-menawarnya, dan setiap kesalahan dijelaskan.

5.12 Soal Latihan

Soal 5.1. Untuk setiap tautan, sebutkan host dan domain terdaftarnya, dan sebutkan aturan mana pada **Tabel 5.2** yang terpicu: (a) `https://login.dompetkita.example/secure`; (b) `http://dompetkita.example.secure-pay.top/login`; (c) `https://www.profitina.com/blog`.

Soal 5.2. Hitunglah dengan tangan, memakai metode tabel pada **Tabel 5.3**, jarak suntingan antara `dana` dan `danna`, serta antara `kilat` dan `klilat`. Mana yang akan dianggap beda satu huruf oleh aturan `lookalike`?

Soal 5.3. Sebuah penyaring diuji pada 40 penipuan dan 60 pesan normal dan menandai 34 penipuan dan 3 pesan normal. Tuliskan matriks kebingungannya dan hitung presisi, *recall*, F_1, akurasi, dan FPR dengan Persamaan (5.1) sampai Persamaan (5.5).

Soal 5.4. Sebuah tolok ukur memuat 200 pesan normal dan tidak satu pun ditandai. Dengan Persamaan (5.6), berikan batas atas 95 % eksak dan hampiran untuk laju positif palsunya. Berapa banyak pesan normal tanpa positif palsu yang diperlukan untuk menekan batas itu ke bawah 0,5 %?

Soal 5.5. Untuk penyaring pada Soal 5.3, hitung nilai prediktif positif dengan Persamaan (5.7) ketika 1 % pesan bertautan adalah penipuan. Berapa laju positif palsu yang diperlukan, dengan *recall* yang sama, agar sekurang-kurangnya separuh dari semua blokir benar?

Soal 5.6. Jelaskan dengan kata-kata Anda sendiri mengapa lapisan AI opsional Profitina berjalan di jalur tulis, dan apa yang harus dibayar umpan bila lapisan itu berjalan di jalur baca. Gunakan angka-angka Subbab 4.11.

Soal 5.7. Jalankan `python evaluate.py --k 1` dan daftarkan keenam positif palsunya. Untuk masing-masing, sebutkan aturan lunak mana yang terpicu dan apakah menurut Anda pesan itu sebaiknya ditampilkan dengan peringatan atau ditampilkan biasa.

Soal 5.8. Tambahkan aturan lunak kedua belas, `at-in-path`, yang terpicu bila *path* memuat `@`. Ramalkan pengaruhnya pada 60 sampel sebelum menjalankan evaluasi, lalu periksa. Mengapa aturan ini lunak, padahal `userinfo` keras?

Soal 5.9. Penipuan #26 (`profitina.com-verify.email`) terlewat pada $k = 2$. Rancanglah satu aturan baru yang menangkapnya tanpa menimbulkan positif palsu pada 30 sampel normal. Nyatakan aturan itu dengan tepat, sebutkan apakah keras atau lunak, dan sebutkan satu tautan sah yang mungkin masih diblokirnya.

Soal 5.10. Tulislah lima pesan normal baru tentang Profitina Laundry yang akan keliru diblokir penyaring saat ini pada $k = 2$, dan lima penipuan baru yang akan terlewat. Apa yang diajarkan latihan ini tentang himpunan sampel yang ditulis oleh orang yang sama dengan pembuat aturannya?

Soal 5.11. Seorang dosen yang memoderasi kanal kelas meminta wewenang untuk "melihat akun siapa di balik sebuah *handle*" ketika sebuah kiriman melanggar aturan. Dengan Subbab 5.6.2, tuliskan jawaban singkat yang menjelaskan mengapa moderator tidak memiliki wewenang itu dan apa yang dapat mereka lakukan sebagai gantinya.

Soal 5.12. Sebuah skrip membuat 100 akun dan mengirim dari masing-masing. Telusuri lapisan-lapisan pada **Gambar 5.6** dan sebutkan di mana setiap lapisan memperlambat skrip itu. Lapisan mana yang menghentikannya paling awal, dan mana yang menangkapnya paling akhir?

5.13 Penyelesaian Soal Latihan

Penyelesaian 5.1. (a) Host `login.dompetkita.example`, domain terdaftar `dompetkita.example`, yang resmi; host berakhir dengan `.dompetkita.example`, sehingga aturan merek dan kredensial dilewati, dan dengan HTTPS, TLD biasa, serta tiga label, tidak ada aturan yang terpicu: SAFE. (b) Host `dompetkita.example.secure-pay.top`, domain terdaftar `secure-pay.top`. Aturan: `no-https`, `risky-tld` (`.top`), `deep-subdomain` (empat label), `brand-elsewhere` (`dompetkita` pada host tidak resmi), dan `credential-path` (`login`): lima aturan lunak, BLOCK. (c) Host `www.profitina.com`, yang berakhir dengan `.profitina.com` dan resmi: SAFE.

Penyelesaian 5.2. `dana` ke `danna`: satu penyisipan, jarak 1. `kilat` ke `klilat`: sisipkan `l` sesudah `k` menghasilkan `klilat`, jarak 1. Keduanya beda satu huruf; aturan `lookalike` hanya terpicu bila salah satunya token host dan yang lain ada di daftar merek yang dilindungi.

Penyelesaian 5.3. $TP = 34$, $FN = 6$, $FP = 3$, $TN = 57$. Presisi $34/37 \approx 0.919$; *recall* $34/40 = 0.850$; $F_1 = 68/(68 + 3 + 6) = 68/77 \approx 0.883$; akurasi $91/100 = 0.910$; FPR $3/60 = 0.050$.

Penyelesaian 5.4. Eksak: $1 - 0.05^{1/200} = 1 - e^{-2.996/200} = 1 - e^{-0.01498} \approx 0.0149$. Hampiran: $3/200 = 0.015$. Agar di bawah 0,005 kita memerlukan $3/n < 0.005$, jadi $n > 600$; dengan rumus eksak, $n \geq 598$. Dalam praktik, sekitar 600 pesan normal tanpa positif palsu.

Penyelesaian 5.5. Dengan *recall* 0,850, FPR 0,050, dan $\pi = 0.01$: $\text{PPV} = 0.0085/(0.0085 + 0.0495) = 0.0085/0.058 \approx 0.147$. Agar PPV sekurang-kurangnya 0,5 kita memerlukan $\text{recall} \cdot \pi \geq \text{FPR} \cdot (1 - \pi)$, jadi $\text{FPR} \leq 0.0085/0.99 \approx 0.0086$: di bawah sekitar 0,9 %, kira-kira enam kali lebih rendah daripada sekarang.

Penyelesaian 5.6. Di jalur tulis, model ditanya sekali per kiriman, selagi hanya pengirim yang menunggu, dan putusannya disimpan. Di jalur baca, model akan ditanya setiap kali anggota mana pun membuka umpan yang memuat kiriman itu, selagi anggota itu menunggu. Subbab 4.11 mengukur satu halaman umpan berisi 30 kiriman pada 868 ms dengan enam panggilan model di jalur baca (ditambah masalah-masalah lain yang dijelaskan di sana), dibandingkan 11 ms setelah jalur baca berhenti memanggil model dan kuerinya diperbaiki. Seribu pembaca akan melipatgandakan panggilan model seribu kali; di jalur tulis, banyaknya panggilan sama dengan banyaknya kiriman.

Penyelesaian 5.7. Pada $k = 1$ keluaran mendaftar enam positif palsu: #39 (*shortener*), #40 (*no-https*), #42 (*no-https*, *urgency*), #43 (*shortener*, *urgency*), #50 (*no-https*, *risky-tld*), dan #51 (*deep-subdomain*). Pertimbangan yang wajar: #39, #40, dan #51 sebaiknya ditampilkan biasa atau

dengan peringatan ringan, karena satu sinyal lunak lazim ada di kiriman jujur; #42, #43, dan #50 jujur tetapi membawa dua sinyal, sehingga peringatan dapat dibenarkan. Persis inilah sebabnya bawaannya $k = 2$ dan mengapa CAUTION ada.

Penyelesaian 5.8. Di Python, sesudah pemeriksaan `credential-path` tambahkan `if "@" in path: fired.append(("soft", "at-in-path", "the path contains '@'"))`, dan padanannya di Java. Tidak satu pun dari 60 sampel memiliki `@` di path, sehingga evaluasi tidak berubah: pengingat yang berguna bahwa sebuah aturan hanya dapat dinilai pada sampel yang menguji aturan itu. Aturan ini lunak karena `@` muncul secara sah di path, misalnya pada halaman profil seperti `/@username` di beberapa platform, sedangkan `@` di *bagian situs* tidak punya kegunaan jujur dalam tautan yang dibagikan.

Penyelesaian 5.9. Salah satu kemungkinan: aturan lunak `official-prefix` yang terpicu bila host *diawali* domain resmi yang diikuti karakter selain `.` atau akhir host, misalnya `profitina.com-`; atau, lebih umum, bila sebuah label selain yang terakhir tepat `com` atau diawali `com-`. Pada 30 sampel normal aturan ini tidak pernah terpicu, dan pada #26 aturan ini menambah aturan lunak kedua, sehingga #26 kemudian memiliki dua aturan lunak (`brand-elsewhere`, `official-prefix`) dan diblokir pada $k = 2$, tanpa positif palsu baru. Tautan sah yang mungkin masih diblokirnya: mitra yang domainnya kebetulan diawali nama domain merek, seperti `profitina.community-events.example` yang hipotetis; itulah sebabnya aturan ini lunak.

Penyelesaian 5.10. Jawabannya beragam. Pesan normal yang lazim keliru diblokir: toko lokal di TLD murah tanpa HTTPS; tautan pendek ke formulir pendaftaran dengan “last chance”; pesan yang mengutip penipuan sebagai peringatan (“Hati-hati dengan http://profitina-support.xyz/login!”). Penipuan yang lazim terlewat: penipuan di domain `.com` biasa dengan HTTPS dan tanpa merek; penipuan yang meminta pembaca mengetik kode dalam balasan; penipuan yang dipecah menjadi dua pesan. Pelajarannya: siapa pun yang menulis aturan juga tahu, sadar atau tidak, sampel mana yang akan ditangkap aturannya. Sampel independen, idealnya laporan sungguhan, diperlukan untuk evaluasi yang jujur.

Penyelesaian 5.11. Salah satu jawaban yang mungkin: “Moderator menjaga apa yang dikatakan di kanal, bukan siapa yang mengatakannya. Mengaitkan *handle* dengan rincian pribadi sebuah akun akan memberi setiap moderator akses ke data pribadi yang wajib kami lindungi menurut UU PDP, dan akan membuat kanal seorang dosen kurang aman bagi mahasiswa untuk berbicara. Yang dapat Anda lakukan adalah menyembunyikan kiriman, membisukan anggota untuk sementara, dan menyematkan pengingat tentang aturan; anggota itu diberi tahu, tindakannya dicatat, dan bila masalahnya serius Anda dapat melaporkannya untuk ditinjau platform.”

Penyelesaian 5.12. OTP surel: masing-masing dari 100 akun memerlukan kotak surat yang berfungsi dan sebuah kode, yang memperlambat pembuatan massal dan mencegah akun tak terverifikasi memperoleh apa pun. Batas harian: setiap akun memperoleh paling banyak jumlah yang dibatasi per tindakan per hari, sebanyak apa pun ia mengirim. Duplikat: kiriman yang identik atau nyaris identik lintas akun memperoleh 0,00 dan menunjuk ke kiriman sebelumnya. Skor kegunaan: teks hasil bangkitan yang remeh memperoleh 0,00. Titipan dan tarik-kembali: hadiah yang belakangan diketahui sebagai penyalahgunaan diambil kembali dan dibakar. Penghenti paling awal adalah verifikasi surel, sebelum ada hadiah apa pun; yang paling akhir adalah tarik-kembali, setelah hadiah sempat dikreditkan sementara.

Daftar Pustaka

[1] Anti-Phishing Working Group, *Phishing Activity Trends Report, 1st Quarter 2026*, May 27, 2026. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2026.pdf

[2] ANTARA, "OJK: IASC memulihkan dana korban penipuan Rp204,3 miliar hingga Juli," Aug. 4, 2026. [Online]. Available: https://www.antaranews.com/berita/5679972/ojk-iasc-memulihkan-dana-korban-penipuan-rp2043-miliar-hingga-juli

[3] Imperva (Thales), "Bad Bot Report 2026: Bots in the Agentic Age," Apr. 29, 2026. [Online]. Available: https://www.imperva.com/blog/bad-bot-report-2026-bots-agentic-age/

[4] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," IETF RFC 3986, Jan. 2005.

[5] A. Costello, "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)," IETF RFC 3492, Mar. 2003.

[6] Unicode Consortium, "Unicode Security Mechanisms," Unicode Technical Standard #39, version 18.0.0, Aug. 27, 2026. [Online]. Available: https://www.unicode.org/reports/tr39/

[7] Profitina, *Profitina Whitepaper — Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[8] Profitina, "Current facts for course material," Profitina Course Library, Sep. 23, 2026.

[9] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge University Press, 2008, ch. 8.

[10] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[11] J. A. Hanley and A. Lippman-Hand, "If nothing goes wrong, is everything all right? Interpreting zero numerators," *JAMA*, vol. 249, no. 13, pp. 1743–1745, 1983.

[12] The Santa Clara Principles on Transparency and Accountability in Content Moderation, 2nd ed., Dec. 2021. [Online]. Available: https://santaclaraprinciples.org/

[13] M. Strathern, "'Improving ratings': audit in the British University system," *European Review*, vol. 5, no. 3, pp. 305–321, 1997.

[14] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[15] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, 1974.

Bab 6

AI Lokal: Mesin Konten dan Model Dua Jalur

Setelah menyelesaikan bab ini, Anda akan mampu menjelaskan dengan kata-kata sederhana apa yang dikerjakan sebuah *large language model* (model bahasa besar) ketika ia "menulis"; menerangkan mengapa Profitina menjalankan seluruh kecerdasan buatannya di mesinnya sendiri dan tidak pernah mengirim data pengguna ke layanan AI yang di-*hosting* pihak lain, beserta harga yang harus dibayar untuk pilihan itu; menggambarkan Ollama dan keluarga model Qwen sebagaimana keadaannya pada September 2026; menggambar model dua jalur, yaitu jalur kualitas untuk draf panjang dan jalur cepat untuk label pendek, lalu menentukan pekerjaan mana yang masuk ke jalur mana dan mengapa sebagian pekerjaan tidak boleh masuk ke jalur mana pun; menjelaskan *quality gate* (gerbang kualitas), yaitu aturan bahwa draf AI hanya disajikan bila mengalahkan templat bawaan, serta membaca hasil publik berupa 5 kemenangan dari 6 kasus uji tanpa menafsirkannya berlebihan; dan merancang perangkat lunak yang berkata "tidak ada model yang tersedia" alih-alih berpura-pura. Di bagian lab Anda akan menulis klien untuk antarmuka HTTP Ollama dalam Python dan Java, sebuah server *stub* kecil yang meniru Ollama sehingga semuanya tetap berjalan bahkan di komputer yang sama sekali tidak memiliki model, sebuah pengarah dua jalur, dan gerbang kualitas buatan Anda sendiri.

6.1 Titik Berangkat Kita

Bab 5 ditutup dengan satu kalimat kecil tetapi penting tentang Link Safety: setiap tautan yang diunggah ke komunitas lebih dahulu disaring oleh *rule floor* (lantai aturan), yaitu serangkaian pemeriksaan sederhana dan deterministik, dan pendapat opsional dari AI lokal boleh duduk di atas lantai itu, tidak pernah di bawahnya. Lantai aturan selalu menjawab, dalam hitungan mikrodetik, dan memberi jawaban yang sama untuk masukan yang sama. Pendapat AI lebih lambat, sesekali tidak tersedia, dan kadang keliru, sehingga produk dibangun agar tetap bekerja tanpanya.

Kalimat itu memuat seluruh sikap bab ini dalam bentuk mini. Profitina menggunakan kecerdasan buatan dengan sungguh-sungguh: Content Engine-nya menulis draf yang langsung menjawab (*answer-first*), pemindainya memakai model untuk menyarankan kategori bisnis, sistem Fair-Play-nya memakai model untuk menilai apakah sebuah unggahan bermanfaat, dan Link Safety dapat meminta pendapat kedua dari model. Meski begitu, produk ini tidak pernah memperlakukan model sebagai kotak ajaib yang selalu ada dan selalu benar. Model adalah komponen yang memiliki kecepatan, tingkat kegagalan, dan tingkat mutu, yang semuanya dapat diukur, dan perangkat lunak di sekelilingnya dirancang berdasarkan hasil pengukuran itu.

Kita mulai dari dasar. Subbab 6.2 menjelaskan apa itu model bahasa bagi pembaca yang belum pernah melihat isinya. Subbab 6.3 menerangkan mengapa Profitina bersikeras menjalankan modelnya secara lokal. Subbab 6.4 memperkenalkan Ollama, program yang melayani model, dan keluarga model Qwen yang dilayaninya. Subbab 6.5 sampai 6.7 menguraikan tiga gagasan yang membuat AI lokal dapat diandalkan di dalam produk: dua jalur, gerbang kualitas, dan kejujuran

ketika tidak ada model yang tersedia. Subbab 6.8 adalah lab, dan Subbab 6.9 kembali ke produk yang sesungguhnya.

6.2 Apa yang Dikerjakan Model Bahasa, dengan Kata-Kata Sederhana

6.2.1 Menebak potongan teks berikutnya

Bayangkan seorang pelayan toko di Profitina Laundry yang sangat banyak membaca: jutaan halaman berisi nota, resep, berita, halaman produk, dan percakapan. Anda berkata kepadanya, "Profitina Laundry adalah layanan binatu di", lalu berhenti. Hampir tanpa berpikir ia melanjutkan: "Surabaya." Ia tidak mencari apa pun. Dari semua bacaannya ia hanya telah mempelajari kata-kata mana yang biasanya menyusul kata-kata lain, dan dari bagian awal kalimat Anda ia menangkap bahwa yang dibicarakan adalah sebuah tempat.

Large language model (LLM) adalah program komputer yang melakukan persis hal itu, tidak lebih misterius: diberi sepenggal teks, ia menebak kelanjutan yang mungkin, sepotong kecil demi sepotong kecil. Potongan kecil itu disebut *token*. Sebuah token sering berupa satu kata utuh yang umum ("laundry"), kadang bagian dari kata yang lebih jarang ("Profit" + "ina"), kadang satu tanda baca. Setelah model menebak satu token, token itu ditambahkan ke teks, dan model menebak token berikutnya dari teks yang kini lebih panjang. Mengulang langkah ini beberapa ratus kali menghasilkan satu paragraf.

> **DEFINISI.** *Model bahasa* memberikan peluang kepada setiap *token* yang mungkin menjadi kelanjutan sebuah teks. *Generasi* adalah perulangan yang berkali-kali memilih satu token berikutnya, menambahkannya, lalu bertanya lagi. Model bahasa *besar* adalah model dengan miliaran bilangan yang dapat diatur, disebut *parameter*, yang dipelajari dari kumpulan teks yang sangat besar.

Rancangan modern di balik hampir semua model semacam ini adalah *transformer*, yang diperkenalkan pada 2017 [1]. Anda tidak memerlukan matematikanya untuk mengikuti buku ini; yang Anda perlukan adalah tiga akibatnya.

Pertama, pengetahuan model membeku di dalam parameternya pada saat pelatihan selesai. Model tidak mengetahui jam buka binatu Anda kecuali Anda memberitahukannya di dalam permintaan, dan ia mungkin dengan yakin menyebut jam yang salah jika Anda tidak memberitahukannya. Karena itulah setiap penggunaan model yang serius di Profitina menyerahkan fakta yang diperlukan di dalam permintaan, alih-alih memercayai ingatan model.

Kedua, generasi adalah sebuah perulangan, sehingga waktunya bertambah seiring panjang jawaban. Label satu kata itu cepat; artikel empat ratus kata itu lambat. Subbab 6.5 membangun seluruh arsitektur di atas fakta tunggal ini.

Ketiga, generasi melibatkan unsur peluang. Pada setiap langkah model memiliki *distribusi* token berikutnya yang mungkin, dan perangkat lunak memilih salah satunya; sebuah pengaturan bernama *temperature* mengendalikan seberapa berani pilihan itu, dan sebuah *seed* membuat pilihan dapat diulang. Dua kali menjalankan permintaan yang sama dapat menghasilkan teks yang berbeda. Karena itu sebuah produk tidak dapat menguji model sekali lalu menganggap jawabannya berlaku selamanya; produk harus memeriksa setiap jawaban, dan itulah yang dilakukan gerbang kualitas di Subbab 6.6. Para praktisi memahami hal ini dengan baik: dalam Stack Overflow Developer Survey 2025, 84% responden menggunakan atau berencana

menggunakan perkakas AI, tetapi lebih banyak yang tidak memercayai ketepatan perkakas itu (46%) daripada yang memercayainya (33%) [2].

6.2.2 Parameter, memori, dan kuantisasi

Ukuran model dinyatakan sebagai jumlah parameter: "7B" berarti kira-kira tujuh miliar. Setiap parameter adalah sebuah bilangan, dan setiap bilangan memakan memori. Jika sebuah model memiliki N parameter dan masing-masing disimpan dengan b bit, bobotnya memerlukan kira-kira

$$M \approx \frac{N \cdot b}{8} \text{ bita} \tag{6.1}$$

memori, karena delapan bit sama dengan satu bita. Pelatihan biasanya memakai 16 bit per parameter. Menjalankan model di rumah atau di satu server biasanya memakai *kuantisasi* (*quantization*): setiap parameter dibulatkan menjadi bilangan yang lebih kasar dengan hanya 4 atau 5 bit, sehingga ukuran model menyusut menjadi sekitar seperempat atau sepertiganya dengan penurunan mutu yang ternyata kecil [3]. Ollama mendistribusikan modelnya dalam format berkas terkuantisasi bernama GGUF, yang berasal dari proyek llama.cpp [4].

> **CONTOH.** Qwen2.5 7B memiliki sekitar $N = 7.6 \times 10^9$ parameter. Pada 16 bit, Persamaan (6.1) memberikan $7.6 \times 10^9 \times 16/8 = 15.2 \times 10^9$ bita, sekitar 15 GB, yang tidak muat di sebagian besar kartu grafis laptop. Dengan kuantisasi 4 bit ditambah sedikit tambahan, misalnya rata-rata $b \approx 4.9$ bit, hasilnya $7.6 \times 10^9 \times 4.9/8 \approx 4.7 \times 10^9$ bita. Angka itu cocok dengan ukuran unduhan 4,7 GB yang dicantumkan pustaka Ollama untuk `qwen2.5:7b` [5].

Aritmetika yang sama menjelaskan mengapa model yang sangat kecil seperti Qwen3 1.7B muat dengan leluasa di kartu grafis sederhana, dan mengapa model 27 miliar parameter memerlukan *workstation*. Ukuran bukanlah mutu, tetapi ukuran menentukan di mana model dapat berjalan, dan tempat model berjalan menentukan seberapa cepat ia menjawab.

Dengan kata lain: model adalah tabel besar berisi bilangan hasil belajar ditambah sebuah perulangan yang menebak token berikutnya. Ukuran tabel menentukan perangkat kerasnya; panjang perulangan menentukan lamanya menunggu; dan unsur peluang pada setiap langkah menentukan bahwa setiap jawaban harus diperiksa.

6.3 Mengapa Hanya Inferensi Lokal: Privasi, Biaya, Kendali

Inferensi berarti menjalankan model yang sudah terlatih untuk memperoleh jawaban, berbeda dengan *melatih* model. Ada dua cara melakukan inferensi di dalam produk. Anda dapat mengirim teks melalui internet ke perusahaan yang menjalankan model untuk Anda dan memungut biaya per permintaan (model *hosted* atau model *cloud*), atau Anda menjalankan model sendiri di mesin yang Anda kendalikan (model *lokal* atau *self-hosted*). Profitina memilih cara kedua untuk setiap fiturnya, dan menghapus seluruh jalur kode untuk AI *hosted*: data pengguna tidak pernah meninggalkan server untuk diproses AI [6]. Subbab ini menjelaskan tiga alasannya, lalu biaya yang jujur harus diakui.

6.3.1 Privasi

Pemilik merek yang memindai situs webnya dengan Profitina sedang memberi tahu platform bisnis apa yang ia jalankan, pesaing mana yang mengkhawatirkannya, pertanyaan apa yang

diajukan pelanggannya, dan di mana kelemahannya. Anggota komunitas yang menulis di kanal kelas memakai nama pengguna, tetapi tetap menulis sebagai seorang manusia. Mengirim bahan seperti itu ke layanan AI pihak ketiga berarti menyalinnya ke server perusahaan lain, di bawah aturan retensi perusahaan lain, kadang di negara lain. *Whitepaper* Profitina menyebut hal ini secara langsung sebagai bagian dari masalah yang hendak dipecahkan: mengirim data merek dan data persaingan ke API pihak ketiga menimbulkan risiko kebocoran dan risiko kepatuhan [6].

Kekhawatiran ini tidak abstrak. Dalam studi tolok ukur privasi Cisco tahun 2025, 64% dari 2.600 profesional privasi dan keamanan menyatakan khawatir secara tidak sengaja membagikan informasi sensitif kepada publik atau pesaing melalui AI generatif, dan hampir separuhnya mengaku pernah memasukkan data pribadi atau data yang tidak publik ke perkakas semacam itu [7]. Studi Cisco tahun 2026, yang terbit pada 26 Januari 2026 dengan 5.200 responden di 12 pasar, menemukan bahwa 86% organisasi percaya data yang disimpan secara lokal pada dasarnya lebih aman, dan 81% menghadapi tuntutan lokalisasi data yang makin tinggi [8].

Undang-Undang Pelindungan Data Pribadi Indonesia, UU PDP Nomor 27 Tahun 2022, memberi kekuatan hukum pada intuisi yang sama. Prinsip minimalisasinya menyatakan bahwa data pribadi hanya diproses sejauh diperlukan untuk tujuannya, transfer lintas negara memerlukan pelindungan yang setara, dan Pasal 10 memberi setiap subjek data hak untuk mengajukan keberatan atas pengambilan keputusan yang hanya didasarkan pada pemrosesan otomatis, termasuk pemrofilan, yang menimbulkan akibat hukum atau berdampak signifikan [9]. Bab 11 membahas undang-undang ini secara rinci. Untuk bab ini intinya sederhana: data pribadi yang paling mudah dilindungi adalah data yang tidak pernah Anda kirim ke mana pun.

6.3.2 Biaya

Model *hosted* memungut biaya per permintaan atau per token. Model lokal memerlukan biaya sekali untuk perangkat keras, lalu biaya kecil per permintaan untuk listrik. Tuliskan n untuk banyaknya permintaan dalam satu periode, p untuk harga layanan *hosted* per permintaan, F untuk biaya tetap perangkat keras lokal selama periode yang sama, dan e untuk biaya operasional lokal per permintaan. Maka

$$C_{\text{hosted}}(n) = n\,p, \qquad C_{\text{lokal}}(n) = F + n\,e \tag{6.2}$$

dan pilihan lokal menjadi lebih murah ketika $F + ne < np$, yaitu ketika

$$n > n^{*} = \frac{F}{p - e} \quad (p > e). \tag{6.3}$$

> **CONTOH.** Angka-angka berikut dibuat-buat dan dibulatkan untuk berlatih Persamaan (6.3), bukan untuk menggambarkan harga sungguhan mana pun. Misalkan sebuah mesin berbiaya $F = \text{Rp}15{,}000{,}000$ untuk setahun layanan, listrik berbiaya $e = \text{Rp}5$ per permintaan, dan layanan *hosted* akan memungut $p = \text{Rp}50$ per permintaan. Maka $n^{*} = 15{,}000{,}000/(50 - 5) \approx 333{,}334$ permintaan per tahun, sedikit di atas 900 per hari. Di bawah volume itu layanan *hosted* lebih murah; di atasnya mesin lokal yang lebih murah. Sekarang ubah satu asumsi: jika mesin itu sudah terbayar karena memang menjalankan aplikasi web, tambahan F untuk AI hampir nol dan inferensi lokal langsung menang.

Aturan internal Profitina menambahkan argumen biaya kedua yang kurang kentara: produk ini sama sekali tidak memakai API pihak ketiga berbayar. Perusahaan rintisan yang setiap fiturnya memanggil layanan berbayar per token memiliki biaya yang tumbuh seiring keberhasilannya sendiri, dapat dinaikkan harganya oleh pemasok dalam semalam, dan tidak dapat menawarkan

paket gratis yang murah hati tanpa rasa waswas. Perusahaan rintisan yang memiliki inferensinya sendiri dapat menawarkan 15 pemindaian gratis per hari (Bab 3) tanpa argometer yang berputar di pusat data orang lain.

6.3.3 Kendali

Alasan ketiga adalah kendali. Model *hosted* dapat diganti namanya, diubah, dibatasi, atau dipensiunkan oleh pemiliknya, dan produk Anda ikut berubah, suka atau tidak. Model lokal adalah sebuah berkas. Versi yang lolos pengujian Anda pada hari Selasa adalah versi yang menjawab pada hari Jumat. Anda yang memutuskan kapan mencoba model yang lebih baru, Anda mengukurnya terhadap model lama dengan contoh-contoh Anda sendiri, dan Anda beralih hanya jika model baru menang, persis disiplin gerbang kualitas. Kendali juga berarti tetap bekerja ketika internet tidak bekerja: model lokal di mesin yang sama dengan aplikasi tidak memiliki lompatan jaringan yang bisa gagal.

Ada manfaat keempat yang lebih senyap: kemandirian pengukuran. Tugas Profitina adalah memperkirakan seberapa siap sebuah situs web dikutip oleh ChatGPT, Gemini, dan Perplexity. Mesin-mesin itu adalah *sasaran* pengukuran, bukan komponennya. Perkakas yang bertanya kepada mesin-mesin itu sendiri akan bergantung padanya, membayarnya, dan hanya melihat apa yang ingin mereka tunjukkan saat itu. Menjaga AI tetap lokal menjaga pengukuran tetap mandiri (Bab 2 dan Bab 3 menjelaskan sinyal kesiapan itu sendiri).

6.3.4 Biaya jujur dari pilihan lokal

Inferensi lokal bukan tanpa kesulitan, dan buku ajar yang hanya menyebut kelebihannya sedang berjualan, bukan mengajar.

- **Batas atas mutu.** Model *hosted* terbesar lebih besar daripada apa pun yang dapat dijalankan tim kecil di satu server. Model lokal 7B menulis paragraf sederhana dengan baik, tetapi tidak akan menyamai model terdepan dalam penalaran yang sulit. Gerbang kualitas ada antara lain karena batas atas ini.
- **Perangkat keras dan operasional.** Seseorang harus memasang server model, mengawasi memorinya, memulainya ulang, dan menyediakan ruang disk untuk berkas model. Bab 9 menunjukkan bagaimana pemeriksaan kesehatan Profitina melaporkan keadaan AI lokal di samping basis data dan disk.
- **Kecepatan.** Model di CPU menghasilkan hanya segelintir token per detik. Itulah sebabnya ada dua jalur (Subbab 6.5).
- **Antre.** Satu mesin hanya dapat menjalankan sedikit generasi sekaligus, sehingga pekerjaan panjang harus antre, dan produk harus jujur memberi tahu pengguna posisi mereka dalam antrean.

Bagi pembaca di Indonesia ada satu pertimbangan tambahan yang patut disebut. Koneksi internet di banyak kota sudah baik, tetapi belum merata antara kota dan desa (Bab 2 memuat angka-angkanya). Sebuah layanan yang setiap fiturnya bergantung pada panggilan ke pusat data di luar negeri mewarisi setiap gangguan jaringan di sepanjang jalan. Server yang menjalankan modelnya sendiri memang tetap memerlukan internet agar penggunanya dapat mengaksesnya, tetapi pekerjaan AI-nya sendiri tidak menambah satu lompatan jaringan pun. Pertimbangan serupa berlaku bagi mahasiswa yang mengerjakan lab bab ini: seluruh lab dapat dijalankan di laptop tanpa koneksi internet sama sekali, karena *stub* dan klien sama-sama berjalan di `127.0.0.1`. Kemandirian semacam ini bukan sekadar kenyamanan teknis; ia adalah bagian dari semangat "dari kita, untuk semua" yang diusung platform ini.

Jadi keputusannya bukan "lokal lebih baik". Keputusannya adalah: bagi platform yang menangani data merek dan data pribadi di Indonesia, yang harus tetap terjangkau, dan yang harus menjaga pengukurannya tetap mandiri, manfaat inferensi lokal lebih besar daripada biaya-biaya ini, dan setiap biaya ditangani oleh sebuah gagasan rancangan di sisa bab ini.

6.4 Ollama dan Keluarga Qwen

6.4.1 Ollama: server model di mesin Anda sendiri

Berkas model di disk tidak melakukan apa pun dengan sendirinya. Harus ada sesuatu yang memuatnya ke memori, memberinya teks, menjalankan perulangan generasi, dan mengembalikan token-tokennya. *Ollama* adalah program gratis dan sumber terbuka yang mengerjakan tugas itu. Anda memasangnya sekali, mengunduh model dengan perintah seperti `ollama pull qwen2.5:7b`, dan sejak itu Ollama mendengarkan permintaan HTTP di komputer Anda sendiri, secara bawaan di porta 11434. Program apa pun yang dapat mengirim permintaan HTTP, dalam bahasa apa pun, lalu dapat memakai model itu. Per 24 September 2026, rilis stabil terbarunya adalah v0.34.3 yang terbit pada 19 September 2026, dengan kandidat rilis v0.34.4 yang keluar pada 23 September [10]. Ollama berubah cepat; bagian yang diandalkan bab ini, yaitu antarmuka HTTP-nya, tetap stabil dari rilis ke rilis.

Jantung antarmuka itu adalah satu *endpoint*, `POST /api/generate` [11]. Permintaannya berupa objek JSON dengan nama `model` dan sebuah `prompt`; kolom opsionalnya antara lain `stream` (bawaannya *true*, artinya jawaban datang sepotong demi sepotong), `format` (isi dengan `"json"` atau berikan skema JSON untuk meminta keluaran terstruktur), `system` (instruksi yang membingkai permintaan), dan `options`, yang memuat pengaturan generasi seperti `temperature`, `seed`, `num_predict` (jumlah maksimum token yang dihasilkan), dan `num_ctx` (ukuran jendela konteks). Jawabannya membawa teks hasil generasi di `response`, penanda `done`, dan kolom-kolom waktu dalam nanodetik: `total_duration`, `load_duration`, `prompt_eval_duration`, dan `eval_duration`, bersama jumlah token `prompt_eval_count` dan `eval_count` [11]. Dua *endpoint* lain penting untuk bab ini: `GET /api/tags` mendaftar model yang terpasang di server, dan `GET /api/version` melaporkan versi server. Subbab 6.8 memakai ketiganya.

> **WAWASAN KUNCI:** Panggil Ollama di `127.0.0.1`, bukan di `localhost`. Kedua nama itu biasanya menunjuk mesin yang sama, tetapi di Windows `localhost` diterjemahkan lebih dahulu ke alamat IPv6, dan percobaan pertama menunggu sebelum beralih ke IPv4, tempat Ollama mendengarkan. Di mesin produksi Windows milik Profitina, perubahan tunggal ini memangkas waktu sampai token pertama dari 2,25 detik menjadi 0,17 detik pada rilis September 2026. Kode lab di bab ini memakai `127.0.0.1` di mana-mana karena alasan yang sama.

6.4.2 Keluarga Qwen, per September 2026

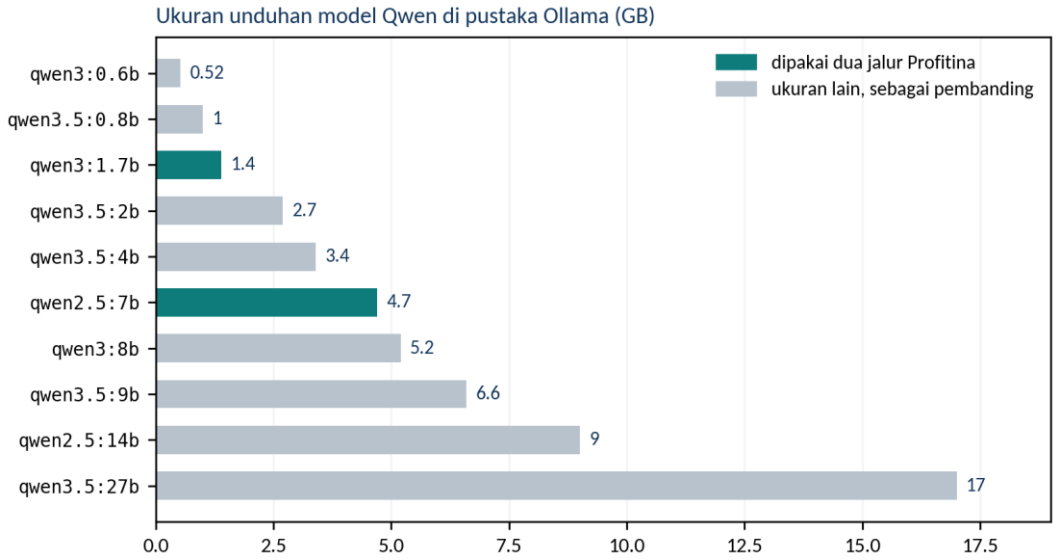
Qwen adalah keluarga model bahasa yang diterbitkan tim Qwen di Alibaba Cloud. Banyak anggotanya dirilis dengan *open weights* (bobot terbuka): siapa saja boleh mengunduh berkas parameternya dan menjalankannya secara lokal, dengan lisensi yang dicantumkan untuk setiap model, paling sering Apache 2.0. Sifat itulah, bukan mode, yang membuat Profitina memakai Qwen: model yang tidak dapat diunduh tidak dapat berjalan di server Anda sendiri. **Tabel 6.1** merangkum generasi yang penting bagi buku ini, diperiksa pada 24 September 2026.

Tabel 6.1 Generasi Qwen yang relevan untuk inferensi lokal, per 24 September 2026. Model "MoE" (*mixture of experts*) seperti 30B-A3B memiliki total 30 miliar parameter tetapi hanya memakai sekitar 3 miliar untuk setiap token.

Generasi	Rilis	Ukuran bobot terbuka (contoh)	Catatan
Qwen2.5	Sep 2024	0.5B, 1.5B, 3B, 7B, 14B, 32B, 72B	18 triliun token pelatihan; Apache 2.0 kecuali 3B dan 72B [12], [5]
Qwen3	Apr 2025	0.6B, 1.7B, 4B, 8B, 14B, 32B; MoE 30B-A3B, 235B-A22B	mode "thinking" dan "non-thinking"; 119 bahasa; Apache 2.0 [13], [14]
Qwen3.5	Feb–Mar 2026	0.8B, 2B, 4B, 9B, 27B, 35B, 122B; 397B-A17B	teks dan gambar; konteks 256K token di Ollama [15], [16]
Qwen3.6	Apr 2026	27B; 35B-A3B	Apache 2.0 [16]
Qwen3.8	Agu 2026	27B; 2.4T-A95B	27B dirilis 14 Agustus 2026 [17]

Dua catatan menjaga tabel ini tetap jujur. Pertama, tidak semua rilis Qwen terbuka: Qwen3.7-Plus, misalnya, hanya ditawarkan sebagai layanan *hosted* [16]. Kedua, lebih baru tidak otomatis lebih baik untuk pekerjaan tertentu. Keluarga yang lebih baru mungkin multimodal, lebih banyak bernalar, atau mendukung masukan lebih panjang, dan semua itu dapat menuntut kecepatan atau memori yang tidak dapat disisihkan oleh satu server.

Gambar 6.1 memperlihatkan ukuran unduhan yang dicantumkan pustaka Ollama untuk beberapa model Qwen, dengan dua model yang dipakai jalur-jalur Profitina disorot. Perhatikan bagaimana Persamaan (6.1) meramalkan polanya: ukuran bertambah kira-kira sebanding dengan jumlah parameter.



Gambar 6.1 Ukuran unduhan model Qwen di pustaka Ollama, dalam gigabita, sebagaimana tercantum pada 24 September 2026 [15], [5], [14]. Dua batang hijau toska adalah model di balik jalur kualitas dan jalur cepat Profitina.

6.4.3 Mengapa Profitina tetap menjalankan Qwen2.5 7B

Pembaca yang baru melihat **Tabel 6.1** mungkin bertanya mengapa produk pada September 2026 menjalankan keluarga model dari 2024 di jalur kualitasnya, berdampingan dengan model 2025 di jalur cepatnya. Jawabannya adalah disiplin yang diajarkan bab ini: model diganti ketika pengukuran menyatakan penggantinya lebih baik untuk pekerjaan *produk ini*, di perangkat keras *produk ini*, dan tidak sebelumnya. Qwen2.5 7B menulis paragraf faktual yang lugas dalam bahasa Indonesia dan Inggris dengan kecepatan yang sanggup dipikul server, dan gerbang Content Engine (Subbab 6.6) mengukur apa yang ditulisnya. Qwen3 1.7B cukup kecil untuk menjawab

pertanyaan klasifikasi pendek dalam sepersekian detik di kartu grafis sederhana. Kedua pilihan itu dapat berubah; perubahan akan datang melalui gerbang yang sama, sebagai kandidat yang harus menang pada tolok ukur produk sendiri, tidak pernah karena muncul nama yang lebih baru.

Profitina juga tidak memberi pengguna pilihan model. Ini disengaja. Pengguna yang memilih model diminta menilai sesuatu yang tidak dapat mereka lihat; platform, yang dapat mengukur, mengambil tanggung jawab itu.

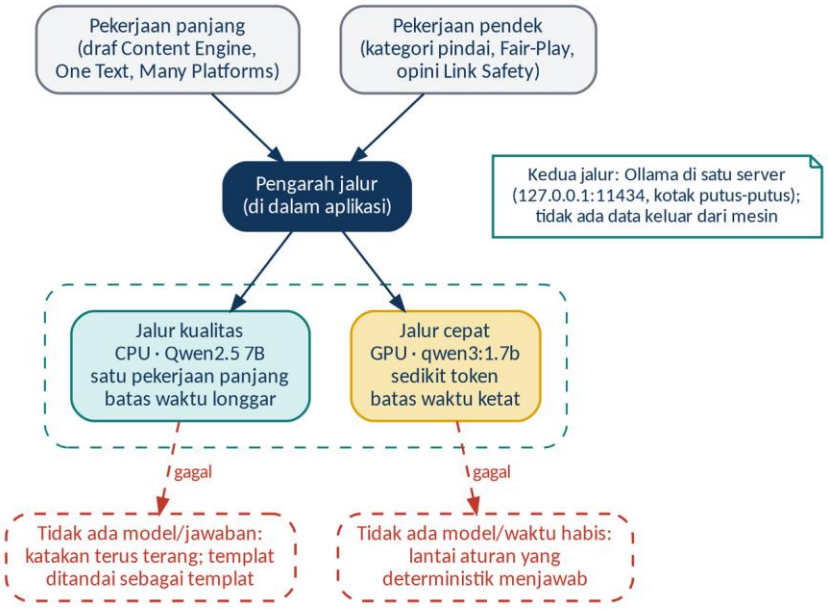
6.5 Model Dua Jalur

6.5.1 Dua jenis pekerjaan

Perhatikan apa saja yang diminta Profitina dari modelnya, dan pekerjaan itu terbagi menjadi dua kelompok.

- **Pekerjaan panjang** menghasilkan teks yang akan dibaca dan diterbitkan orang: draf *answer-first* dari Content Engine beserta FAQ dan data terstrukturnya, atau versi satu teks yang disesuaikan untuk salah satu dari enam platform media sosial dalam *One Text, Many Platforms*. Jawaban ini panjangnya ratusan token, mutunya sangat penting, dan pengguna bersedia menunggunya jika waktu tunggunya diperlihatkan dengan jujur.
- **Pekerjaan pendek** menghasilkan label yang langsung dipakai perangkat lunak: kategori sebuah bisnis ketika pemindai ragu, apakah sebuah unggahan komunitas bermanfaat atau remeh untuk Fair-Play, apakah sebuah tautan tampak seperti penipuan untuk pendapat opsional Link Safety. Jawaban ini hanya beberapa token, dan ia berada di tengah sesuatu yang sedang dikerjakan pengguna, sehingga harus cepat atau tidak ada sama sekali.

Menaruh kedua jenis pekerjaan pada satu model dalam satu antrean menghasilkan yang terburuk dari keduanya: pekerjaan pendek menunggu di belakang pekerjaan panjang, dan pengguna yang mengunggah pesan menunggu artikel orang lain selesai. Karena itu Profitina menjalankan dua *jalur* (*lane*), seperti tampak pada **Gambar 6.2**.



Gambar 6.2 Model dua jalur. Pekerjaan panjang menuju jalur kualitas, pekerjaan pendek ke jalur cepat, keduanya dilayani Ollama di mesin yang sama. Setiap jalur memiliki jalan kegagalan yang jujur.

DEFINISI. *Jalur* (*lane*) adalah rute bagi pekerjaan AI dengan model, perangkat keras, panjang keluaran, batas waktu, dan perilaku gagalnya sendiri. *Jalur kualitas* menjalankan model yang lebih besar untuk teks panjang yang diperiksa dengan cermat; *jalur cepat* menjalankan model yang sangat kecil untuk label pendek dengan batas waktu yang ketat.

Tabel 6.2 menyandingkan kedua jalur itu sebagaimana berjalan pada rilis September 2026.

Tabel 6.2 Jalur kualitas dan jalur cepat pada rilis September 2026.

	Jalur kualitas	Jalur cepat
Model	Qwen2.5 7B	qwen3:1.7b
Perangkat keras	CPU	GPU
Keluaran umum	ratusan token	beberapa token
Contoh pekerjaan	draf Content Engine, One Text, Many Platforms	kategori pindai, kebermanfaatan Fair-Play, opini Link Safety
Konkurensi	satu pekerjaan panjang sekaligus, antrean terlihat	pekerjaan pendek, dijawab cepat
Ketika gagal	katakan terus terang; templat ditandai sebagai templat	lantai aturan deterministik yang menjawab

6.5.2 Mengapa panjang itu penting: sedikit aritmetika

Waktu sebuah generasi kira-kira sama dengan waktu memuat model (nol jika sudah ada di memori), ditambah waktu membaca *prompt*, ditambah waktu menghasilkan jawaban. Jika jawaban terdiri atas n token dan model menghasilkan r token per detik, maka

$$T \approx T_{\text{muat}} + T_{\text{prompt}} + \frac{n}{r}. \tag{6.4}$$

Ollama melaporkan r secara langsung: jumlah token yang dihasilkan adalah `eval_count` dan waktu untuk menghasilkannya adalah `eval_duration` dalam nanodetik, sehingga

$$r = \frac{\text{eval_count}}{\text{eval_duration}/10^{9}} \text{ token per detik.} \tag{6.5}$$

CONTOH. Sebagai ilustrasi, misalkan model 7B di CPU menghasilkan $r = 8$ token per detik dan sebuah draf terdiri atas $n = 400$ token. Menurut Persamaan (6.4), generasinya saja memakan $400/8 = 50$ detik. Misalkan model 1.7B di GPU menghasilkan $r = 60$ token per detik dan sebuah label terdiri atas $n = 3$ token: $3/60 = 0.05$ detik. Perbandingannya seribu banding satu. Jika label itu harus antre di belakang draf, pengguna yang mengunggah pesan akan menunggu 50 detik untuk pekerjaan yang hanya memerlukan seperdua puluh detik.

6.5.3 Satu pekerjaan panjang sekaligus, dan antrean yang jujur

CPU yang menjalankan model 7B sepenuhnya sibuk dengan satu generasi. Memulai generasi kedua tidak menggandakan keluaran; kecepatan keduanya kira-kira menjadi separuh, dan tekanan memori dapat memperburuknya. Karena itu jalur kualitas menjalankan satu pekerjaan panjang sekaligus dan mengantrekan sisanya. Jika ada k pekerjaan di depan Anda dan masing-masing memakan sekitar T_{rata} detik, perkiraan waktu tunggu Anda sebelum mulai adalah

$$W_k \approx k\, T_{\text{rata}}. \tag{6.6}$$

Profitina memperlihatkan persis hal ini kepada pengguna *One Text, Many Platforms*: permintaan diproses satu per satu di seluruh sistem, dan halaman menampilkan posisi antrean serta perkiraan waktu tunggu yang jujur. Operasi panjang juga menampilkan kemajuan terperinci, dengan label sub-langkah seperti "drafting — 466 tokens" dan perkiraan sisa waktu, sehingga pengguna dapat melihat bahwa pekerjaan sedang berlangsung alih-alih menatap ikon berputar [6]. Persamaan (6.6) memang sederhana, tetapi ia mengubah penantian yang cemas menjadi penantian yang terinformasi.

6.5.4 Pekerjaan yang tidak masuk jalur mana pun

Aturan jalur yang paling penting justru tentang kapan model *tidak* boleh dipanggil sama sekali. Pada September 2026 Profitina menyelidiki mengapa berpindah kanal di umpan komunitas terasa sangat lambat. Dari tiga akar masalah, salah satunya adalah bahwa proses menampilkan umpan melakukan panggilan sinkron ke model lokal untuk setiap tautan yang mencurigakan pada setiap tampilan. Perbaikannya menghapus semua panggilan model dari jalur baca. Bersama perbaikan lainnya, umpan berisi 30 unggahan turun dari 868 milidetik menjadi 11 milidetik, dan panggilan AI per halaman turun dari 6 menjadi 0, pada rilis September 2026. Aturan yang kemudian dicatat tim layak dihafal:

> **PERINGATAN:** Fungsi yang dipanggil untuk setiap butir saat menampilkan tidak boleh menyentuh model atau membuka koneksinya sendiri. Nilai sebuah tautan ketika tautan itu diunggah, simpan putusannya, dan biarkan jalur baca hanya membaca putusan yang tersimpan.

Inilah pula sebabnya lantai aturan berada di bawah jalur cepat. Ketika sebuah pekerjaan pendek tidak dapat dijawab model tepat waktu, karena model sibuk, tidak ada, atau lambat, aturan deterministik yang menjawab, dan produk tetap bekerja. Bab 5 menguraikan lantai aturan untuk Link Safety; bentuk yang sama muncul di mana pun jalur cepat dipakai.

6.6 Content Engine dan Gerbang Kualitas

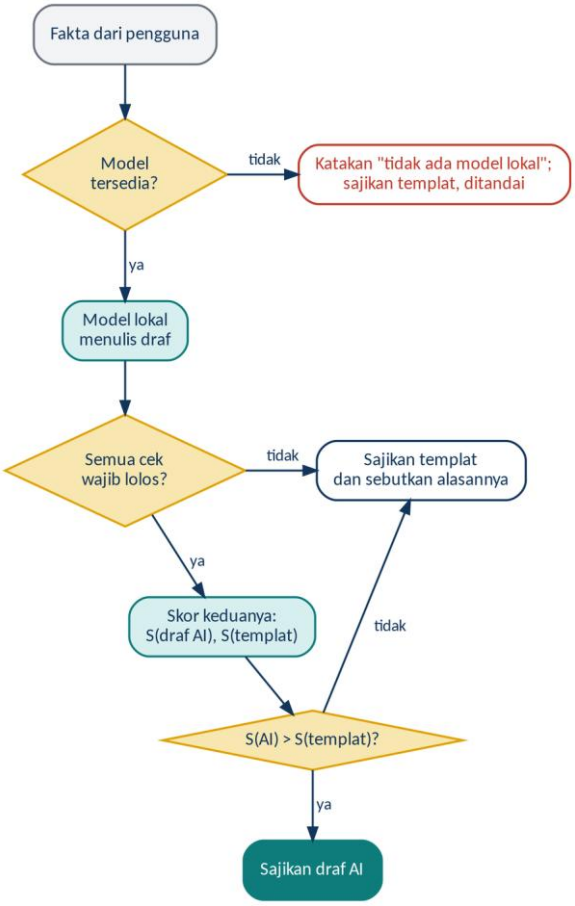
6.6.1 Apa yang dihasilkan Content Engine

Content Engine adalah langkah "Optimasi" dalam siklus Pindai · Optimasi · Pantau Profitina (Bab 3). Setelah pemindaian menemukan bagian-bagian yang menyulitkan mesin jawaban untuk mengutip sebuah bisnis, Content Engine mengubah celah itu menjadi bahan yang siap diterbitkan: paragraf *answer-first* yang pada kalimat pertamanya menyebut siapa bisnis itu dan apa yang ditawarkannya, entitas-entitas kuncinya, sekumpulan pertanyaan yang sering diajukan beserta jawaban langsungnya, dan JSON-LD schema.org yang menggambarkan bisnis itu dalam bentuk yang dapat dibaca mesin. Di sampingnya ada Technical Assistant yang membantu urusan robots.txt untuk *crawler* AI dan memandu pengiriman *sitemap* ke Bing (Bab 2 menjelaskan mengapa Bing penting bagi ChatGPT dan Copilot). Penyusunan draf berjalan di jalur kualitas.

6.6.2 Gerbang: sajikan AI hanya ketika lebih baik

Draf hasil generasi bisa sangat baik, biasa saja, atau keliru dengan cara yang memalukan bila diterbitkan: ia bisa mengarang tahun pendirian, menjanjikan "layanan terbaik di kota", atau melupakan satu-satunya fakta yang diberikan pengguna. Jawaban Profitina adalah *quality gate* (gerbang kualitas). Untuk setiap permintaan, sistem juga memiliki templat bawaan, yaitu teks yang lugas, benar, dan agak kaku yang dibangun langsung dari fakta. Draf AI hanya disajikan jika

mengalahkan templat itu; jika tidak, templat yang disajikan. **Gambar 6.3** memperlihatkan alurnya sebagaimana diterapkan di lab bab ini.



Gambar 6.3 Gerbang kualitas sebagaimana dibangun di lab bab ini. Draf harus lolos semua cek wajib, lalu skornya harus benar-benar lebih tinggi daripada templat; bila tidak ada model yang tersedia, pengguna diberi tahu.

Dalam simbol, tulis d untuk draf AI dan t untuk templat. Misalkan $H(d)$ bernilai benar bila draf lolos semua *cek wajib* (aturan yang tidak boleh dilanggar sama sekali), dan $S(\cdot)$ adalah *skor lunak* yang mengukur seberapa baik sebuah teks. Maka

$$\text{serve}(d, t) = \begin{cases} d & \text{jika } H(d) \text{ dan } S(d) > S(t), \\ t & \text{selainnya.} \end{cases} \tag{6.7}$$

Tiga pilihan rancangan tersembunyi di dalam Persamaan (6.7). Cek wajib didahulukan, sehingga tidak ada skor, setinggi apa pun, yang dapat meloloskan draf yang mengarang fakta. Perbandingannya ketat, $S(d) > S(t)$, sehingga hasil seri dimenangkan templat: bila AI tidak menambahkan apa-apa, teks yang dapat diramalkan yang menang. Dan templat bukanlah lawan yang sengaja dibuat lemah. Templat benar karena cara pembuatannya, sehingga menjadi garis dasar yang bermakna: draf AI yang tidak dapat mengalahkan templat yang benar dan lengkap tidak layak disajikan.

Apa yang dimasukkan ke H dan S adalah keputusan produk. Lab di Subbab 6.8 memakai daftar periksa buatannya sendiri untuk Profitina Laundry: draf harus menyebut nama bisnis di kalimat pertama, tidak boleh memakai kata superlatif seperti “best” atau “terbaik”, tidak boleh memuat angka yang tidak ada di dalam fakta, dan panjangnya harus wajar; skor lunaknya menghitung

berapa fakta yang tercakup dan apakah draf dibuka dalam bentuk *answer-first*. Gerbang Anda sendiri boleh memeriksa hal lain. Yang penting adalah prinsipnya.

6.6.3 Membaca hasil publik: 5 dari 6

Profitina menerbitkan satu angka tentang gerbangnya: setelah revisi Content Engine pada September 2026, draf AI mengalahkan templat pada 5 dari 6 kasus tolok ukur [6]. Tingkat kemenangannya adalah

$$\hat{w} = \frac{5}{6} \approx 0.833. \tag{6.8}$$

Bacalah Persamaan (6.8) dengan cermat. Angka itu menyatakan bahwa mesin yang sudah direvisi biasanya lebih baik daripada templat pada kasus-kasus yang dicoba, dan itulah yang ingin diketahui sebelum menyajikan draf AI sama sekali. Angka itu tidak menyatakan bahwa 83,3% dari semua draf di masa depan akan disajikan, dan tidak menyatakan bahwa kasus keenam adalah bencana; ia menyatakan bahwa pada kasus itu templat setidaknya sama baiknya. Dengan hanya enam kasus, setiap kasus menggeser tingkat kemenangan sebesar $1/6 \approx 16.7$ poin persentase, sehingga satu hasil yang berbeda akan memberi 4 dari 6 atau 6 dari 6. Angka itu jujur justru karena kecil dan spesifik; gerbang, yang berjalan untuk setiap permintaan, itulah yang melindungi setiap pengguna.

6.7 Perilaku Jujur Saat "Tidak Ada Model yang Tersedia"

Model lokal bisa saja tidak ada. Mungkin belum diunduh, server model sedang dimulai ulang, mesin kekurangan memori, atau batas waktu terlewati sebelum token pertama tiba. Apa yang harus dilakukan produk?

Jawaban yang menggoda adalah diam-diam menyajikan templat agar pengguna tidak pernah menyadarinya. Profitina menolak jawaban itu. Ketika tidak ada model yang tersedia, aplikasi *mengatakannya* alih-alih diam-diam menyajikan templat [6]. Pengguna melihat bahwa teks di hadapannya adalah templat, bukan draf AI, dan dapat memutuskan apakah akan menunggu lalu mencoba lagi.

Mengapa bersikeras pada sesuatu yang sekilas tampak seperti mengakui kelemahan? Karena cadangan yang senyap adalah kebohongan kecil, dan kebohongan kecil di perangkat lunak bertumbuh. Pengguna yang mengira sedang membaca draf AI akan menilai AI dari kekakuan templat, tidak akan melaporkan gangguan, dan tidak tahu bahwa ia perlu mencoba lagi. Operator yang tidak pernah melihat cadangan itu di log tidak akan sadar bahwa model sudah mati seharian. Profitina memetik pelajaran yang sama di tempat lain: sebuah `try`/`except` di sekitar penghitungan basis data yang gagal pernah hampir menampilkan angka "0" yang meyakinkan di sebuah dasbor; sejak itu aturannya adalah bahwa galat harus bersuara keras. Produk-produk saudara dalam Profitina Group memegang sikap yang sama, yang dirangkum dalam semboyan "never stop silently" (jangan pernah berhenti diam-diam).

Rancangan yang membuat kejujuran mudah terdiri atas tiga bagian, dan lab membangun ketiganya.

1. **Kegagalan yang khas dan bertipe.** Klien memunculkan eksepsi khusus, `NoModelAvailable`, untuk setiap cara model bisa tidak ada: server tidak dapat dihubungi, model tidak terpasang, atau tidak ada jawaban dalam batas waktu. Pemanggil tidak mungkin mengiranya sebagai hasil biasa.

2. **Alasan yang dapat dibaca manusia.** Eksepsi itu membawa penjelasan singkat seperti "cannot reach http://127.0.0.1:11434 (connection refused)" atau "HTTP 404: model 'qwen2.5:7b' not found", yang dapat masuk ke log dan, dengan kata-kata yang lebih sederhana, sampai ke pengguna.
3. **Cadangan yang diberi label.** Bila ada cadangan, seperti templat atau lantai aturan, keluarannya menyebut siapa yang menjawab. Bila tidak ada, produk menyatakan bahwa fitur itu sedang tidak tersedia.

> **WAWASAN KUNCI:** Menurun dengan anggun dan menurun dengan diam-diam adalah dua hal yang berbeda. Penurunan yang anggun menjaga produk tetap berguna ketika sebuah komponen gagal; kejujuran memberi tahu pengguna versi produk mana yang sedang mereka terima. Sistem yang dapat diandalkan memerlukan keduanya.

6.8 Lab G5: Berbicara dengan Model Lokal melalui HTTP

6.8.1 Apa yang akan kita bangun, dan dari mana keluarannya berasal

Lab ini membangun lima program kecil, masing-masing dalam Python dan Java, hanya dengan pustaka standar setiap bahasa:

1. `stub_ollama.py` / `StubOllama.java`: server kecil yang meniru tiga *endpoint* Ollama dari Subbab 6.4.1, sehingga semuanya berjalan di komputer mana pun, bahkan yang sama sekali tidak memiliki model;
2. `ollama_client.py` / `OllamaClient.java`: klien untuk antarmuka HTTP Ollama yang bekerja tanpa perubahan terhadap *stub* maupun terhadap Ollama sungguhan, dan memunculkan `NoModelAvailable` dengan jujur;
3. `two_lanes.py` / `TwoLanes.java`: pengarah dengan jalur kualitas, jalur cepat, dan lantai aturan;
4. `quality_gate.py` / `QualityGate.java`: gerbang kualitas yang menyajikan draf AI hanya bila mengalahkan templat;
5. `json_label.py` / `JsonLabel.java`: keluaran terstruktur dengan validasi dan satu kali percobaan ulang (penyelesaian Soal 6.6).

Program Java juga memerlukan `MiniJson.java`, pembaca dan penyusun JSON kecil, karena JDK tidak memiliki pustaka JSON sendiri. Kodenya ada di folder `Chapter06` pada kumpulan kode buku ini, lengkap dengan README yang memuat perintah untuk Windows, macOS, dan Linux, baik di terminal maupun di VS Code.

Pernyataan terus terang tentang keluaran yang dicetak di subbab ini: setiap keluaran yang ditampilkan di sini berasal dari server *stub*, bukan dari model bahasa sungguhan. Kami mencoba memasang program Ollama dari halaman rilis GitHub-nya dan menarik model Qwen yang sangat kecil di mesin yang dipakai menyiapkan bab ini, tetapi kebijakan jaringan mesin itu menolak unduhan tersebut (github.com, ollama.com, registri model Ollama, dan Hugging Face semuanya dijawab dengan HTTP 403 oleh *proxy*). Daripada mencetak keluaran model yang dikarang, kami menampilkan persis apa yang dihasilkan *stub*, dan Subbab 6.8.8 menjelaskan apa yang berubah ketika program yang sama diarahkan ke Ollama sungguhan di komputer Anda sendiri. Dengan kata lain, perilaku jujur "tidak ada model yang tersedia" dari Subbab 6.7 diterapkan pada buku ini sendiri.

Server di lab ini mendengarkan di porta 7060 pada `127.0.0.1`, sehingga tidak pernah bertabrakan dengan Ollama sungguhan di porta 11434. Hentikan setiap server dengan Ctrl+C setelah selesai memakainya.

6.8.2 *Stub* yang meniru `/api/generate`

Stub adalah pengganti sebuah komponen sungguhan yang berperilaku sama di antarmukanya tetapi mengerjakan sesuatu yang jauh lebih sederhana di dalamnya. *Stub* kita menjawab `GET /api/version`, `GET /api/tags`, dan `POST /api/generate` dengan nama-nama kolom JSON yang sama dengan Ollama [11], dan menawarkan dua "model", `stub-quality` dan `stub-fast`. Keduanya sama sekali tidak cerdas. Jika *prompt* diawali `CLASSIFY:`, *stub* mengembalikan satu label yang dipilih dengan beberapa aturan kata kunci. Jika tidak, ia mencari baris `Business: …` dan baris-baris yang diawali `-`, lalu menyusun ulang fakta-fakta itu menjadi satu paragraf. Opsi `seed` sengaja memilih salah satu dari tiga perilaku: *seed* 0 menghasilkan draf yang santun, *seed* 1 draf yang menyombong dan mengarang tahun pendirian, dan *seed* 2 draf yang terlalu sedikit isinya. Gerbang di Subbab 6.8.7 memerlukan ketiganya untuk memperlihatkan apa yang dikerjakannya.

Kode 6.1 adalah versi Python, dibangun di atas `http.server`.

Kode 6.1 `stub_ollama.py`: pengganti `/api/version`, `/api/tags`, dan `/api/generate` milik Ollama, dengan *streaming* berupa JSON yang dipisah baris baru.

```
"""A tiny stand-in for Ollama's HTTP API, so the lab runs without a model.

It answers GET /api/version, GET /api/tags and POST /api/generate with the
same JSON shapes as Ollama. The "models" are not intelligent: they only
rearrange the facts found in the prompt, deterministically.

    python stub_ollama.py [port] [--no-models] [--sleep-ms N]
"""
import json
import re
import sys
import time
from http.server import BaseHTTPRequestHandler, ThreadingHTTPServer

MODELS = {  # name -> simulated nanoseconds per generated token
    "stub-quality": 50_000_000,
    "stub-fast": 8_000_000,
}
NO_MODELS = "--no-models" in sys.argv
SLEEP_S = 0.005  # real pause per streamed token, so streaming is visible
if "--sleep-ms" in sys.argv:
    SLEEP_S = int(sys.argv[sys.argv.index("--sleep-ms") + 1]) / 1000


def classify(text):
    """Fast-lane job: one label for a short customer message."""
    t = text.lower()
    if re.search(r"\b(late|lost|broken|torn|refund|rusak|hilang)\b", t):
        return "complaint"
    if re.search(r"\b(thank|thanks|great|mantap)\b", t):
        return "praise"
    if "?" in t:
        return "question"
    return "other"
```

Kode 6.1 (Lanjutan) *stub_ollama.py: pengganti /api/version, /api/tags, dan /api/generate milik Ollama, dengan streaming berupa JSON yang dipisah baris baru.*

```
def draft(prompt, seed):
    """Quality-lane job: an answer-first paragraph built from '- ' facts."""
    name = re.search(r"^Business: (.+)$", prompt, re.M)
    name = name.group(1).strip() if name else "The business"
    facts = [f.strip().rstrip(".") for f in re.findall(r"^- (.+)$", prompt, re.M)]
    if not facts:
        return name + " did not give me any facts to write about."
    variant = seed % 3
    if variant == 0:  # a well-behaved draft
        rest = " ".join(f[0].upper() + f[1:] + "." for f in facts[1:])
        return f"{name} is {facts[0]}. {rest}".strip()
    if variant == 1:  # a boastful draft that invents a fact
        return (f"{name} is the best laundry in town, trusted since 1990. "
                f"It is {facts[0]}.")
    return f"{name}: {facts[0]}."  # a draft that says too little


def answer(model, prompt, options, fmt):
    if prompt.startswith("CLASSIFY:"):
        label = classify(prompt[len("CLASSIFY:"):])
        return json.dumps({"label": label}) if fmt == "json" else label
    return draft(prompt, int(options.get("seed", 0)))


class Handler(BaseHTTPRequestHandler):
    protocol_version = "HTTP/1.1"

    def log_message(self, *args):  # keep the terminal quiet
        pass

    def send_json(self, status, obj):
        body = json.dumps(obj).encode("utf-8")
        self.send_response(status)
        self.send_header("Content-Type", "application/json; charset=utf-8")
        self.send_header("Content-Length", str(len(body)))
        self.end_headers()
        self.wfile.write(body)
```

Kode 6.1 (Lanjutan) *stub_ollama.py: pengganti /api/version, /api/tags, dan /api/generate milik Ollama, dengan streaming berupa JSON yang dipisah baris baru.*

```
    def do_GET(self):
        if self.path == "/api/version":
            self.send_json(200, {"version": "0.0.0-stub"})
        elif self.path == "/api/tags":
            names = [] if NO_MODELS else sorted(MODELS)
            self.send_json(200, {"models": [{"name": n, "model": n}
                                            for n in names]})
        else:
            self.send_json(404, {"error": "not found"})

    def do_POST(self):
        if self.path != "/api/generate":
            self.send_json(404, {"error": "not found"})
            return
        length = int(self.headers.get("Content-Length", "0"))
        try:
            req = json.loads(self.rfile.read(length) or b"{}")
        except json.JSONDecodeError:
            self.send_json(400, {"error": "invalid JSON"})
            return
        model = req.get("model", "")
        if NO_MODELS or model not in MODELS:
            self.send_json(404, {"error": f"model '{model}' not found"})
            return
        text = answer(model, req.get("prompt", ""), req.get("options") or {},
                      req.get("format"))
        tokens = re.findall(r"\S+\s*", text)
        stats = {
            "done": True, "done_reason": "stop",
            "total_duration": 20_000_000 + len(tokens) * MODELS[model],
            "load_duration": 20_000_000,
            "prompt_eval_count": len(req.get("prompt", "").split()),
            "eval_count": len(tokens),
            "eval_duration": len(tokens) * MODELS[model],
        }
        head = {"model": model, "created_at": "2026-09-24T00:00:00Z"}
        if req.get("stream", True) is False:
            self.send_json(200, {**head, "response": text, **stats})
            return
        self.send_response(200)  # newline-delimited JSON, chunked
        self.send_header("Content-Type", "application/x-ndjson")
        self.send_header("Transfer-Encoding", "chunked")
        self.end_headers()
        for tok in tokens:
            self.write_chunk({**head, "response": tok, "done": False})
            time.sleep(SLEEP_S)
```

Kode 6.1 (Lanjutan) *stub_ollama.py: pengganti /api/version, /api/tags, dan /api/generate milik Ollama, dengan streaming berupa JSON yang dipisah baris baru.*

```
        self.write_chunk({**head, "response": "", **stats})
        self.wfile.write(b"0\r\n\r\n")

    def write_chunk(self, obj):
        line = (json.dumps(obj) + "\n").encode("utf-8")
        self.wfile.write(b"%x\r\n%s\r\n" % (len(line), line))
        self.wfile.flush()


def main():
    port = int(sys.argv[1]) if len(sys.argv) > 1 and sys.argv[1].isdigit() \
        else 7060
    server = ThreadingHTTPServer(("127.0.0.1", port), Handler)
    state = ("with no models" if NO_MODELS
             else "models: " + ", ".join(sorted(MODELS)))
    print(f"stub Ollama on http://127.0.0.1:{port} ({state})", flush=True)
    try:
        server.serve_forever()
    except KeyboardInterrupt:
        pass


if __name__ == "__main__":
    main()
```

Bacalah dari bawah ke atas. `main` mengikat `ThreadingHTTPServer` ke `127.0.0.1`, tidak pernah ke semua antarmuka jaringan: *stub* pengembangan tidak punya urusan untuk dapat dijangkau dari mesin lain. Porta opsional serta penanda `--no-models` dan `--sleep-ms` dibaca dari baris perintah. `do_GET` menjawab dua *endpoint* kecil. `do_POST` membaca tepat sebanyak `Content-Length` bita dari badan permintaan, mengurai JSON-nya, dan mengembalikan `404` dengan bentuk galat Ollama, `{"error": "model '…' not found"}`, bila modelnya tidak dikenal atau server dijalankan dengan `--no-models`. Lalu `answer` menghasilkan teks, dan teks itu dipotong menjadi "token" seukuran kata dengan ekspresi reguler. Blok statistik meniru kolom-kolom Ollama; durasinya *disimulasikan*, tetap 50 milidetik per token untuk `stub-quality` dan 8 milidetik untuk `stub-fast`, supaya laju yang dicetak nanti dapat diulang. Bila permintaan berisi `"stream": false`, satu objek JSON dikembalikan. Jika tidak, *stub* mengirim satu objek JSON per baris dengan *chunked transfer encoding* HTTP: setiap potongan berupa panjangnya dalam heksadesimal, pindah baris, bita-bitanya, dan pindah baris lagi, dan potongan sepanjang nol mengakhiri jawaban. Begitulah persisnya Ollama sungguhan melakukan *streaming*.

Kode 6.2 adalah server yang sama dalam Java, memakai `com.sun.net.httpserver.HttpServer` bawaan JDK.

Kode 6.2 `StubOllama.java`: *stub* yang sama dengan `com.sun.net.httpserver`; `sendResponseHeaders(200, 0)` memilih *streaming* bertahap (*chunked*).

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;
import java.util.concurrent.Executors;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** A tiny stand-in for Ollama's HTTP API, so the lab runs without a model.
 *  java StubOllama [port] [--no-models] [--sleep-ms N] */
public class StubOllama {
    static final Map<String, Long> MODELS = new TreeMap<>(Map.of(
            "stub-quality", 50_000_000L,           // simulated ns per token
            "stub-fast", 8_000_000L));
    static boolean noModels = false;
    static long sleepMs = 5;                       // real pause per streamed token

    static String classify(String text) {
        String t = text.toLowerCase();
        if (Pattern.compile("\\b(late|lost|broken|torn|refund|rusak|hilang)\\b")
                .matcher(t).find()) return "complaint";
        if (Pattern.compile("\\b(thank|thanks|great|mantap)\\b")
                .matcher(t).find()) return "praise";
        return t.contains("?") ? "question" : "other";
    }

    static String draft(String prompt, long seed) {
        Matcher mn = Pattern.compile("(?m)^Business: (.+)$").matcher(prompt);
        String name = mn.find() ? mn.group(1).strip() : "The business";
        List<String> facts = new ArrayList<>();
        Matcher mf = Pattern.compile("(?m)^- (.+)$").matcher(prompt);
        while (mf.find()) facts.add(mf.group(1).strip().replaceAll("\\.+$", ""));
        if (facts.isEmpty()) {
            return name + " did not give me any facts to write about.";
        }
        long variant = seed % 3;
        if (variant == 0) {                        // a well-behaved draft
            StringBuilder rest = new StringBuilder();
```

Kode 6.2 (Lanjutan) *StubOllama.java: stub yang sama dengan com.sun.net.httpserver; sendResponseHeaders(200, 0) memilih streaming bertahap (chunked).*

```
            for (String f : facts.subList(1, facts.size())) {
                rest.append(Character.toUpperCase(f.charAt(0)))
                    .append(f.substring(1)).append(". ");
            }
            return (name + " is " + facts.get(0) + ". " + rest).strip();
        }
        if (variant == 1) {                       // boastful, invents a fact
            return name + " is the best laundry in town, trusted since 1990. "
                    + "It is " + facts.get(0) + ".";
        }
        return name + ": " + facts.get(0) + ".";   // says too little
    }

    static String answer(String prompt, Map<String, Object> options, Object fmt) {
        if (prompt.startsWith("CLASSIFY:")) {
            String label = classify(prompt.substring("CLASSIFY:".length()));
            return "json".equals(fmt) ? MiniJson.write(Map.of("label", label))
                                      : label;
        }
        Object seed = options.getOrDefault("seed", 0L);
        return draft(prompt, ((Number) seed).longValue());
    }

    static void sendJson(HttpExchange ex, int status, Object obj)
            throws IOException {
        byte[] body = MiniJson.write(obj).getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type",
                                    "application/json; charset=utf-8");
        ex.sendResponseHeaders(status, body.length);
        try (OutputStream out = ex.getResponseBody()) { out.write(body); }
    }

    static void handle(HttpExchange ex) throws IOException {
        String path = ex.getRequestURI().getPath();
        String method = ex.getRequestMethod();
        if (method.equals("GET") && path.equals("/api/version")) {
            sendJson(ex, 200, Map.of("version", "0.0.0-stub"));
        } else if (method.equals("GET") && path.equals("/api/tags")) {
            List<Object> list = new ArrayList<>();
            if (!noModels) {
                for (String n : MODELS.keySet()) {
                    list.add(MiniJson.obj("name", n, "model", n));
                }
            }
            sendJson(ex, 200, Map.of("models", list));
        } else if (method.equals("POST") && path.equals("/api/generate")) {
```

Kode 6.2 (Lanjutan) *StubOllama.java: stub yang sama dengan com.sun.net.httpserver; sendResponseHeaders(200, 0) memilih streaming bertahap (chunked).*

```
            generate(ex);
        } else {
            sendJson(ex, 404, Map.of("error", "not found"));
        }
    }

    @SuppressWarnings("unchecked")
    static void generate(HttpExchange ex) throws IOException {
        Map<String, Object> req;
        try {
            String raw = new String(ex.getRequestBody().readAllBytes(),
                                    StandardCharsets.UTF_8);
            req = MiniJson.parseObject(raw.isBlank() ? "{}" : raw);
        } catch (RuntimeException e) {
            sendJson(ex, 400, Map.of("error", "invalid JSON"));
            return;
        }
        String model = String.valueOf(req.getOrDefault("model", ""));
        if (noModels || !MODELS.containsKey(model)) {
            sendJson(ex, 404, Map.of("error", "model '" + model + "' not found"));
            return;
        }
        String prompt = String.valueOf(req.getOrDefault("prompt", ""));
        Object opts = req.get("options");
        Map<String, Object> options =
                opts instanceof Map ? (Map<String, Object>) opts : Map.of();
        String text = answer(prompt, options, req.get("format"));
        List<String> tokens = new ArrayList<>();
        Matcher m = Pattern.compile("\\S+\\s*").matcher(text);
        while (m.find()) tokens.add(m.group());
        long perToken = MODELS.get(model), n = tokens.size();
        Map<String, Object> head = MiniJson.obj("model", model,
                "created_at", "2026-09-24T00:00:00Z");
        Map<String, Object> stats = MiniJson.obj(
                "done", true, "done_reason", "stop",
                "total_duration", 20_000_000L + n * perToken,
                "load_duration", 20_000_000L,
                "prompt_eval_count", (long) prompt.split("\\s+").length,
                "eval_count", n, "eval_duration", n * perToken);
        if (Boolean.FALSE.equals(req.get("stream"))) {
            Map<String, Object> all = new LinkedHashMap<>(head);
            all.put("response", text);
            all.putAll(stats);
            sendJson(ex, 200, all);
            return;
        }
```

Kode 6.2 (Lanjutan) *StubOllama.java: stub yang sama dengan com.sun.net.httpserver; sendResponseHeaders(200, 0) memilih streaming bertahap (chunked).*

```
        ex.getResponseHeaders().set("Content-Type", "application/x-ndjson");
        ex.sendResponseHeaders(200, 0);            // 0 = chunked, length unknown
        try (OutputStream out = ex.getResponseBody()) {
            for (String tok : tokens) {
                Map<String, Object> chunk = new LinkedHashMap<>(head);
                chunk.put("response", tok);
                chunk.put("done", false);
                writeLine(out, chunk);
                try {
                    Thread.sleep(sleepMs);
                } catch (InterruptedException e) {
                    return;
                }
            }
            Map<String, Object> last = new LinkedHashMap<>(head);
            last.put("response", "");
            last.putAll(stats);
            writeLine(out, last);
        }
    }

    static void writeLine(OutputStream out, Object obj) throws IOException {
        out.write((MiniJson.write(obj) + "\n").getBytes(StandardCharsets.UTF_8));
        out.flush();
    }

    public static void main(String[] args) throws IOException {
        List<String> a = Arrays.asList(args);
        int port = !a.isEmpty() && a.get(0).matches("\\d+")
                ? Integer.parseInt(a.get(0)) : 7060;
        noModels = a.contains("--no-models");
        if (a.contains("--sleep-ms")) {
            sleepMs = Long.parseLong(a.get(a.indexOf("--sleep-ms") + 1));
        }
        HttpServer server =
                HttpServer.create(new InetSocketAddress("127.0.0.1", port), 0);
        server.createContext("/", ex -> {
            try { handle(ex); } finally { ex.close(); }
        });
        server.setExecutor(Executors.newCachedThreadPool());
        server.start();
        String state = noModels ? "with no models"
                                : "models: " + String.join(", ", MODELS.keySet());
        System.out.println("stub Ollama on http://127.0.0.1:" + port
                           + " (" + state + ")");
    }
```

Kode 6.2 (Lanjutan) *StubOllama.java: stub yang sama dengan com.sun.net.httpserver; sendResponseHeaders(200, 0) memilih streaming bertahap (chunked).*

```
}
```

Strukturnya sama baris demi baris dengan versi Python. Dua detail Java patut dicatat. `ex.sendResponseHeaders(200, 0)` berarti "panjang tidak diketahui", yang membuat server memakai pengodean *chunked* untuk jawaban yang di-*stream*, sedangkan panjang positif mengirim jawaban biasa. Dan `TreeMap` menjaga nama-nama model tetap terurut, sehingga kedua *stub* mendaftarnya dalam urutan yang sama.

Kedua program Java membaca dan menulis JSON dengan **Kode 6.3**, pengurai dan penyusun yang sengaja dibuat kecil. Ia menangani objek, larik, *string* dengan karakter *escape*, bilangan, `true`, `false`, dan `null`, yaitu semua yang dikirim Ollama dalam format JSON sebagaimana didefinisikan

RFC 8259 [18], dan penyusunnya memakai spasi yang sama dengan `json.dumps` milik Python, sehingga kedua *stub* menghasilkan jawaban yang identik bita demi bita.

Kode 6.3 `MiniJson.java`: pembaca JSON *recursive-descent* yang minimal beserta penyusunnya, cukup untuk lab ini.

```
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

/** Just enough JSON for this lab: objects, arrays, strings, numbers,
 *  true, false and null. Parsed objects become LinkedHashMap. */
public final class MiniJson {
    private final String s;
    private int i;

    private MiniJson(String s) { this.s = s; }

    public static Object parse(String text) {
        MiniJson p = new MiniJson(text);
        Object v = p.value();
        p.ws();
        if (p.i != p.s.length()) throw p.err("trailing characters");
        return v;
    }

    @SuppressWarnings("unchecked")
    public static Map<String, Object> parseObject(String text) {
        Object v = parse(text);
        if (!(v instanceof Map)) {
            throw new IllegalArgumentException("not an object");
        }
        return (Map<String, Object>) v;
    }

    private IllegalArgumentException err(String what) {
        return new IllegalArgumentException("bad JSON at " + i + ": " + what);
    }

    private void ws() {
        while (i < s.length() && Character.isWhitespace(s.charAt(i))) i++;
    }
```

Kode 6.3 (Lanjutan) *MiniJson.java: pembaca JSON recursive-descent yang minimal beserta penyusunnya, cukup untuk lab ini.*

```
    private Object value() {
        ws();
        if (i >= s.length()) throw err("unexpected end");
        char c = s.charAt(i);
        if (c == '{') return object();
        if (c == '[') return array();
        if (c == '"') return string();
        if (s.startsWith("true", i)) { i += 4; return Boolean.TRUE; }
        if (s.startsWith("false", i)) { i += 5; return Boolean.FALSE; }
        if (s.startsWith("null", i)) { i += 4; return null; }
        return number();
    }

    private Map<String, Object> object() {
        Map<String, Object> m = new LinkedHashMap<>();
        i++;                                        // skip {
        ws();
        if (s.charAt(i) == '}') { i++; return m; }
        while (true) {
            ws();
            String key = string();
            ws();
            if (s.charAt(i++) != ':') throw err("':' expected");
            m.put(key, value());
            ws();
            char c = s.charAt(i++);
            if (c == '}') return m;
            if (c != ',') throw err("',' or '}' expected");
        }
    }

    private List<Object> array() {
        List<Object> a = new ArrayList<>();
        i++;                                        // skip [
        ws();
        if (s.charAt(i) == ']') { i++; return a; }
        while (true) {
            a.add(value());
            ws();
            char c = s.charAt(i++);
            if (c == ']') return a;
            if (c != ',') throw err("',' or ']' expected");
        }
    }
```

Kode 6.3 (Lanjutan) *MiniJson.java: pembaca JSON recursive-descent yang minimal beserta penyusunnya, cukup untuk lab ini.*

```
    private String string() {
        if (s.charAt(i) != '"') throw err("string expected");
        StringBuilder b = new StringBuilder();
        i++;
        while (true) {
            char c = s.charAt(i++);
            if (c == '"') return b.toString();
            if (c != '\\') { b.append(c); continue; }
            char e = s.charAt(i++);
            switch (e) {
                case 'n' -> b.append('\n');
                case 't' -> b.append('\t');
                case 'r' -> b.append('\r');
                case 'b' -> b.append('\b');
                case 'f' -> b.append('\f');
                case 'u' -> {
                    b.append((char) Integer.parseInt(s.substring(i, i + 4), 16));
                    i += 4;
                }
                default -> b.append(e);            // \" \\ \/
            }
        }
    }

    private Object number() {
        int start = i;
        while (i < s.length() && "+-0123456789.eE".indexOf(s.charAt(i)) >= 0) i++;
        String t = s.substring(start, i);
        if (t.isEmpty()) throw err("value expected");
        if (t.contains(".") || t.contains("e") || t.contains("E")) {
            return Double.parseDouble(t);
        }
        return Long.parseLong(t);
    }

    /** Write a value as JSON, with the same spacing as Python's json.dumps. */
    public static String write(Object v) {
        StringBuilder b = new StringBuilder();
        write(v, b);
        return b.toString();
    }
```

Kode 6.3 (Lanjutan) *MiniJson.java: pembaca JSON recursive-descent yang minimal beserta penyusunnya, cukup untuk lab ini.*

```
    private static void write(Object v, StringBuilder b) {
        if (v == null) b.append("null");
        else if (v instanceof String str) quote(str, b);
        else if (v instanceof Map<?, ?> m) {
            b.append('{');
            boolean first = true;
            for (Map.Entry<?, ?> e : m.entrySet()) {
                if (!first) b.append(", ");
                first = false;
                quote(String.valueOf(e.getKey()), b);
                b.append(": ");
                write(e.getValue(), b);
            }
            b.append('}');
        } else if (v instanceof List<?> list) {
            b.append('[');
            for (int k = 0; k < list.size(); k++) {
                if (k > 0) b.append(", ");
                write(list.get(k), b);
            }
            b.append(']');
        } else b.append(v);                          // numbers and booleans
    }

    private static void quote(String str, StringBuilder b) {
        b.append('"');
        for (char c : str.toCharArray()) {
            switch (c) {
                case '"' -> b.append("\\\"");
                case '\\' -> b.append("\\\\");
                case '\n' -> b.append("\\n");
                case '\r' -> b.append("\\r");
                case '\t' -> b.append("\\t");
                default -> {
                    if (c >= 0x20 && c <= 0x7e) b.append(c);
                    else b.append(String.format("\\u%04x", (int) c));
                }
            }
        }
        b.append('"');
    }
```

Kode 6.3 (Lanjutan) *MiniJson.java: pembaca JSON recursive-descent yang minimal beserta penyusunnya, cukup untuk lab ini.*

```
    /** Small helper for building objects: obj("a", 1, "b", "x"). */
    public static Map<String, Object> obj(Object... kv) {
        Map<String, Object> m = new LinkedHashMap<>();
        for (int k = 0; k < kv.length; k += 2) m.put((String) kv[k], kv[k + 1]);
        return m;
    }
}
```

`MiniJson` adalah pengurai *recursive-descent* (turunan rekursif): `value` melihat karakter berikutnya lalu menyerahkan pekerjaan ke `object`, `array`, `string`, atau `number`, dan `object` serta `array` memanggil `value` lagi untuk setiap elemen, sehingga JSON bersarang ditangani oleh panggilan bersarang. Dalam proyek yang lebih besar Anda akan memakai pustaka seperti Jackson atau Gson; di sini, menghindari dependensi membuat lab tetap dapat dijalankan hanya dengan JDK.

6.8.3 Menjalankan *stub*

Buka terminal di `Chapter06/python` lalu jalankan *stub* Python, atau di `Chapter06/java` kompilasi dan jalankan versi Java. Pakai salah satu saja: keduanya mendengarkan di porta 7060.

```
python stub_ollama.py
javac *.java
java StubOllama
```

Keduanya mencetak satu baris lalu menunggu permintaan:

```
stub Ollama on http://127.0.0.1:7060 (models: stub-fast, stub-quality)
```

Di macOS dan Linux, perintah Python mungkin `python3`. Biarkan terminal ini terbuka dan pakai terminal kedua untuk program-program berikutnya.

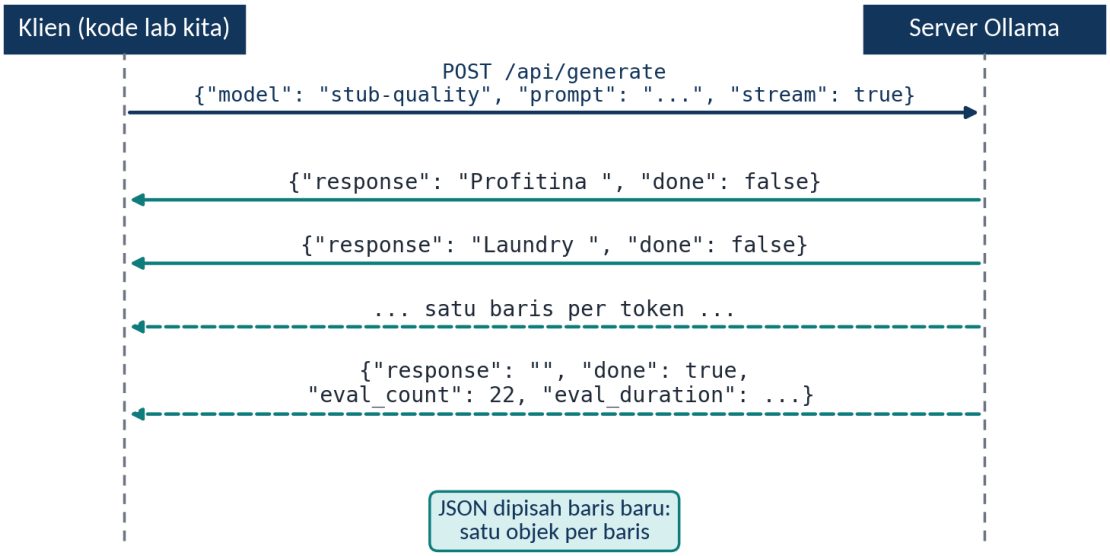
6.8.4 Mengintip kabelnya

Sebelum menulis klien, ada gunanya melihat pertukaran mentahnya. Dengan *stub* berjalan, kirim satu permintaan *streaming* dengan `curl` dan rapikan setiap baris jawabannya dengan modul `json.tool` milik Python (di Windows, ketik `curl.exe` di PowerShell, karena `curl` di sana adalah alias untuk perintah lain, dan taruh JSON-nya di sebuah berkas dengan `-d @request.json` bila tanda kutip menyulitkan):

```
curl -s http://127.0.0.1:7060/api/generate -d '{"model": "stub-fast", "prompt":
"CLASSIFY: thanks, great ironing"}' | python -m json.tool --json-lines
```

```
{
    "model": "stub-fast",
    "created_at": "2026-09-24T00:00:00Z",
    "response": "praise",
    "done": false
}
{
    "model": "stub-fast",
    "created_at": "2026-09-24T00:00:00Z",
    "response": "",
    "done": true,
    "done_reason": "stop",
    "total_duration": 28000000,
    "load_duration": 20000000,
    "prompt_eval_count": 4,
    "eval_count": 1,
    "eval_duration": 8000000
}
```

Dua objek JSON tiba, satu per baris. Yang pertama membawa satu-satunya token jawaban, `praise`, dengan `"done": false`. Yang kedua membawa `response` kosong, `"done": true`, dan statistiknya. Menurut Persamaan (6.5), laju generasi simulasi *stub* adalah $1/(8{,}000{,}000/10^9) = 125$ token per detik. **Gambar 6.4** menggambar pertukaran yang sama untuk jawaban yang lebih panjang.



Gambar 6.4 Panggilan *streaming* ke `/api/generate`. Klien mengirim satu permintaan JSON; server menjawab dengan satu objek JSON per baris, dan objek terakhir bertanda `"done": true` serta membawa jumlah token dan durasinya.

Sebagai pembanding, dokumentasi Ollama memberikan contoh jawaban lengkap tanpa *streaming* dari server sungguhan berikut ini (dikutip dari dokumentasi [11], bukan hasil kami):

```
{
  "model": "gemma4",
  "created_at": "2025-10-17T23:14:07.414671Z",
  "response": "Hello! How can I help you today?",
  "done": true,
  "done_reason": "stop",
  "total_duration": 174560334,
  "load_duration": 101397084,
  "prompt_eval_count": 11,
  "prompt_eval_duration": 13074791,
  "eval_count": 18,
  "eval_duration": 52479709
}
```

Nama dan makna kolomnya sama dengan yang dipakai *stub* kita. Menerapkan Persamaan (6.5) pada contoh dokumentasi itu memberi $18/0.0525 \approx 343$ token per detik pada perangkat keras apa pun yang menghasilkannya.

6.8.5 Klien

Kode 6.4 adalah klien dalam Python.

Kode 6.4 `ollama_client.py`: klien pustaka standar untuk antarmuka HTTP Ollama, dengan satu eksepsi jujur untuk setiap cara model bisa tidak ada.

```
"""A small client for Ollama's HTTP API, standard library only.

    python ollama_client.py [--host URL] [--model NAME] [--stream] PROMPT

Works the same against a real Ollama (default http://127.0.0.1:11434) and
against stub_ollama.py (http://127.0.0.1:7060).
"""
import json
import sys
import urllib.error
import urllib.request

DEFAULT_HOST = "http://127.0.0.1:11434"  # 127.0.0.1, not localhost


class NoModelAvailable(Exception):
    """No local model can answer. The caller must say so, not pretend."""


class OllamaClient:
    def __init__(self, host=DEFAULT_HOST, timeout=60.0):
        self.host = host.rstrip("/")
        self.timeout = timeout
        # Local traffic must never be sent through a proxy.
        self.opener = urllib.request.build_opener(
            urllib.request.ProxyHandler({}))

    def _open(self, path, payload=None):
        data = None if payload is None else json.dumps(payload).encode()
        req = urllib.request.Request(
            self.host + path, data=data,
            headers={"Content-Type": "application/json"})
        try:
            return self.opener.open(req, timeout=self.timeout)
        except urllib.error.HTTPError as e:
            detail = e.read().decode("utf-8", "replace")
            try:
                detail = json.loads(detail).get("error", detail)
            except json.JSONDecodeError:
                pass
            raise NoModelAvailable(f"HTTP {e.code}: {detail}") from None
        except OSError as e:  # URLError, refused, timed out, ...
            reason = getattr(e, "reason", e)
            if isinstance(reason, TimeoutError):  # socket.timeout too
                raise NoModelAvailable(
                    f"no answer within {self.timeout} s") from None
```

Kode 6.4 (Lanjutan) *ollama_client.py: klien pustaka standar untuk antarmuka HTTP Ollama, dengan satu eksepsi jujur untuk setiap cara model bisa tidak ada.*

```
            if isinstance(reason, ConnectionRefusedError):
                reason = "connection refused"
            raise NoModelAvailable(
                f"cannot reach {self.host} ({reason})") from None

    def models(self):
        with self._open("/api/tags") as resp:
            return [m["name"] for m in json.load(resp).get("models", [])]

    def generate(self, model, prompt, options=None, fmt=None):
        """One request, one complete answer (stream: false)."""
        payload = {"model": model, "prompt": prompt, "stream": False}
        if options:
            payload["options"] = options
        if fmt:
            payload["format"] = fmt
        try:
            with self._open("/api/generate", payload) as resp:
                return json.load(resp)
        except TimeoutError:  # the answer started but did not finish
            raise NoModelAvailable(
                f"no answer within {self.timeout} s") from None

    def stream(self, model, prompt, options=None):
        """Yield the partial answers as they arrive (stream: true)."""
        payload = {"model": model, "prompt": prompt, "stream": True}
        if options:
            payload["options"] = options
        with self._open("/api/generate", payload) as resp:
            for line in resp:  # one JSON object per line
                if line.strip():
                    yield json.loads(line)


def main(argv):
    host, model, stream, words = DEFAULT_HOST, "qwen2.5:7b", False, []
    i = 0
    while i < len(argv):
        if argv[i] == "--host":
            host, i = argv[i + 1], i + 2
        elif argv[i] == "--model":
            model, i = argv[i + 1], i + 2
        elif argv[i] == "--stream":
            stream, i = True, i + 1
        else:
            words, i = words + [argv[i]], i + 1
```

Kode 6.4 (Lanjutan) *ollama_client.py: klien pustaka standar untuk antarmuka HTTP Ollama, dengan satu eksepsi jujur untuk setiap cara model bisa tidak ada.*

```
    prompt = " ".join(words) or "Say hello in one short sentence."
    client = OllamaClient(host)
    try:
        print("models on server:", ", ".join(client.models()) or "(none)")
        if stream:
            for chunk in client.stream(model, prompt):
                print(chunk["response"], end="", flush=True)
                if chunk["done"]:
                    final = chunk
            print()
        else:
            final = client.generate(model, prompt)
            print(final["response"])
    except NoModelAvailable as e:
        print("No local model available:", e)
        return 2
    rate = final["eval_count"] / (final["eval_duration"] / 1e9)
    print(f"[{final['model']}: eval_count={final['eval_count']}, "
          f"{rate:.1f} tokens/s as reported by the server]")
    return 0


if __name__ == "__main__":
    sys.exit(main(sys.argv[1:]))
```

Tiga keputusan membentuk klien ini.

- **Tanpa *proxy* untuk lalu lintas lokal.** `urllib` biasanya mengikuti pengaturan *proxy* dari lingkungan. Banyak jaringan kantor dan kampus memasang *proxy*, dan mengirim permintaan untuk `127.0.0.1` melaluinya gagal dengan cara yang membingungkan. `ProxyHandler({})` membangun *opener* yang tidak pernah memakai *proxy*. Klien Java membuat pilihan yang sama dengan `HttpClient.Builder.NO_PROXY`.
- **Satu eksepsi, banyak sebab.** `_open` mengubah setiap kegagalan menjadi `NoModelAvailable` dengan alasan yang dapat dibaca: galat HTTP seperti 404 (kolom `error` dari server diambil dari badan JSON-nya), koneksi yang ditolak (tidak ada yang mendengarkan), atau batas waktu habis (server mendengarkan tetapi tidak menjawab tepat waktu). Pemanggil cukup memiliki satu klausa `except`, dan tidak mungkin lupa bahwa model bisa tidak ada, karena eksepsi itulah satu-satunya cara klien melaporkannya.
- **Dua cara menerima jawaban.** `generate` meminta `"stream": false` dan mengembalikan satu *dictionary*. `stream` meminta *streaming* dan merupakan sebuah *generator*: ia melakukan `yield` untuk setiap baris JSON begitu baris itu tiba, sehingga pemanggil dapat mencetak token selagi model masih menghasilkannya, dan itulah yang membuat draf panjang terasa hidup.

Kode 6.5 adalah versi Java.

Kode 6.5 `OllamaClient.java`: klien yang sama dengan `java.net.http.HttpClient`; *streaming* memakai `BodyHandlers.ofLines()`.

```
import java.io.IOException;
import java.net.ConnectException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.net.http.HttpTimeoutException;
import java.time.Duration;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.function.Consumer;
import java.util.stream.Stream;

/** A small client for Ollama's HTTP API, JDK standard library only.
 *  java OllamaClient [--host URL] [--model NAME] [--stream] PROMPT */
public class OllamaClient {
    public static final String DEFAULT_HOST = "http://127.0.0.1:11434";

    /** No local model can answer. The caller must say so, not pretend. */
    public static class NoModelAvailable extends Exception {
        public NoModelAvailable(String message) { super(message); }
    }

    private final String host;
    private final double timeoutS;
    private final HttpClient http;

    public OllamaClient(String host, double timeoutS) {
        this.host = host.replaceAll("/+$", "");
        this.timeoutS = timeoutS;
        this.http = HttpClient.newBuilder()
                .proxy(HttpClient.Builder.NO_PROXY)  // local traffic, no proxy
                .connectTimeout(Duration.ofMillis((long) (timeoutS * 1000)))
                .build();
    }
```

Kode 6.5 (Lanjutan) *OllamaClient.java: klien yang sama dengan java.net.http.HttpClient; streaming memakai BodyHandlers.ofLines().*

```
    private <T> HttpResponse<T> send(String path, Object payload,
                                     HttpResponse.BodyHandler<T> handler)
            throws NoModelAvailable {
        HttpRequest.Builder b = HttpRequest.newBuilder(URI.create(host + path))
                .timeout(Duration.ofMillis((long) (timeoutS * 1000)))
                .header("Content-Type", "application/json");
        if (payload == null) b.GET();
        else b.POST(HttpRequest.BodyPublishers.ofString(MiniJson.write(payload)));
        try {
            return http.send(b.build(), handler);
        } catch (HttpTimeoutException e) {
            throw new NoModelAvailable("no answer within " + timeoutS + " s");
        } catch (ConnectException e) {
            throw new NoModelAvailable("cannot reach " + host
                                       + " (connection refused)");
        } catch (IOException e) {
            throw new NoModelAvailable("cannot reach " + host + " (" + e + ")");
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            throw new NoModelAvailable("interrupted");
        }
    }

    private static void check(int status, String body) throws NoModelAvailable {
        if (status == 200) return;
        String detail = body;
        try {
            Object err = MiniJson.parseObject(body).get("error");
            if (err != null) detail = err.toString();
        } catch (RuntimeException ignored) {
            // the body was not JSON; keep it as it is
        }
        throw new NoModelAvailable("HTTP " + status + ": " + detail);
    }

    @SuppressWarnings("unchecked")
    public List<String> models() throws NoModelAvailable {
        HttpResponse<String> r = send("/api/tags", null,
                                      HttpResponse.BodyHandlers.ofString());
        check(r.statusCode(), r.body());
        List<String> names = new ArrayList<>();
        Object models = MiniJson.parseObject(r.body()).get("models");
        for (Object m : (List<Object>) models) {
            names.add((String) ((Map<String, Object>) m).get("name"));
        }
        return names;
```

Kode 6.5 (Lanjutan) *OllamaClient.java: klien yang sama dengan java.net.http.HttpClient; streaming memakai BodyHandlers.ofLines().*

```
    }

    /** One request, one complete answer (stream: false). */
    public Map<String, Object> generate(String model, String prompt,
            Map<String, Object> options, String format) throws NoModelAvailable {
        Map<String, Object> payload = MiniJson.obj(
                "model", model, "prompt", prompt, "stream", false);
        if (options != null && !options.isEmpty()) payload.put("options", options);
        if (format != null) payload.put("format", format);
        HttpResponse<String> r = send("/api/generate", payload,
                                      HttpResponse.BodyHandlers.ofString());
        check(r.statusCode(), r.body());
        return MiniJson.parseObject(r.body());
    }

    /** Hand each partial answer to onChunk as it arrives (stream: true). */
    public void stream(String model, String prompt, Map<String, Object> options,
                       Consumer<Map<String, Object>> onChunk)
            throws NoModelAvailable {
        Map<String, Object> payload = MiniJson.obj(
                "model", model, "prompt", prompt, "stream", true);
        if (options != null && !options.isEmpty()) payload.put("options", options);
        HttpResponse<Stream<String>> r = send("/api/generate", payload,
                                              HttpResponse.BodyHandlers.ofLines());
        if (r.statusCode() != 200) {
            check(r.statusCode(), String.join("\n", r.body().toList()));
        }
        try (Stream<String> lines = r.body()) {    // one JSON object per line
            lines.filter(l -> !l.isBlank())
                 .forEach(l -> onChunk.accept(MiniJson.parseObject(l)));
        } catch (java.io.UncheckedIOException e) {
            throw new NoModelAvailable("stream broke off (" + e.getCause() + ")");
        }
    }

    public static void main(String[] args) {
        String host = DEFAULT_HOST, model = "qwen2.5:7b";
        boolean stream = false;
        List<String> words = new ArrayList<>();
        for (int i = 0; i < args.length; i++) {
            switch (args[i]) {
                case "--host" -> host = args[++i];
                case "--model" -> model = args[++i];
                case "--stream" -> stream = true;
                default -> words.add(args[i]);
            }
```

Kode 6.5 (Lanjutan) *OllamaClient.java: klien yang sama dengan java.net.http.HttpClient; streaming memakai BodyHandlers.ofLines().*

```
        }
        String prompt = words.isEmpty() ? "Say hello in one short sentence."
                                        : String.join(" ", words);
        OllamaClient client = new OllamaClient(host, 60.0);
        Map<String, Object> fin = new LinkedHashMap<>();
        try {
            List<String> names = client.models();
            System.out.println("models on server: "
                    + (names.isEmpty() ? "(none)" : String.join(", ", names)));
            if (stream) {
                client.stream(model, prompt, null, chunk -> {
                    System.out.print(chunk.get("response"));
                    System.out.flush();
                    if (Boolean.TRUE.equals(chunk.get("done"))) fin.putAll(chunk);
                });
                System.out.println();
            } else {
                fin.putAll(client.generate(model, prompt, null, null));
                System.out.println(fin.get("response"));
            }
        } catch (NoModelAvailable e) {
            System.out.println("No local model available: " + e.getMessage());
            System.exit(2);
        }
        long n = ((Number) fin.get("eval_count")).longValue();
        double secs = ((Number) fin.get("eval_duration")).doubleValue() / 1e9;
        System.out.printf(Locale.ROOT, "[%s: eval_count=%d, %.1f tokens/s as"
                + " reported by the server]%n", fin.get("model"), n, n / secs);
    }
}
```

Di Java, `NoModelAvailable` merupakan turunan `Exception`, jadi ia eksepsi *checked*: kompilator menolak mengompilasi pemanggil yang tidak menangkap maupun mendeklarasikannya. Itulah bahasa pemrograman yang menegakkan Subbab 6.7 untuk kita. `HttpResponse.BodyHandlers.ofLines()` menyerahkan badan jawaban yang di-*stream* sebagai `Stream<String>` berisi baris-baris, yang diurai klien satu per satu lalu diserahkan ke sebuah `Consumer`, padanan Java untuk *generator* Python. `Locale.ROOT` pada `printf` terakhir menjaga titik desimal tetap titik di komputer yang pengaturan bahasanya akan mencetak `20,0`.

Sekarang jalankan klien terhadap *stub*. Dari `Chapter06/python` (atau dengan `java OllamaClient` dan argumen yang sama dari `Chapter06/java`):

```
python ollama_client.py --host http://127.0.0.1:7060 --model stub-fast "CLASSIFY: my
shirt came back torn"
python ollama_client.py --host http://127.0.0.1:7060 --model stub-quality --stream "Tell
me about the laundry"
```

```
models on server: stub-fast, stub-quality
complaint
[stub-fast: eval_count=1, 125.0 tokens/s as reported by the server]
models on server: stub-fast, stub-quality
The business did not give me any facts to write about.
[stub-quality: eval_count=11, 20.0 tokens/s as reported by the server]
```

"Model" cepat melabeli pesan itu sebagai keluhan (*complaint*); "model" kualitas, yang tidak diberi fakta apa pun, mengatakannya terus terang alih-alih mengarang. Lajunya, 125 dan 20 token per detik, adalah laju simulasi *stub*, dihitung dengan Persamaan (6.5) dari kolom-kolom yang dikembalikannya.

Kini tiga cara untuk tidak memiliki model. Pertama, minta model yang tidak dimiliki server (bawaan klien adalah `qwen2.5:7b`); kedua, arahkan klien ke porta yang tidak didengarkan siapa pun; ketiga, jalankan ulang *stub* dengan `python stub_ollama.py --no-models` lalu minta lagi:

```
python ollama_client.py --host http://127.0.0.1:7060 "Say hello"
python ollama_client.py --host http://127.0.0.1:7069 "Say hello"
python ollama_client.py --host http://127.0.0.1:7060 --model stub-quality "Say hello"
```

```
models on server: stub-fast, stub-quality
No local model available: HTTP 404: model 'qwen2.5:7b' not found
No local model available: cannot reach http://127.0.0.1:7069 (connection refused)
models on server: (none)
No local model available: HTTP 404: model 'stub-quality' not found
```

Setiap kegagalan menghasilkan kalimat yang jelas dan kode keluar 2, tidak pernah jawaban karangan. Klien Java mencetak baris-baris yang sama; kami membandingkan keluaran kedua bahasa dengan `diff`, terhadap kedua *stub*, dan hasilnya identik.

6.8.6 Dua jalur dalam kode

Kode 6.6 mengubah **Gambar 6.2** menjadi program. Sebuah `Lane` mencatat nama, model, batas waktu, dan jumlah token maksimum. Jalur kualitas mendapat 120 detik dan paling banyak 400 token; jalur cepat mendapat 3 detik dan 8 token. Ini adalah nilai milik lab ini sendiri, dipilih untuk laptop; sesuaikan dengan perangkat keras Anda.

Kode 6.6 `two_lanes.py`: jalur kualitas yang menjalankan satu pekerjaan panjang sekaligus dan melaporkan posisi antrean, jalur cepat untuk label, dan lantai aturan di bawahnya.

```
"""Two lanes for local AI: a quality lane for long drafts, a fast lane for
short labels. Start stub_ollama.py with --sleep-ms 100 first.

    python two_lanes.py [--host URL]
"""
import re
import sys
import threading
from dataclasses import dataclass

from ollama_client import NoModelAvailable, OllamaClient


@dataclass
class Lane:
    name: str
    model: str
    timeout_s: float
    max_tokens: int


QUALITY = Lane("quality", "stub-quality", 120.0, 400)
FAST = Lane("fast", "stub-fast", 3.0, 8)


def rule_floor(message):
    """Deterministic labels that need no model at all."""
    t = message.lower()
    if re.search(r"\b(late|lost|broken|torn|refund)\b", t):
        return "complaint"
    return "question" if "?" in t else "other"


class Router:
    def __init__(self, host):
        self.host = host
        self.quality_lock = threading.Lock()  # one long job at a time
        self.waiting = 0
        self.log_lock = threading.Lock()

    def log(self, text):
        with self.log_lock:
            print(text, flush=True)
```

Kode 6.6 (Lanjutan) *two_lanes.py: jalur kualitas yang menjalankan satu pekerjaan panjang sekaligus dan melaporkan posisi antrean, jalur cepat untuk label, dan lantai aturan di bawahnya.*

```
    def classify(self, message, lane=FAST):
        client = OllamaClient(self.host, lane.timeout_s)
        try:
            out = client.generate(lane.model, "CLASSIFY: " + message,
                                  {"num_predict": lane.max_tokens})
            return out["response"].strip(), f"{lane.name} lane", ""
        except NoModelAvailable as e:
            return rule_floor(message), "rule floor", str(e)

    def draft(self, job, prompt):
        with self.log_lock:
            self.waiting += 1
            position = self.waiting - 1
        if position:
            self.log(f"  {job}: queued, position {position} in quality lane")
        with self.quality_lock:
            self.log(f"  {job}: drafting on {QUALITY.model} ...")
            client = OllamaClient(self.host, QUALITY.timeout_s)
            try:
                text = "".join(c["response"] for c in client.stream(
                    QUALITY.model, prompt, {"num_predict": QUALITY.max_tokens}))
                self.log(f"  {job}: done, {len(text.split())} words: "
                         f"{text[:34]}...")
            except NoModelAvailable as e:
                self.log(f"  {job}: no local model available ({e})")
            finally:
                with self.log_lock:
                    self.waiting -= 1


PROMPT = """Business: Profitina Laundry
- a laundry service in Surabaya
- open every day from 07:00 to 21:00
- express service is ready in 6 hours"""

MESSAGES = ["Is express really ready in 6 hours?",
            "My order came back late and one shirt was torn",
            "Thanks, the ironing was great"]
```

Kode 6.6 (Lanjutan) *two_lanes.py: jalur kualitas yang menjalankan satu pekerjaan panjang sekaligus dan melaporkan posisi antrean, jalur cepat untuk label, dan lantai aturan di bawahnya.*

```
def main(argv):
    host = argv[argv.index("--host") + 1] if "--host" in argv \
        else "http://127.0.0.1:7060"
    router = Router(host)
    print("1. Two long drafts go to the quality lane:")
    jobs = [threading.Thread(target=router.draft, args=(j, PROMPT))
            for j in ("draft A", "draft B")]
    for t in jobs:
        t.start()
        t.join(0.2)  # let A take the lane before B arrives
    print("2. Meanwhile, short labels go to the fast lane:")
    for m in MESSAGES:
        label, source, _ = router.classify(m)
        router.log(f"  {label:<9} <- {m!r} [{source}]")
    for t in jobs:
        t.join()
    print("3. The fast lane loses its model; the rule floor answers:")
    gone = Lane("fast", "stub-missing", 3.0, 8)
    for m in MESSAGES[:2]:
        label, source, why = router.classify(m, gone)
        print(f"  {label:<9} <- {m!r} [{source}]")
    print(f"  (the fast lane said: {why})")


if __name__ == "__main__":
    main(sys.argv[1:])
```

`Router` menegakkan aturan-aturan jalur. `draft` menghitung pekerjaan yang menunggu untuk melaporkan posisi antrean yang jujur, yaitu Persamaan (6.6) dengan k yang nyata, lalu mengambil `quality_lock`, kunci yang hanya mengizinkan satu pekerjaan panjang memakai jalur kualitas pada satu waktu. `classify` tidak pernah menunggu kunci itu: ia langsung menuju jalur cepat dengan batas waktunya sendiri yang pendek, dan bila jalur cepat memunculkan `NoModelAvailable`, `rule_floor` yang deterministik menjawab dan hasilnya menyebut hal itu. `log` mencetak di bawah kunci kedua agar baris dari *thread* yang berbeda tidak pernah bercampur.

`main` mementaskan drama kecil dalam tiga babak. Dua draf dimulai di *thread* latar belakang, berselang 0,2 detik, sehingga draf A mengambil jalur dan draf B harus antre. Selagi draf-draf itu ditulis, tiga pesan pelanggan diberi label di *thread* utama. Terakhir, jalur cepat diarahkan ke model yang tidak ada untuk memperlihatkan lantai aturan.

Kode 6.7 adalah pengarah yang sama dalam Java. `record` menggantikan *dataclass* Python, `ReentrantLock` yang adil menjaga jalur kualitas, dan metode `synchronized` melindungi penghitung dan pencetakan.

Kode 6.7 `TwoLanes.java`: pengarah dua jalur dalam Java, dengan `record`, `ReentrantLock`, dan *thread*.

```
import java.util.List;
import java.util.Map;
import java.util.concurrent.locks.ReentrantLock;
import java.util.regex.Pattern;

/** Two lanes for local AI: a quality lane for long drafts, a fast lane for
 *  short labels. Start StubOllama with --sleep-ms 100 first.
 *  java TwoLanes [--host URL] */
public class TwoLanes {
    record Lane(String name, String model, double timeoutS, int maxTokens) {}

    static final Lane QUALITY = new Lane("quality", "stub-quality", 120.0, 400);
    static final Lane FAST = new Lane("fast", "stub-fast", 3.0, 8);

    record Label(String label, String source, String why) {}

    /** Deterministic labels that need no model at all. */
    static String ruleFloor(String message) {
        String t = message.toLowerCase();
        if (Pattern.compile("\\b(late|lost|broken|torn|refund)\\b")
                .matcher(t).find()) return "complaint";
        return t.contains("?") ? "question" : "other";
    }

    private final String host;
    private final ReentrantLock qualityLock = new ReentrantLock(true);
    private int waiting = 0;

    TwoLanes(String host) { this.host = host; }

    synchronized void log(String text) { System.out.println(text); }

    Label classify(String message, Lane lane) {
        OllamaClient client = new OllamaClient(host, lane.timeoutS());
        try {
            Map<String, Object> out = client.generate(lane.model(),
                    "CLASSIFY: " + message, Map.of("num_predict", lane.maxTokens()),
                    null);
            return new Label(out.get("response").toString().strip(),
                             lane.name() + " lane", "");
        } catch (OllamaClient.NoModelAvailable e) {
            return new Label(ruleFloor(message), "rule floor", e.getMessage());
        }
    }
```

Kode 6.7 (Lanjutan) *TwoLanes.java: pengarah dua jalur dalam Java, dengan record, ReentrantLock, dan thread.*

```
    void draft(String job, String prompt) {
        int position;
        synchronized (this) { position = waiting++; }
        if (position > 0) {
            log("  " + job + ": queued, position " + position + " in quality lane");
        }
        qualityLock.lock();                          // one long job at a time
        try {
            log("  " + job + ": drafting on " + QUALITY.model() + " ...");
            OllamaClient client = new OllamaClient(host, QUALITY.timeoutS());
            StringBuilder text = new StringBuilder();
            client.stream(QUALITY.model(), prompt,
                          Map.of("num_predict", QUALITY.maxTokens()),
                          c -> text.append(c.get("response")));
            String t = text.toString();
            log("  " + job + ": done, " + t.split("\\s+").length + " words: "
                + t.substring(0, Math.min(34, t.length())) + "...");
        } catch (OllamaClient.NoModelAvailable e) {
            log("  " + job + ": no local model available (" + e.getMessage() + ")");
        } finally {
            synchronized (this) { waiting--; }
            qualityLock.unlock();
        }
    }

    static final String PROMPT = """
            Business: Profitina Laundry
            - a laundry service in Surabaya
            - open every day from 07:00 to 21:00
            - express service is ready in 6 hours""";

    static final List<String> MESSAGES = List.of(
            "Is express really ready in 6 hours?",
            "My order came back late and one shirt was torn",
            "Thanks, the ironing was great");

    static String show(String label, String message, String source) {
        return String.format("  %-9s <- '%s' [%s]", label, message, source);
    }
```

Kode 6.7 (Lanjutan) *TwoLanes.java: pengarah dua jalur dalam Java, dengan record, ReentrantLock, dan thread.*

```
    public static void main(String[] args) throws InterruptedException {
        List<String> a = List.of(args);
        String host = a.contains("--host") ? a.get(a.indexOf("--host") + 1)
                                           : "http://127.0.0.1:7060";
        TwoLanes router = new TwoLanes(host);
        System.out.println("1. Two long drafts go to the quality lane:");
        Thread[] jobs = new Thread[2];
        String[] names = {"draft A", "draft B"};
        for (int k = 0; k < 2; k++) {
            String job = names[k];
            jobs[k] = new Thread(() -> router.draft(job, PROMPT));
            jobs[k].start();
            jobs[k].join(200);                    // let A take the lane first
        }
        System.out.println("2. Meanwhile, short labels go to the fast lane:");
        for (String m : MESSAGES) {
            Label r = router.classify(m, FAST);
            router.log(show(r.label(), m, r.source()));
        }
        for (Thread t : jobs) t.join();
        System.out.println(
                "3. The fast lane loses its model; the rule floor answers:");
        Lane gone = new Lane("fast", "stub-missing", 3.0, 8);
        String why = "";
        for (String m : MESSAGES.subList(0, 2)) {
            Label r = router.classify(m, gone);
            System.out.println(show(r.label(), m, r.source()));
            why = r.why();
        }
        System.out.println("  (the fast lane said: " + why + ")");
    }
}
```

Agar antrean terlihat, jalankan ulang *stub* sehingga ia berhenti 100 milidetik untuk setiap token yang di-*stream*, lalu jalankan pengarahnya:

```
python stub_ollama.py --sleep-ms 100
python two_lanes.py
```

```
1. Two long drafts go to the quality lane:
  draft A: drafting on stub-quality ...
  draft B: queued, position 1 in quality lane
2. Meanwhile, short labels go to the fast lane:
  question  <- 'Is express really ready in 6 hours?' [fast lane]
  complaint <- 'My order came back late and one shirt was torn' [fast lane]
  praise    <- 'Thanks, the ironing was great' [fast lane]
  draft A: done, 22 words: Profitina Laundry is a laundry ser...
  draft B: drafting on stub-quality ...
  draft B: done, 22 words: Profitina Laundry is a laundry ser...
3. The fast lane loses its model; the rule floor answers:
  question  <- 'Is express really ready in 6 hours?' [rule floor]
  complaint <- 'My order came back late and one shirt was torn' [rule floor]
  (the fast lane said: HTTP 404: model 'stub-missing' not found)
```

Bacalah keluaran itu sebagai garis waktu. Draf A mulai disusun dan draf B diberi tahu, dengan jujur, bahwa ia berada di urutan pertama antrean. Selagi draf A masih men-*stream* 22 katanya, ketiga label sudah kembali dari jalur cepat, karena tidak perlu menunggu di belakang draf; itulah inti **Gambar 6.2**. Baru setelah draf A selesai, draf B dimulai. Pada babak 3 model jalur cepat tidak ada, lantai aturan melabeli kedua pesan dengan benar, dan program menyebut sumber mana yang menjawab serta mengapa jalur cepat gagal. Versi Java mencetak baris-baris yang persis sama, terhadap *stub* mana pun.

6.8.7 Gerbang kualitas dalam kode

Kode 6.8 menerapkan Persamaan (6.7) untuk Profitina Laundry.

Kode 6.8 `quality_gate.py`: cek wajib, skor lunak, dan perbandingan ketat terhadap templat yang benar.

```
"""A quality gate: serve an AI draft only when it beats a template.

The checklist below is this lab's own, chosen by the reader. Start
stub_ollama.py first (any --sleep-ms).

    python quality_gate.py [--host URL]
"""
import re
import sys
import textwrap

from ollama_client import NoModelAvailable, OllamaClient

NAME = "Profitina Laundry"
FACTS = ["a laundry service in Surabaya",
         "open every day from 07:00 to 21:00",
         "express service is ready in 6 hours"]
BANNED = ["best", "terbaik", "number one", "#1", "guaranteed", "cheapest"]


def template():
    """The safe fallback: correct, complete, and a little stiff."""
    return f"{NAME} - " + "; ".join(FACTS) + "."


def prompt():
    lines = [f"Business: {NAME}"] + [f"- {f}" for f in FACTS]
    return "\n".join(lines)


def hard_checks(text):
    """Every check must pass, or the draft is never served."""
    fails = []
    words = text.split()
    if not 5 <= len(words) <= 120:
        fails.append(f"length {len(words)} words is outside 5..120")
    first = re.split(r"(?<=\.)\s", text, maxsplit=1)[0]
    if NAME not in first:
        fails.append("the first sentence does not name the business")
    low = text.lower()
    for b in BANNED:
        if re.search(r"(?<!\w)" + re.escape(b) + r"(?!\w)", low):
            fails.append(f"banned phrase {b!r}")
    known = set(re.findall(r"\d+", " ".join(FACTS)))
    for n in re.findall(r"\d+", text):
        if n not in known:
```

Kode 6.8 (Lanjutan) *quality_gate.py: cek wajib, skor lunak, dan perbandingan ketat terhadap templat yang benar.*

```
            fails.append(f"number {n} is not in the facts")
    return fails


def score(text):
    """Soft score: facts covered (0..3) + 1 if it opens with 'NAME is'."""
    low = text.lower()
    covered = sum(all(w in low for w in f.lower().split()[-3:])
                  for f in FACTS)
    return covered + (1 if text.startswith(NAME + " is ") else 0)


def show(label, text):
    print(textwrap.fill(text, 78, initial_indent=f"  {label:<9}: ",
                        subsequent_indent=" " * 13, break_on_hyphens=False))


def gate(client, model, seed):
    base = template()
    try:
        ai = client.generate(model, prompt(), {"seed": seed})["response"]
    except NoModelAvailable as e:
        return base, f"template (no local model available: {e})", False
    fails = hard_checks(ai)
    show("AI draft", ai)
    if fails:
        return base, "template (AI draft failed: " + "; ".join(fails) + ")", \
            False
    s_ai, s_t = score(ai), score(base)
    show("scores", f"AI {s_ai}, template {s_t}")
    if s_ai > s_t:
        return ai, "AI draft (it beats the template)", True
    return base, "template (AI draft did not beat it)", False


def main(argv):
    host = argv[argv.index("--host") + 1] if "--host" in argv \
        else "http://127.0.0.1:7060"
    client = OllamaClient(host, timeout=60)
    wins = 0
    cases = [("stub-quality", 0), ("stub-quality", 1), ("stub-quality", 2),
             ("stub-missing", 0)]
    for i, (model, seed) in enumerate(cases, 1):
        print(f"Case {i}: model={model}, seed={seed}")
        text, why, ai_won = gate(client, model, seed)
        wins += ai_won
```

Kode 6.8 (Lanjutan) *quality_gate.py: cek wajib, skor lunak, dan perbandingan ketat terhadap templat yang benar.*

```
        show("decision", why)
        show("SERVED", text)
    print(f"AI drafts served in {wins} of {len(cases)} cases.")


if __name__ == "__main__":
    main(sys.argv[1:])
```

`FACTS` memuat tiga pernyataan benar tentang binatu itu; hanya itulah sumber kebenaran. `template` membangun teks yang benar, lengkap, dan kaku darinya, dan `prompt` menyerahkan fakta yang sama kepada model. `hard_checks` menerapkan $H(d)$ dari Persamaan (6.7) sebagai daftar alasan, sehingga daftar kosong berarti "lolos" dan daftar yang berisi menjelaskan

kegagalannya. Aturan paling menarik adalah yang terakhir: setiap angka di dalam draf harus muncul di suatu tempat di dalam fakta, dan itu menangkap harga, tahun, dan nomor telepon karangan dengan satu ekspresi reguler. `score` menerapkan *S*: satu poin untuk setiap fakta yang tiga kata terakhirnya muncul semua di dalam teks, ditambah satu poin bila teks dibuka dengan "Profitina Laundry is", bentuk *answer-first*. `gate` menggabungkan keduanya persis seperti yang digambar **Gambar 6.3**, termasuk cabang jujur ketika tidak ada model yang tersedia. **Kode 6.9** adalah versi Java; metode `show` di dalamnya meniru pemenggalan baris `textwrap.fill` milik Python, sehingga kedua program mencetak baris-baris yang identik.

Kode 6.9 `QualityGate.java`: gerbang yang sama dalam Java; `show` meniru `textwrap.fill` Python sehingga keduanya mencetak baris yang identik.

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** A quality gate: serve an AI draft only when it beats a template.
 *  The checklist is this lab's own. Start StubOllama first.
 *  java QualityGate [--host URL] */
public class QualityGate {
    static final String NAME = "Profitina Laundry";
    static final List<String> FACTS = List.of(
            "a laundry service in Surabaya",
            "open every day from 07:00 to 21:00",
            "express service is ready in 6 hours");
    static final List<String> BANNED = List.of(
            "best", "terbaik", "number one", "#1", "guaranteed", "cheapest");

    record Outcome(String text, String why, boolean aiWon) {}

    /** The safe fallback: correct, complete, and a little stiff. */
    static String template() {
        return NAME + " - " + String.join("; ", FACTS) + ".";
    }

    static String prompt() {
        StringBuilder b = new StringBuilder("Business: " + NAME);
        for (String f : FACTS) b.append("\n- ").append(f);
        return b.toString();
    }

    static List<String> numbers(String text) {
        List<String> out = new ArrayList<>();
        Matcher m = Pattern.compile("\\d+").matcher(text);
        while (m.find()) out.add(m.group());
        return out;
    }
```

Kode 6.9 (Lanjutan) *QualityGate.java: gerbang yang sama dalam Java; show meniru textwrap.fill Python sehingga keduanya mencetak baris yang identik.*

```
    /** Every check must pass, or the draft is never served. */
    static List<String> hardChecks(String text) {
        List<String> fails = new ArrayList<>();
        int words = text.strip().split("\\s+").length;
        if (words < 5 || words > 120) {
            fails.add("length " + words + " words is outside 5..120");
        }
        String first = text.split("(?<=\\.)\\s", 2)[0];
        if (!first.contains(NAME)) {
            fails.add("the first sentence does not name the business");
        }
        String low = text.toLowerCase();
        for (String b : BANNED) {
            if (Pattern.compile("(?<!\\w)" + Pattern.quote(b) + "(?!\\w)")
                       .matcher(low).find()) {
                fails.add("banned phrase '" + b + "'");
            }
        }
        Set<String> known = new HashSet<>(numbers(String.join(" ", FACTS)));
        for (String n : numbers(text)) {
            if (!known.contains(n)) {
                fails.add("number " + n + " is not in the facts");
            }
        }
        return fails;
    }

    /** Soft score: facts covered (0..3) + 1 if it opens with 'NAME is'. */
    static int score(String text) {
        String low = text.toLowerCase();
        int covered = 0;
        for (String f : FACTS) {
            String[] w = f.toLowerCase().split(" ");
            boolean all = true;
            for (int k = Math.max(0, w.length - 3); k < w.length; k++) {
                all &= low.contains(w[k]);
            }
            if (all) covered++;
        }
        return covered + (text.startsWith(NAME + " is ") ? 1 : 0);
    }
```

Kode 6.9 (Lanjutan) *QualityGate.java: gerbang yang sama dalam Java; show meniru textwrap.fill Python sehingga keduanya mencetak baris yang identik.*

```
    /** Greedy word wrap, the same as Python's textwrap.fill here. */
    static void show(String label, String text) {
        String indent = String.format("  %-9s: ", label);
        StringBuilder line = new StringBuilder(indent);
        boolean empty = true;
        for (String w : text.split(" ")) {
            if (!empty && line.length() + 1 + w.length() > 78) {
                System.out.println(line);
                line = new StringBuilder(" ".repeat(13));
                empty = true;
            }
            if (!empty) line.append(' ');
            line.append(w);
            empty = false;
        }
        System.out.println(line);
    }

    static Outcome gate(OllamaClient client, String model, long seed) {
        String base = template();
        String ai;
        try {
            ai = client.generate(model, prompt(), Map.of("seed", seed), null)
                       .get("response").toString();
        } catch (OllamaClient.NoModelAvailable e) {
            return new Outcome(base, "template (no local model available: "
                               + e.getMessage() + ")", false);
        }
        List<String> fails = hardChecks(ai);
        show("AI draft", ai);
        if (!fails.isEmpty()) {
            return new Outcome(base, "template (AI draft failed: "
                               + String.join("; ", fails) + ")", false);
        }
        int sAi = score(ai), sT = score(base);
        show("scores", "AI " + sAi + ", template " + sT);
        if (sAi > sT) {
            return new Outcome(ai, "AI draft (it beats the template)", true);
        }
        return new Outcome(base, "template (AI draft did not beat it)", false);
    }
```

Kode 6.9 (Lanjutan) *QualityGate.java: gerbang yang sama dalam Java; show meniru textwrap.fill Python sehingga keduanya mencetak baris yang identik.*

```
    public static void main(String[] args) {
        List<String> a = List.of(args);
        String host = a.contains("--host") ? a.get(a.indexOf("--host") + 1)
                                           : "http://127.0.0.1:7060";
        OllamaClient client = new OllamaClient(host, 60.0);
        String[] models = {"stub-quality", "stub-quality", "stub-quality",
                           "stub-missing"};
        long[] seeds = {0, 1, 2, 0};
        int wins = 0;
        for (int i = 0; i < models.length; i++) {
            System.out.println("Case " + (i + 1) + ": model=" + models[i]
                               + ", seed=" + seeds[i]);
            Outcome o = gate(client, models[i], seeds[i]);
            if (o.aiWon()) wins++;
            show("decision", o.why());
            show("SERVED", o.text());
        }
        System.out.println("AI drafts served in " + wins + " of " + models.length
                           + " cases.");
    }
}
```

Dengan *stub* berjalan (pada kecepatan berapa pun), jalankan gerbangnya:

```
python quality_gate.py
```

```
Case 1: model=stub-quality, seed=0
  AI draft : Profitina Laundry is a laundry service in Surabaya. Open every
             day from 07:00 to 21:00. Express service is ready in 6 hours.
  scores   : AI 4, template 3
  decision : AI draft (it beats the template)
  SERVED   : Profitina Laundry is a laundry service in Surabaya. Open every
             day from 07:00 to 21:00. Express service is ready in 6 hours.
Case 2: model=stub-quality, seed=1
  AI draft : Profitina Laundry is the best laundry in town, trusted since
             1990. It is a laundry service in Surabaya.
  decision : template (AI draft failed: banned phrase 'best'; number 1990 is
             not in the facts)
  SERVED   : Profitina Laundry - a laundry service in Surabaya; open every day
             from 07:00 to 21:00; express service is ready in 6 hours.
Case 3: model=stub-quality, seed=2
  AI draft : Profitina Laundry: a laundry service in Surabaya.
  scores   : AI 1, template 3
  decision : template (AI draft did not beat it)
  SERVED   : Profitina Laundry - a laundry service in Surabaya; open every day
             from 07:00 to 21:00; express service is ready in 6 hours.
Case 4: model=stub-missing, seed=0
  decision : template (no local model available: HTTP 404: model
             'stub-missing' not found)
  SERVED   : Profitina Laundry - a laundry service in Surabaya; open every day
             from 07:00 to 21:00; express service is ready in 6 hours.
AI drafts served in 1 of 4 cases.
```

Kasus 1 adalah draf yang kita harapkan: ia lolos semua cek wajib, mencakup ketiga fakta, dan dibuka dalam bentuk *answer-first*, sehingga skornya 4 melawan 3 milik templat dan draf itu disajikan. Kasus 2 akan memalukan bila diterbitkan; gerbang menyebut kedua kesalahannya, kata superlatif dan tahun karangan, lalu menyajikan templat. Kasus 3 tidak melanggar aturan apa pun tetapi hanya mencakup satu fakta, sehingga kalah pada skor lunak. Kasus 4 sama sekali tidak memiliki model, dan baris keputusannya menyatakannya dengan kata-kata sederhana sementara templat disajikan. Satu dari empat draf AI disajikan, dan itulah jawaban yang tepat untuk keempat draf ini. Ini juga pengingat bahwa angka publik 5 dari 6 di Subbab 6.6.3 menggambarkan mesin

Profitina yang sudah direvisi pada tolok ukurnya sendiri, bukan apa yang akan dilakukan model mana pun di mana pun.

Terakhir, jalankan ulang *stub* dengan `--no-models` dan jalankan gerbangnya lagi. Kini setiap kasus melaporkan bahwa tidak ada model lokal yang tersedia dan menyajikan templat yang diberi label:

```
Case 1: model=stub-quality, seed=0
  decision : template (no local model available: HTTP 404: model
             'stub-quality' not found)
  SERVED   : Profitina Laundry - a laundry service in Surabaya; open every day
             from 07:00 to 21:00; express service is ready in 6 hours.
Case 2: model=stub-quality, seed=1
  decision : template (no local model available: HTTP 404: model
             'stub-quality' not found)
  SERVED   : Profitina Laundry - a laundry service in Surabaya; open every day
             from 07:00 to 21:00; express service is ready in 6 hours.
Case 3: model=stub-quality, seed=2
  decision : template (no local model available: HTTP 404: model
             'stub-quality' not found)
  SERVED   : Profitina Laundry - a laundry service in Surabaya; open every day
             from 07:00 to 21:00; express service is ready in 6 hours.
Case 4: model=stub-missing, seed=0
  decision : template (no local model available: HTTP 404: model
             'stub-missing' not found)
  SERVED   : Profitina Laundry - a laundry service in Surabaya; open every day
             from 07:00 to 21:00; express service is ready in 6 hours.
AI drafts served in 0 of 4 cases.
```

6.8.8 Kode yang sama terhadap Ollama sungguhan

Untuk melihat keluaran model sungguhan di komputer Anda sendiri, pasang Ollama dari situs resminya atau dari halaman rilis GitHub-nya [10], lalu unduh satu model kualitas dan satu model cepat, dan periksa bahwa server menjawab:

```
ollama pull qwen2.5:7b
ollama pull qwen3:1.7b
ollama list
python ollama_client.py --model qwen3:1.7b "Say hello in one short sentence."
```

Tanpa `--host`, klien berbicara dengan `http://127.0.0.1:11434`, yaitu Ollama sungguhan. Untuk menjalankan program jalur dan gerbang terhadapnya, ganti nama model di `QUALITY`, `FAST`, dan `cases` dari nama `stub-…` menjadi model yang Anda unduh, ganti konvensi `CLASSIFY:` dengan instruksi yang jelas seperti “Answer with one word: question, complaint, praise or other”, dan berikan `--host http://127.0.0.1:11434`. Harapkan tiga perbedaan dari *stub*. Teksnya akan ditulis model, sehingga akan berubah-ubah menurut *seed* dan *temperature* dan sesekali gagal cek wajib Anda, dan justru untuk itulah gerbang ada. Durasinya akan nyata, sehingga laju dari Persamaan (6.5) akan bercerita tentang perangkat keras Anda. Dan permintaan pertama setelah jeda akan memuat `load_duration` yang cukup terasa, karena Ollama memuat model ke memori pada pemakaian pertama dan melepaskannya setelah lama tidak dipakai. Sekalian bandingkan `127.0.0.1` dengan `localhost`: Soal 6.4 memintanya.

6.9 Profitina dalam Praktik

Semua yang dibahas di bab ini memiliki padanan di produk yang sedang berjalan, yang di sini diuraikan pada tingkat kotak dan panah.

Hanya lokal. Setiap fitur AI di app.profitina.com berjalan di Ollama dengan model Qwen pada server milik platform sendiri. Jalur kode AI *hosted* sudah dihapus seluruhnya; data pengguna tidak pernah meninggalkan server untuk diproses AI, dan pengguna tidak diminta memilih model [6]. ChatGPT, Gemini, dan Perplexity hanya muncul di Profitina sebagai mesin jawaban yang kesiapannya untuk mengutip sebuah situs web diperkirakan; mesin-mesin itu tidak pernah dipanggil untuk menjalankan fitur apa pun. (Profitina AI, aplikasi AI *local-first* yang terpisah dalam grup yang sama, dapat mengarahkan pekerjaan antara model lokal dan model *cloud* untuk penggunanya sendiri; aplikasi itu bukan bagian dari app.profitina.com, dan platform GEO tidak memakai AI *cloud* sama sekali.)

Dua jalur. Jalur kualitas menjalankan Qwen2.5 7B di CPU untuk pekerjaan berbentuk panjang seperti draf Content Engine; jalur cepat menjalankan qwen3:1.7b di GPU untuk klasifikasi pendek: kategori pindai, kebermanfaatan Fair-Play, dan opini opsional Link Safety [6]. Pada rilis September 2026, opini model untuk Link Safety hanya berjalan di jalur cepat, dan jalur baca umpan komunitas sama sekali tidak memanggil model (Subbab 6.5.4).

Content Engine dan gerbangnya. Draf, FAQ, dan JSON-LD berasal dari jalur kualitas, dan gerbang kualitas hanya menyajikan draf AI bila mengalahkan templat bawaan; setelah revisi September 2026, draf AI menang pada 5 dari 6 kasus tolok ukur. Ketika tidak ada model yang tersedia, aplikasi mengatakannya [6].

Kemajuan dan antrean yang jujur. Operasi panjang menampilkan kemajuan 0–100% dengan label sub-langkah seperti "drafting — 466 tokens" dan perkiraan sisa waktu. *One Text, Many Platforms* memproses satu permintaan sekaligus di seluruh sistem dan menampilkan posisi antrean serta perkiraan waktu tunggu. Apa yang tidak dijanjikannya pun dinyatakan: hasil yang "lebih menarik" atau "viral" tidak dijanjikan [6].

Operasional. *Endpoint* kesehatan platform, `/api/health`, melaporkan keadaan AI lokal di samping basis data, disk, dan lama menyala, dan Command Center admin menampilkan kesehatan yang sama sekilas pandang (Bab 9). Aturan `127.0.0.1` di Subbab 6.4.1 tertulis dalam aturan kerja proyek karena hukuman 2,25 detik yang terukur itu.

Apa yang bukan lab bab ini. *Stub*, batas waktu dan batas token jalur-jalur, cek wajib, dan skor lunak di Subbab 6.8 adalah pilihan pengajaran buku ini sendiri untuk Profitina Laundry. Semuanya memperlihatkan gagasannya; produk memiliki *prompt*, templat, dan gerbangnya sendiri, yang disetel pada tolok ukurnya sendiri.

6.10 Kesalahpahaman yang Umum

"AI lokal berarti AI luring bermutu rendah, jadi hanya untuk penghobi." Lokal menunjuk *di mana* model berjalan, bukan seberapa baik ia. Model 7B menulis paragraf faktual yang jernih; batas atas mutunya nyata, tetapi dikelola oleh gerbang yang hanya menyajikan hasil kerja model bila lebih baik daripada templat yang benar.

"Jika data tidak pernah meninggalkan server, urusan privasi selesai." Inferensi lokal menghilangkan satu risiko besar, yaitu menyalin data ke pihak ketiga. Ia tidak menggantikan kendali akses, batas retensi, persetujuan, atau hak untuk berkeberatan atas keputusan otomatis (Bab 11). Ia adalah satu langkah privasi di antara beberapa.

"Model yang lebih besar atau lebih baru selalu merupakan peningkatan." Model yang lebih besar lebih lambat dan memerlukan lebih banyak memori menurut Persamaan (6.1); keluarga yang lebih baru mungkin memiliki kekuatan yang berbeda. Sebuah model merupakan

peningkatan bila ia menang pada tolok ukur Anda, di perangkat keras Anda, pada kecepatan yang dapat Anda terima.

"Satu model dan satu antrean lebih sederhana, jadi lebih baik." Lebih sederhana untuk digambar, lebih buruk untuk dipakai: menurut Persamaan (6.4), label yang hanya memerlukan seperdua puluh detik dapat menunggu draf yang memerlukan satu menit. Dua jalur memisahkan pekerjaan menurut panjang dan urgensinya.

"Jika AI mati, pakai templat diam-diam; pengguna tidak akan sadar." Pengguna akan menyadari mutunya dan menyalahkan hal yang salah, dan operator tidak akan menyadari gangguannya. Sajikan templat bila memang perlu, tetapi katakan bahwa itu templat.

"5 dari 6 berarti 83,3% draf adalah draf AI." Artinya mesin yang direvisi mengalahkan templat pada lima dari enam kasus tolok ukur. Setiap permintaan tetap melewati gerbang, dan enam kasus terlalu sedikit untuk persentase yang presisi; satu kasus menggeser angka itu sebesar 16,7 poin.

"`localhost` dan `127.0.0.1` identik, jadi tidak penting mana yang saya tulis." Keduanya menuju mesin yang sama, tetapi penerjemahan nama dapat mencoba IPv6 lebih dahulu. Di Windows hal itu membuat Profitina kehilangan lebih dari dua detik per panggilan sampai alamatnya ditulis secara eksplisit.

"Memanggil model saat menampilkan halaman tidak apa-apa jika modelnya cepat." Penampilan terjadi untuk setiap butir pada setiap tampilan; biaya kecil per butir menjadi biaya besar per halaman, dan model yang sesekali lambat membuat seluruh halaman sesekali lambat. Nilailah ketika konten dibuat; bacalah putusan yang tersimpan ketika konten ditampilkan.

6.11 Ringkasan Bab

- Model bahasa menebak token berikutnya dari sebuah teks, berulang-ulang. Pengetahuannya membeku saat pelatihan, waktu jalannya bertambah seiring panjang jawaban, dan keluarannya mengandung unsur peluang, sehingga setiap jawaban harus diperiksa.
- Model dengan N parameter yang disimpan dalam b bit memerlukan sekitar $Nb/8$ bita (Persamaan (6.1)); kuantisasi ke 4–5 bit membuat model 7B muat dalam sekitar 4,7 GB.
- Profitina menjalankan seluruh AI secara lokal demi privasi (tidak ada data merek atau data pribadi yang disalin ke pihak ketiga, sejalan dengan UU PDP), biaya (biaya tetap alih-alih argometer per permintaan, Persamaan (6.2) dan Persamaan (6.3)), dan kendali (model yang dipatok dan hanya berganti bila pengukuran menyatakannya). Harganya adalah batas atas mutu, pekerjaan perangkat keras, kecepatan yang lebih rendah, dan antrean.
- Ollama melayani model lokal melalui HTTP di porta 11434; `POST /api/generate` menerima model dan *prompt* lalu mengembalikan teks beserta jumlah token dan durasi dalam nanodetik. Rilis stabil terbaru pada 24 September 2026 adalah v0.34.3. Panggillah di `127.0.0.1`.
- Keluarga Qwen menawarkan model berbobot terbuka dari Qwen2.5 (2024), Qwen3 (2025), hingga Qwen3.5, 3.6, dan 3.8 (2026). Jalur kualitas Profitina menjalankan Qwen2.5 7B dan jalur cepatnya qwen3:1.7b.
- Model dua jalur memisahkan draf panjang yang diperiksa dengan cermat dari label pendek yang mendesak; lantai aturan menjawab bila jalur cepat tidak mampu, dan jalur baca tidak pernah memanggil model.

- Gerbang kualitas hanya menyajikan draf AI jika lolos semua cek wajib dan skornya benar-benar lebih tinggi daripada templat yang benar (Persamaan (6.7)). Hasil publik Profitina adalah 5 kemenangan dari 6 kasus tolok ukur.
- Ketika tidak ada model yang tersedia, perangkat lunak mengatakannya: kegagalan bertipe, alasan yang dapat dibaca, dan cadangan yang diberi label.
- Lab membangun *stub* antarmuka Ollama, sebuah klien, pengarah dua jalur, dan gerbang kualitas dalam Python dan Java; semua keluaran yang dicetak di bab ini berasal dari *stub*.

6.12 Soal Latihan

Soal 6.1 (memori). Gunakan Persamaan (6.1) untuk memperkirakan memori yang diperlukan bobot model 27 miliar parameter (a) pada 16 bit per parameter dan (b) pada 4,5 bit per parameter. Ollama mencantumkan `qwen3.5:27b` sebagai unduhan 17 GB [15]. Berikan dua alasan mengapa ukuran yang tercantum dapat lebih besar daripada perkiraan (b) Anda.

Soal 6.2 (titik impas). Sebuah perusahaan kecil membayar $F = \text{Rp}24{,}000{,}000$ per tahun untuk mesin yang akan menjalankan model lokal, mengeluarkan $e = \text{Rp}4$ untuk listrik per permintaan, dan ditawari model *hosted* seharga $p = \text{Rp}40$ per permintaan. Angka-angka ini dikarang untuk latihan. (a) Tentukan volume titik impas n^* per tahun dan per hari. (b) Harga layanan *hosted* turun menjadi Rp20. Berapa n^* yang baru? (c) Sebutkan satu biaya yang diabaikan Persamaan (6.2) di masing-masing sisi.

Soal 6.3 (latensi dan antrean yang jujur). Sebuah jalur kualitas menghasilkan $r = 6$ token per detik. Panjang draf sekitar 300 token, model sudah dimuat, dan membaca *prompt* memakan sekitar 2 detik. (a) Perkirakan waktu satu draf dengan Persamaan (6.4). (b) Ada dua draf di depan seorang pengguna dalam antrean. Perkirakan waktu tunggu sebelum draf pengguna itu dimulai, dengan Persamaan (6.6), dan total waktu sampai draf itu selesai. (c) Tuliskan satu kalimat pesan yang seharusnya dilihat pengguna.

Soal 6.4 (127.0.0.1 lawan localhost). Pasang Ollama di komputer Anda sendiri dan unduh model kecil. Ukur *time to first token* (waktu sejak permintaan *streaming* dikirim sampai potongan pertama tiba) sepuluh kali dengan host `http://127.0.0.1:11434` dan sepuluh kali dengan `http://localhost:11434`, lalu laporkan mediannya. Jelaskan hasil Anda.

Soal 6.5 (menjalankan gerbang dengan tangan). Dengan cek wajib dan skor lunak dari **Kode 6.8**, tentukan apa yang disajikan gerbang untuk setiap draf berikut, beserta alasannya. Skor templat adalah 3.

(a) “Profitina Laundry is the cheapest laundry service in Surabaya, open every day from 07:00 to 21:00.”

(b) “Open every day from 07:00 to 21:00, Profitina Laundry is a laundry service in Surabaya.”

(c) “Profitina Laundry is a laundry service in Surabaya. Express service is ready in 6 hours, from Rp12,000 per kilogram.”

(d) “Profitina Laundry is a laundry service in Surabaya, open every day from 07:00 to 21:00.”

Soal 6.6 (keluaran terstruktur). Kolom `format` Ollama dapat meminta JSON. Tulislah program, dalam Python dan Java, yang meminta jalur cepat memberikan `{"label": …}` untuk sebuah pesan pelanggan, menerima jawaban hanya bila berupa JSON yang sah dengan label `question`, `complaint`, atau `praise`, mencoba ulang sekali dengan *seed* yang berbeda, dan jika tetap gagal melaporkan dengan jujur bahwa tidak diperoleh label yang sah sehingga manusia yang harus

memutuskan. Jalankan terhadap *stub* untuk “Where is the shop?”, “My jacket was lost”, dan “Good morning”.

Soal 6.7 (jalur mana?). Untuk setiap pekerjaan berikut, tentukan apakah ia masuk jalur kualitas, jalur cepat, atau tidak masuk jalur mana pun, dan mengapa: (a) menulis enam jawaban FAQ untuk sebuah toko roti dari fakta-faktanya; (b) menentukan apakah sebuah unggahan baru ditulis dalam bahasa Indonesia atau Inggris; (c) menentukan, ketika umpan sedang ditampilkan, apakah setiap tautan di 30 unggahan aman; (d) menyarankan kategori bisnis ketika aturan pemindai ragu.

Soal 6.8 (kata-kata yang jujur). Tulis ulang setiap pesan berikut agar jujur dan berguna bagi pengguna yang bukan orang teknis: (a) halaman Content Engine yang menampilkan templat dengan judul “Draf AI Anda”; (b) “Error 500”; (c) ikon berputar tanpa teks yang sudah berputar 90 detik sementara pengguna berada di urutan keempat antrean.

Soal 6.9 (mengapa harus lebih besar?). Persamaan (6.7) hanya menyajikan draf AI bila $S(d) > S(t)$. Berikan satu argumen untuk perbandingan ketat ini, dan jelaskan apa yang berubah pada Kasus 1 dan pada draf (d) Soal 6.5 bila perbandingan itu diganti dengan $S(d) \geq S(t)$.

Soal 6.10 (sampel kecil). Hasil sebuah gerbang dilaporkan sebagai “w kemenangan dari n kasus”. (a) Dengan $n = 6$, sebutkan semua tingkat kemenangan yang mungkin. (b) Seberapa besar n agar satu kasus mengubah tingkat itu paling banyak 2 poin persentase? (c) Mengapa tetap berguna menerbitkan hasil yang didasarkan pada enam kasus?

6.13 Penyelesaian Soal Latihan

Penyelesaian 6.1. (a) $27 \times 10^9 \times 16/8 = 54 \times 10^9$ bita, sekitar 54 GB. (b) $27 \times 10^9 \times 4.5/8 \approx 15.2 \times 10^9$ bita, sekitar 15,2 GB. Angka 17 GB yang tercantum lebih besar karena (1) format kuantisasi mempertahankan sebagian tensor, seperti *embedding* atau bobot normalisasi, pada presisi yang lebih tinggi, sehingga rata-rata bit per parameter berada di atas angka nominal 4 atau 4,5; dan (2) model Qwen3.5 bersifat multimodal, dan unduhannya memuat penyandi gambar serta metadata di samping bobot model bahasa. Saat berjalan Anda juga memerlukan memori untuk konteks (*KV cache*), yang bertambah seiring banyaknya token dalam percakapan, sehingga mesin memerlukan lebih dari ukuran berkasnya.

Penyelesaian 6.2. (a) Menurut Persamaan (6.3), $n^* = 24{,}000{,}000/(40 - 4) = 24{,}000{,}000/36 \approx 666{,}667$ permintaan per tahun, atau $666{,}667/365 \approx 1{,}827$ per hari. (b) $n^* = 24{,}000{,}000/(20 - 4) = 1{,}500{,}000$ per tahun, sekitar 4.110 per hari: ketika harga layanan *hosted* turun, inferensi lokal memerlukan volume lebih besar agar balik modal. (c) Sisi *hosted*: biaya risiko privasi dan kepatuhan karena data dikirim keluar, serta risiko perubahan harga atau pensiunnya model. Sisi lokal: jam kerja orang untuk memasang, memantau, dan memperbarui server model, serta batas atas mutu, yang dapat merugikan penjualan bila draf lebih lemah.

Penyelesaian 6.3. (a) $T \approx 0 + 2 + 300/6 = 52$ detik. (b) Dengan $k = 2$, Persamaan (6.6) memberi $W \approx 2 \times 52 = 104$ detik sebelum draf dimulai, dan $104 + 52 = 156$ detik, sekitar 2,6 menit, sampai draf selesai. (c) Misalnya: “Ada dua draf di depan draf Anda. Perkiraan mulai sekitar 2 menit lagi, selesai sekitar 3 menit lagi. Anda boleh membiarkan halaman ini terbuka atau kembali nanti.”

Penyelesaian 6.4. Ukurlah dengan sedikit mengubah `stream` di **Kode 6.4**: catat `time.perf_counter()` sebelum `_open` dan sekali lagi ketika baris pertama tiba. Di Linux dan macOS kedua host biasanya memberi median yang sama, beberapa puluh milidetik setelah model dimuat, karena sistem berhasil mencoba IPv6 atau langsung beralih tanpa jeda. Di Windows,

`localhost` dapat diterjemahkan lebih dahulu ke alamat IPv6 `::1`; bila Ollama hanya mendengarkan di IPv4, percobaan koneksi pertama harus gagal atau habis waktunya sebelum klien mencoba `127.0.0.1`, dan itu menambah jeda pada setiap koneksi baru. Profitina mengukur 2,25 detik lawan 0,17 detik di mesin Windows-nya. Apa pun angka Anda, kesimpulannya adalah tuliskan alamat yang Anda maksud.

Penyelesaian 6.5. (a) Templat: cek wajib gagal karena kata terlarang “cheapest”, berapa pun skornya. (b) Templat: draf lolos cek wajib (nama bisnis disebut di satu-satunya kalimat, dan 07, 00, 21, serta 00 semuanya ada di dalam fakta), tetapi skornya 2: ia mencakup dua fakta dan tidak dibuka dengan “Profitina Laundry is”, dan $2 > 3$ bernilai salah. (c) Templat: angka 12 dan 000 dalam “Rp12,000” tidak ada di dalam fakta, sehingga draf mengarang harga, dan itulah kesalahan yang hendak ditangkap aturan angka. (d) Templat, karena seri: draf lolos cek wajib dan skornya 3 (dua fakta ditambah pembukaan *answer-first*), sama dengan 3 milik templat, dan hasil seri dimenangkan templat. Menjalankan cek dari **Kode 6.8** di Python menegaskan keempat hasil ini.

Penyelesaian 6.6. **Kode 6.10** dan **Kode 6.11** memberikan satu penyelesaian.

Kode 6.10 `json_label.py`: minta JSON, validasi bentuk dan nilai yang diizinkan, coba ulang sekali, lalu serahkan kepada manusia.

```
"""Structured output: ask for JSON, validate it, retry once, then give up
honestly. Start stub_ollama.py first.

    python json_label.py [--host URL]
"""
import json
import sys

from ollama_client import NoModelAvailable, OllamaClient

ALLOWED = {"question", "complaint", "praise"}


def valid(text):
    """Return the label if text is {"label": <allowed>}, else None."""
    try:
        obj = json.loads(text)
    except json.JSONDecodeError:
        return None
    if isinstance(obj, dict) and obj.get("label") in ALLOWED:
        return obj["label"]
    return None


def label(client, message, attempts=2):
    for attempt in range(1, attempts + 1):
        try:
            out = client.generate("stub-fast", "CLASSIFY: " + message,
                                  {"seed": attempt}, fmt="json")
        except NoModelAvailable as e:
            return None, f"no local model available ({e})"
        got = valid(out["response"])
        if got:
            return got, f"valid JSON on attempt {attempt}"
        print(f"    attempt {attempt}: rejected {out['response']!r}")
    return None, f"no valid label after {attempts} attempts; a human decides"
```

Kode 6.10 (Lanjutan) *json_label.py: minta JSON, validasi bentuk dan nilai yang diizinkan, coba ulang sekali, lalu serahkan kepada manusia.*

```
def main(argv):
    host = argv[argv.index("--host") + 1] if "--host" in argv \
        else "http://127.0.0.1:7060"
    client = OllamaClient(host, timeout=10)
    for m in ["Where is the shop?", "My jacket was lost", "Good morning"]:
        print(f"{m!r}")
        got, why = label(client, m)
        print(f"    -> {got or 'UNLABELED'} ({why})")


if __name__ == "__main__":
    main(sys.argv[1:])
```

Kode 6.11 `JsonLabel.java`: validasi dan percobaan ulang yang sama dalam Java, memakai `MiniJson`.

```
import java.util.List;
import java.util.Map;
import java.util.Set;

/** Structured output: ask for JSON, validate it, retry once, then give up
 *  honestly. Start StubOllama first.  java JsonLabel [--host URL] */
public class JsonLabel {
    static final Set<String> ALLOWED = Set.of("question", "complaint", "praise");

    record Result(String label, String why) {}

    /** Return the label if text is {"label": <allowed>}, else null. */
    static String valid(String text) {
        try {
            Object label = MiniJson.parseObject(text).get("label");
            return ALLOWED.contains(label) ? (String) label : null;
        } catch (RuntimeException e) {
            return null;
        }
    }

    static Result label(OllamaClient client, String message, int attempts) {
        for (int attempt = 1; attempt <= attempts; attempt++) {
            Map<String, Object> out;
            try {
                out = client.generate("stub-fast", "CLASSIFY: " + message,
                                      Map.of("seed", attempt), "json");
            } catch (OllamaClient.NoModelAvailable e) {
                return new Result(null, "no local model available ("
                                  + e.getMessage() + ")");
            }
            String text = out.get("response").toString();
            String got = valid(text);
            if (got != null) {
                return new Result(got, "valid JSON on attempt " + attempt);
            }
            System.out.println("    attempt " + attempt + ": rejected '"
                               + text + "'");
        }
        return new Result(null, "no valid label after " + attempts
                          + " attempts; a human decides");
    }
```

Kode 6.11 (Lanjutan) *JsonLabel.java: validasi dan percobaan ulang yang sama dalam Java, memakai MiniJson.*

```
    public static void main(String[] args) {
        List<String> a = List.of(args);
        String host = a.contains("--host") ? a.get(a.indexOf("--host") + 1)
                                           : "http://127.0.0.1:7060";
        OllamaClient client = new OllamaClient(host, 10.0);
        for (String m : List.of("Where is the shop?", "My jacket was lost",
                                "Good morning")) {
            System.out.println("'" + m + "'");
            Result r = label(client, m, 2);
            String shown = r.label() == null ? "UNLABELED" : r.label();
            System.out.println("    -> " + shown + " (" + r.why() + ")");
        }
    }
}
```

Terhadap *stub*, kedua program mencetak:

```
'Where is the shop?'
    -> question (valid JSON on attempt 1)
'My jacket was lost'
    -> complaint (valid JSON on attempt 1)
'Good morning'
    attempt 1: rejected '{"label": "other"}'
    attempt 2: rejected '{"label": "other"}'
    -> UNLABELED (no valid label after 2 attempts; a human decides)
```

Dua pesan pertama langsung mendapat label yang sah. Untuk “Good morning”, model dua kali mengembalikan JSON yang bentuknya benar dengan label `other`, yang tidak termasuk himpunan yang diizinkan, sehingga validasi menolaknya, percobaan ulang tidak mengubah apa pun, dan program menyatakannya alih-alih menerima label yang tidak diharapkan bagian lain sistem. Ada dua hal yang berlaku umum: validasilah *isi* keluaran terstruktur, bukan hanya sintaksnya; dan batasi percobaan ulang, karena model yang gagal dua kali dengan cara yang sama biasanya gagal pula untuk ketiga kalinya.

Penyelesaian 6.7. (a) Jalur kualitas: teks panjang yang akan diterbitkan, layak diperiksa dengan gerbang. (b) Sering kali tidak perlu jalur: aturan berbasis karakter dan daftar kata mengenali bahasa Indonesia atau Inggris dengan andal, dan jalur cepat dapat menjadi cadangan untuk teks campuran. (c) Tidak ada jalur saat menampilkan: nilai setiap tautan ketika diunggah (lantai aturan ditambah opini opsional jalur cepat) dan simpan putusannya; umpan hanya membaca putusan yang tersimpan (Subbab 6.5.4). (d) Jalur cepat dengan lantai aturan: label pendek yang diperlukan selagi pengguna menunggu, dan ketika model ragu atau tidak tersedia, perilaku yang jujur adalah berkata “belum yakin” lalu meminta pengguna memilih.

Penyelesaian 6.8. (a) “Draf templat (penyusun draf AI sedang tidak tersedia). Teks ini disusun langsung dari fakta Anda. Coba lagi beberapa menit lagi untuk mendapatkan draf AI.” (b) “Terjadi kesalahan di pihak kami saat membuat draf Anda. Tidak ada isian Anda yang hilang. Silakan coba lagi; jika terus terjadi, beri tahu kami di info@profitina.com.” (c) “Anda berada di urutan ke-4. Perkiraan waktu tunggu: sekitar 3 menit. Menyusun draf — 120 dari sekitar 400 token.”

Penyelesaian 6.9. Perbandingan ketat menjadikan templat yang benar dan dapat diramalkan sebagai pilihan bawaan setiap kali AI tidak menambahkan sesuatu yang terukur; menyajikan teks hasil generasi membawa risiko yang tidak tertangkap skor, sehingga teks itu harus membuktikan kelayakannya. Dengan $S(d) \geq S(t)$, Kasus 1 tidak berubah (4 lebih besar daripada 3), tetapi draf (d) Soal 6.5, yang seri pada 3, akan disajikan menggantikan templat.

Penyelesaian 6.10. (a) $0, \frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}, 1$, yaitu 0%, 16,7%, 33,3%, 50%, 66,7%, 83,3%, dan 100%. (b) Satu kasus mengubah tingkat itu sebesar $1/n$, dan $1/n \leq 0.02$ mensyaratkan $n \geq 50$. (c) Karena hasil itu jujur dan dapat diperiksa: ia menyebut apa yang diukur dan pada berapa kasus, ia menunjukkan bahwa revisi menggeser hasil ke arah yang benar, dan ia didukung gerbang yang memutuskan setiap permintaan menurut kelayakannya sendiri, sehingga tidak ada pengguna yang bergantung pada presisi persentase itu.

Daftar Pustaka

[1] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017, pp. 5998–6008.

[2] Stack Overflow, "AI," *2025 Stack Overflow Developer Survey*, Jul. 2025. [Online]. Available: https://survey.stackoverflow.co/2025/ai

[3] T. Dettmers and L. Zettlemoyer, "The case for 4-bit precision: k-bit inference scaling laws," in *Proc. 40th Int. Conf. Machine Learning (ICML)*, 2023, pp. 7750–7774.

[4] ggml-org, "GGUF file format," *ggml documentation*. [Online]. Available: https://github.com/ggml-org/ggml/blob/master/docs/gguf.md (accessed Sep. 24, 2026).

[5] Ollama, "qwen2.5," *Ollama model library*. [Online]. Available: https://ollama.com/library/qwen2.5 (accessed Sep. 24, 2026).

[6] Profitina, *Profitina Whitepaper — Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[7] Cisco Systems, "Cisco's 2025 Data Privacy Benchmark Study: Privacy landscape grows increasingly complex in the age of AI," press release, Apr. 2, 2025. [Online]. Available: https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2025/m04/cisco-2025-data-privacy-benchmark-study-privacy-landscape-grows-increasingly-complex-in-the-age-of-ai.html

[8] Cisco Systems, "AI fuels surge in data privacy investments and redefines governance, Cisco reports (Cisco 2026 Data and Privacy Benchmark Study)," press release, Jan. 26, 2026. [Online]. Available: https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2026/m01/ai-data-privacy-investments-governance-cisco-report.html

[9] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi* (UU PDP), Oct. 17, 2022, esp. Art. 10(1). [Online]. Available: https://jdih.komdigi.go.id/produk_hukum/view/id/832/t/undangundang+nomor+27+tahun+202

[10] Ollama, "Releases: v0.34.3 (Sep. 19, 2026) and v0.34.4-rc0 (Sep. 23, 2026)," *GitHub*. [Online]. Available: https://github.com/ollama/ollama/releases (accessed Sep. 24, 2026).

[11] Ollama, "Generate a response (POST /api/generate)," *Ollama API documentation*. [Online]. Available: https://docs.ollama.com/api/generate (accessed Sep. 24, 2026).

[12] Qwen Team, A. Yang *et al.*, "Qwen2.5 technical report," arXiv:2412.15115, Dec. 2024 (rev. Jan. 2025).

[13] A. Yang *et al.*, "Qwen3 technical report," arXiv:2505.09388, May 2025.

[14] Ollama, "qwen3," *Ollama model library*. [Online]. Available: https://ollama.com/library/qwen3 (accessed Sep. 24, 2026).

[15] Ollama, "qwen3.5," *Ollama model library*. [Online]. Available: https://ollama.com/library/qwen3.5 (accessed Sep. 24, 2026).

[16] Wikipedia contributors, "Qwen," *Wikipedia* (secondary source for the 2026 release list). [Online]. Available: https://en.wikipedia.org/wiki/Qwen (accessed Sep. 24, 2026).

[17] Qwen Team, "Qwen3.8," *GitHub repository QwenLM/Qwen3.8*, news of Aug. 12 and Aug. 14, 2026. [Online]. Available: https://github.com/QwenLM/Qwen3.8 (accessed Sep. 24, 2026).

[18] T. Bray, Ed., "The JavaScript Object Notation (JSON) data interchange format," IETF RFC 8259, Dec. 2017.

Bab 7

Siklus Umpan Balik: Manusia Memutuskan, Model Belajar

Setelah menyelesaikan bab ini, Anda akan mampu menjelaskan mengapa produk AI yang tidak dapat belajar dari penggunanya perlahan tertinggal dari mereka, dan mengapa produk yang belajar dari penggunanya tanpa manusia yang memegang kendali itu berbahaya; menelusuri *pipeline* umpan balik lima langkah Profitina, yaitu kirim → kurasi → dataset → latih dan ukur → rilis, dan menyebutkan pada setiap langkah apa yang dikerjakan mesin, apa yang diputuskan manusia, dan apa yang dapat dilihat oleh pemberi umpan balik; mengubah umpan balik yang sudah dikurasi menjadi dataset berisi pasangan preferensi dalam format JSON Lines; mengukur model sebelum dan sesudah dengan himpunan uji terpisah, menghitung tingkat kelolosan, dan menemukan regresi; menjelaskan bagaimana Profitina memberi hadiah berupa sejumlah kecil PROFIT kepada orang yang memperbaiki jawaban AI, mengapa hadiah itu memiliki batas harian yang diatur ulang pada tengah malam Waktu Indonesia Barat, dan mengapa hanya manusia terverifikasi yang memperolehnya; serta menyatakan prinsip yang mengalir di sepanjang bab ini, *aplikasi memberi saran, pengguna yang memutuskan*. Lab membangun *pipeline* mini yang lengkap dalam Python dan Java: ia menerima sebelas butir umpan balik tentang jawaban layanan bantuan Profitina Laundry, memilahnya, menerapkan keputusan seorang manusia, menulis dataset JSONL, memberikan hadiah lab di bawah batas harian, mengukur delapan pertanyaan uji terpisah sebelum dan sesudah, lalu berhenti untuk menunggu keputusan rilis dari manusia.

7.1 Titik Berangkat Kita

Bab 6 membangun perangkat AI lokal: dua jalur, gerbang kualitas yang hanya menyajikan draf AI bila mengalahkan templat, dan perilaku jujur ketika tidak ada model yang tersedia. Semua itu membuat model *aman dipakai* hari ini. Tidak satu pun membuat model *lebih baik* esok hari.

Model yang dibiarkan tidak akan membaik. Pengetahuannya membeku pada akhir pelatihan, ia mengulang kesalahan-kesalahan kecil yang sama, dan dunia yang digambarkannya terus bergerak: binatu mengubah harganya, metode pembayaran baru muncul, sebuah peraturan mulai berlaku. Orang yang pertama kali menyadari kesalahan-kesalahan itu adalah para pengguna. Seorang pemilik merek membaca draf Content Engine dan melihat bahwa jawaban FAQ tentang jam buka terlalu kabur. Seorang mahasiswa di kanal kelas menyadari bahwa penjelasan hasil pindai memakai sebuah istilah secara keliru. Masing-masing memegang sepotong kecil pengetahuan yang tidak dimiliki model. Bagi sebuah platform yang melayani pengguna di Indonesia, potongan-potongan itu juga sering bersifat lokal: nama metode pembayaran, kebiasaan jam buka usaha kecil, istilah yang lazim di suatu daerah, atau aturan yang baru berlaku. Model yang dilatih terutama dari teks global tidak mungkin mengetahui semuanya, sehingga umpan balik dari pengguna setempat justru menjadi sumber pengetahuan yang paling berharga.

Bab ini membahas cara mengumpulkan potongan-potongan itu dengan cermat dan adil, lalu mengubahnya menjadi perbaikan tanpa pernah membiarkan mesin memutuskan sendiri apa arti “lebih baik”. Bab ini menjawab empat pertanyaan. Bagaimana umpan balik berpindah dari seseorang ke sebuah model (Subbab 7.2 sampai 7.4)? Bagaimana umpan balik dibersihkan,

disimpan, dan diukur (Subbab 7.5 sampai 7.7)? Siapa yang memutuskan apakah sebuah perbaikan dirilis (Subbab 7.8)? Dan bagaimana Profitina berterima kasih kepada orang yang membantu tanpa menciptakan mesin yang membayar bot untuk memproduksi kebisingan (Subbab 7.9)?

7.2 Mengapa Umpan Balik, dan Mengapa Manusia yang Memutuskan

7.2.1 Model belajar dari manusia

Setiap model bahasa yang berguna saat ini dibentuk oleh penilaian manusia setelah pelatihan awalnya. Metode yang paling dikenal, *reinforcement learning from human feedback* (RLHF, pembelajaran penguatan dari umpan balik manusia), meminta orang membandingkan dua jawaban dari model yang sama dan menyatakan mana yang lebih baik; perbandingan itu melatih model agar lebih menyukai jawaban yang disukai manusia [1], [2]. Studi InstructGPT yang mempopulerkan metode ini menemukan bahwa orang lebih menyukai jawaban model 1,3 miliar parameter yang disetel dengan cara ini daripada jawaban model 175 miliar parameter tanpa penyetelan itu [1]. Pelajarannya menggembirakan bagi tim kecil: model sederhana ditambah penilaian manusia yang cermat dapat mengalahkan model yang jauh lebih besar tanpa penilaian itu.

Membandingkan juga lebih mudah bagi manusia daripada menulis jawaban sempurna dari nol. Sulit menulis jawaban FAQ ideal tentang layanan ekspres; mudah untuk berkata "jawaban ini salah, enam jam, bukan dua hari" lalu mengusulkan kalimat yang lebih baik. *Pipeline* umpan balik Profitina dibangun di atas koreksi kecil dan konkret semacam itu.

7.2.2 Mengapa manusia harus memutuskan

Jika umpan balik begitu berharga, mengapa sistem tidak menerima setiap usulan lalu melatih ulang dirinya secara otomatis? Ada tiga alasan, dan masing-masing muncul di lab.

1. **Umpan balik bisa salah.** Seorang pengguna mungkin mengusulkan harga yang lebih murah karena ia berharap harganya lebih murah, atau menyalin jawaban dari pesaing. Butir F08 di lab melakukan persis hal itu.
2. **Umpan balik bisa bermusuhan atau otomatis.** Sistem apa pun yang memberi hadiah untuk masukan akan menarik orang, dan program, yang memproduksi masukan hanya demi hadiahnya. Bab 5 menguraikan bagaimana Fair-Play melindungi komunitas; pertahanan yang sama melindungi siklus umpan balik.
3. **Perbaikan di satu tempat dapat merusak tempat lain.** Koreksi yang membuat satu jawaban menjadi tepat dapat menghapus detail yang diperlukan pertanyaan lain. Hanya pengukuran yang dapat mengungkapnya, dan hanya manusia yang dapat menimbang sebuah perbaikan terhadap sebuah regresi.

Survei terhadap para praktisi menunjuk ke arah yang sama. Dalam Stack Overflow Developer Survey 2025, frustrasi paling umum terhadap perkakas AI, yang disebut 66% responden, adalah "solusi AI yang hampir benar, tetapi tidak sepenuhnya", dan 75,3% menyatakan tetap akan bertanya kepada seseorang "ketika saya tidak memercayai jawaban AI" [3]. Penelitian tentang interaksi manusia–AI menganjurkan rancangan yang memperjelas apa yang dapat dilakukan sistem, mendukung koreksi yang efisien, dan membiarkan orang mengabaikan atau mengesampingkan sistem [4].

7.2.3 Aplikasi memberi saran, pengguna yang memutuskan

Profitina menuliskan prinsip ini sebagai aturan internal: *aplikasi memberi saran; pengguna yang memutuskan; jangan pernah menyajikan keluaran sebagai jaminan* [5]. Prinsip ini berlaku ke dua arah.

Ke arah pengguna produk, prinsip ini berarti skor visibilitas adalah perkiraan metodologis, draf Content Engine adalah usulan, dan platform menyatakannya: hasil visibilitas adalah perkiraan metodologis, bukan jaminan atas keluaran mesin AI pihak ketiga yang sesungguhnya [5]. Ketika pengisian otomatis kategori ragu, ia berkata "belum yakin" dan meminta pengguna memilih, alih-alih menebak diam-diam.

Ke arah produk itu sendiri, prinsip ini berarti setiap keputusan yang mengubah apa yang dikatakan produk kepada orang, baik apakah sebuah usulan masuk dataset maupun apakah versi model yang baru dirilis, diambil oleh seorang manusia, dengan hasil pengukuran *pipeline* sebagai saran. Hukum Indonesia menunjuk ke arah yang sama: Pasal 10 UU PDP Nomor 27 Tahun 2022 memberi orang hak untuk mengajukan keberatan atas keputusan yang hanya didasarkan pada pemrosesan otomatis yang berdampak signifikan bagi mereka [6]. Siklus umpan balik yang diputuskan manusia bukan hanya rekayasa yang lebih baik; ia adalah jenis sistem yang diharapkan hukum.

> **DEFINISI.** Proses *human-in-the-loop* (manusia di dalam siklus) adalah proses tempat mesin mengusulkan, memilah, atau mengukur, sedangkan seorang manusia yang jelas identitasnya mengambil setiap keputusan yang mengubah perilaku produk, dengan alasan yang dicatat. Di bab ini, *pipeline memberi saran* dan manusia *memutuskan*.

7.2.4 Sebuah kisah dalam lima langkah

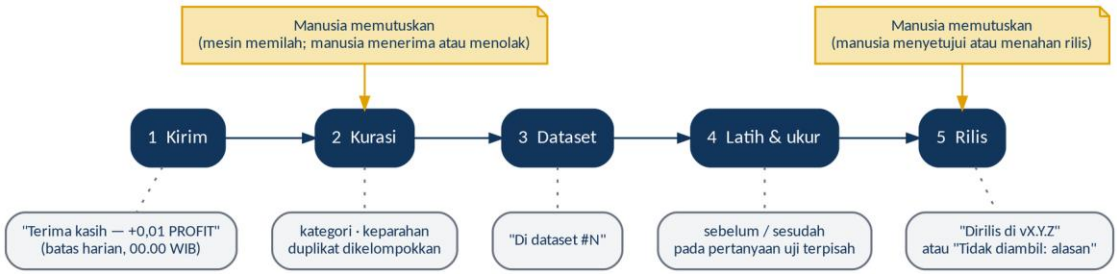
Sebelum masuk ke rinciannya, inilah seluruh siklus itu sebagai sebuah kisah. Ibu Sari mengelola toko roti kecil di Sidoarjo dan memakai Profitina agar situs webnya lebih mudah dikutip mesin jawaban. Content Engine menyusun FAQ untuknya, dan salah satu jawabannya menyatakan bahwa toko itu "mengantar ke seluruh Jawa Timur". Padahal tokonya hanya mengantar dalam radius 5 km. Ia menekan tombol umpan balik pada jawaban itu, menulis "Salah: kami hanya mengantar dalam radius 5 km dari toko", dan mengusulkan kalimat yang sudah dikoreksi. Ucapan terima kasih muncul, disertai hadiah PROFIT yang kecil.

Malam itu butirnya menjadi salah satu dari empat puluh butir di antrean tinjauan. Panggilan pemilah telah memberinya label *wrong fact* (fakta salah), keparahan *medium*, dan mengelompokkannya dengan dua laporan serupa dari toko roti lain tentang wilayah antar. Seorang peninjau membaca kelompok itu, melihat bahwa draf-draf mengarang wilayah antar ketika sebuah bisnis tidak menyebutkannya, lalu menerima butir Ibu Sari dengan alasan "wilayah antar harus berasal dari fakta pengguna". Dasbornya kini menampilkan "In dataset #212".

Seminggu kemudian kandidat berikutnya diukur. Pada pertanyaan uji terpisah tentang pengantaran, kandidat itu kini menanyakan atau memakai wilayah milik bisnis itu sendiri alih-alih mengarangnya; dua jawaban lain yang tidak berkaitan menjadi sedikit lebih buruk. Peninjau membaca laporannya, memutuskan bahwa regresi itu hanya soal susunan kata yang kecil, mencatat keputusannya, lalu menyetujui rilis. Dasbor Ibu Sari bertuliskan "Shipped in v1.4", dan karena ia memilih ikut serta, nama samarannya muncul di halaman "You changed this". Ia tidak perlu memahami apa pun tentang model untuk memperbaiki sebuah model. Itulah janji siklus umpan balik yang transparan dan diatur manusia; sisa bab ini menjelaskan setiap langkahnya. (Ibu Sari, toko rotinya, dan angka-angka dalam kisah ini adalah rekaan untuk ilustrasi.)

7.3 *Pipeline* Lima Langkah

Pipeline umpan balik Profitina memiliki lima langkah, dan semuanya sudah berjalan per September 2026 [5]. **Gambar 7.1** memperlihatkannya beserta apa yang dilihat pemberi umpan balik pada setiap langkah, dan di mana manusia memutuskan.



Gambar 7.1 *Pipeline* umpan balik lima langkah. Kotak gelap adalah langkah-langkahnya; kotak abu-abu menunjukkan apa yang dilihat pemberi umpan balik; catatan berwarna kuning menandai dua tempat manusia memutuskan. Setelah rilis, siklus dimulai lagi dengan umpan balik baru.

Langkah 1: Kirim. Pengguna mengirim umpan balik tentang sesuatu yang dihasilkan AI: penjelasan hasil pindai, draf Content Engine, usulan kategori. Umpan balik atas hasil pindai dapat memperoleh sejumlah kecil PROFIT, dalam batas harian (Subbab 7.9). Pemberi umpan balik melihat ucapan terima kasih dan, bila berlaku, hadiahnya.

Langkah 2: Kurasi. Satu panggilan AI lokal di jalur cepat memilah setiap butir: ia mengusulkan kategori, tingkat keparahan, dan ringkasan satu baris, serta mengelompokkan butir-butir yang menyatakan hal yang sama. Kemudian seorang manusia meninjau antrean yang sudah dipilah dan menerima atau menolak setiap butir, dengan alasan. Pemberi umpan balik melihat perubahan statusnya.

Langkah 3: Dataset. Butir yang diterima ditambahkan ke dataset pelatihan yang bernomor. Pemberi umpan balik melihat "In dataset #N", tanda yang terlihat dan dapat diperiksa bahwa umpan baliknya dipakai.

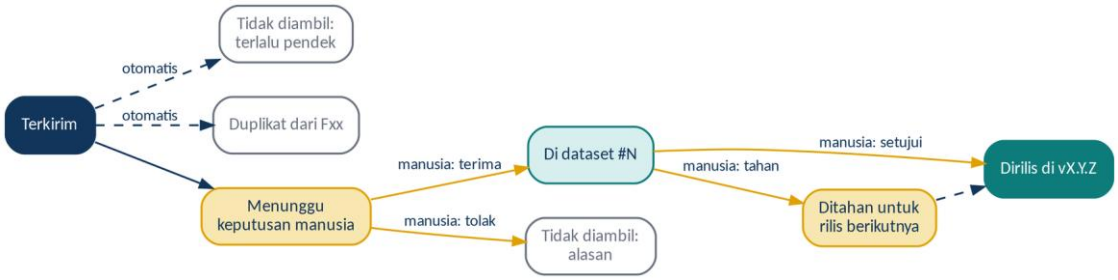
Langkah 4: Latih dan ukur. Dataset dipakai untuk memperbaiki perilaku model lokal, dan setiap kandidat diukur pada contoh-contoh uji terpisah sebelum dan sesudah, sehingga perbaikan maupun regresi sama-sama terlihat. Apakah dan kapan bobot model disetel ulang adalah pilihan teknis di dalam langkah ini; penyetelan penuh bobot model dengan metode seperti LoRA [7] masih merupakan butir peta jalan per September 2026, dan buku ini tidak menggambarkannya sebagai fitur yang sudah berjalan.

Langkah 5: Rilis. Seorang manusia memutuskan apakah kandidat dirilis. Pemberi umpan balik melihat "Shipped in vX.Y.Z" atau "Not taken", selalu dengan alasan. Pemberi umpan balik yang memilih ikut serta diberi kredit dengan nama samaran di halaman publik "You changed this" [5].

Dua sifat membuat *pipeline* ini layak dipercaya. Ia *transparan*: pada setiap langkah pemberi umpan balik dapat melihat di mana butirnya berada, yang mengubah "kami menghargai umpan balik Anda" dari slogan menjadi sebaris status. Dan ia *diatur manusia*: keputusan produk selalu diambil oleh manusia [5]. Mesin mengerjakan bagian yang melelahkan, yaitu memilah, mengelompokkan, dan mengukur, sedangkan manusia mengerjakan bagian yang memerlukan pertimbangan.

7.4 Perjalanan Satu Butir Umpan Balik

Ada gunanya mengikuti satu butir melewati *pipeline* sebagai *mesin keadaan* (*state machine*): himpunan kecil keadaan bernama dan perpindahan yang diizinkan di antaranya. **Gambar 7.2** menggambar keadaan-keadaan yang dipakai lab bab ini, yang mencerminkan status-status yang ditampilkan produk.



Gambar 7.2 Perjalanan satu butir umpan balik. Perpindahan bergaris putus-putus bersifat otomatis dan dapat dijelaskan; perpindahan berwarna kuning adalah keputusan manusia. Hanya butir yang diterima yang dapat dirilis.

Tabel 7.1 mendaftar setiap keadaan, siapa yang menaruh sebuah butir di sana, dan apa yang dibaca pemberi umpan balik.

Tabel 7.1 Keadaan-keadaan butir umpan balik di lab bab ini, siapa yang memindahkan butir ke setiap keadaan, dan apa yang dilihat pemberi umpan balik.

Keadaan	Siapa yang memutuskan	Apa yang dibaca pemberi umpan balik
Terkirim	pemberi umpan balik	"Terima kasih, umpan balik Anda sudah diterima"
Tidak diambil: terlalu pendek	aturan otomatis	"Tidak diambil: terlalu pendek"
Duplikat dari F*xx*	aturan otomatis	"Duplikat dari F*xx*", dengan tautan ke butir sebelumnya
Menunggu keputusan manusia	belum ada	"Menunggu tinjauan"
Di dataset #*N*	manusia (menerima)	"Di dataset #*N*"
Tidak diambil: *alasan*	manusia (menolak)	"Tidak diambil", dengan alasannya
Dirilis di v*X.Y.Z*	manusia (menyetujui rilis)	"Dirilis di v*X.Y.Z*"
Ditahan untuk rilis berikutnya	manusia (menahan rilis)	"Di dataset #*N*; belum dirilis"

Tiga aturan terlihat di tabel itu. Perpindahan otomatis hanya pernah *membuang* kebisingan (terlalu pendek, duplikat), dan masing-masing menyebut alasannya. Setiap perpindahan yang mengubah apa yang akan dilakukan produk adalah keputusan manusia. Dan tidak ada keadaan tempat sebuah butir hilang diam-diam: penolakan pun disertai alasan.

Ikuti satu butir dari lab untuk melihat keadaan-keadaan itu bekerja. F06 datang dari pengguna u5 pukul 12.00 WIB: "Yes, free pickup and delivery within 3 km." Butir itu cukup panjang dan bukan hampir-kembar dari butir mana pun sebelumnya, sehingga tidak ada aturan otomatis yang menyentuhnya, dan ia masuk ke *Menunggu keputusan manusia*. Peninjau memastikan jaraknya kepada pemilik toko lalu menerimanya, sehingga butir itu berpindah ke *Di dataset #5*. Ketika kandidat yang memuatnya diukur dan seseorang menyetujui rilis, butir itu akan berpindah ke *Dirilis di v1.1*; pada eksekusi pertama lab rilisnya ditahan, sehingga butir itu tetap *Di dataset #5* dan *Ditahan untuk rilis berikutnya*. Hadiahnya mengikuti jalur terpisah yang tidak digambarkan mesin keadaan: u5 belum memverifikasi alamat surelnya, sehingga hadiahnya ditahan pada 0,00 sampai verifikasi dilakukan. Memisahkan keputusan isi dari keputusan hadiah itu disengaja:

koreksi yang baik tetap dipakai walaupun pemberinya belum memenuhi syarat untuk menerima hadiah, dan hadiah tidak pernah menjadi alasan untuk menerima sebuah koreksi.

7.5 Kurasi: Mesin Memilah, Manusia Memutuskan

7.5.1 Mengapa memilah sebelum manusia melihat

Perhatian manusia adalah sumber daya paling langka di dalam *pipeline*. Jika seorang peninjau menerima seratus butir menurut urutan kedatangan, masalah keselamatan yang mendesak mungkin berada di butir ke-87, dan butir ke-12, ke-40, dan ke-71 mungkin menyatakan hal yang sama dengan kata-kata berbeda. Pemilahan memecahkan kedua masalah itu: taruh butir yang parah di depan, dan kelompokkan duplikat agar peninjau membaca setiap gagasan sekali saja.

Profitina memakai satu panggilan AI jalur cepat per butir untuk pemilahan ini, mengusulkan kategori, tingkat keparahan, dan ringkasan satu baris serta mengelompokkan duplikat [5]. Lab mengerjakan tugas yang sama dengan aturan yang transparan alih-alih model, agar setiap langkah dapat ditelusuri dengan tangan: komentar yang menyebut “unsafe” atau “damage” menjadi *unsafe advice* (saran berbahaya) dengan keparahan *high*, yang menyebut “wrong” menjadi *wrong fact* (fakta salah) dengan keparahan *medium*, yang menyebut “missing” menjadi *missing info* (informasi kurang), dan sisanya *other*. Soal 7.6 meminta Anda mengganti aturan-aturan itu dengan panggilan ke jalur cepat dari Bab 6, sambil mempertahankan aturan sebagai lantai di bawahnya.

7.5.2 Menemukan duplikat dengan kesamaan kata

Dua usulan hampir kembar bila sebagian besar katanya sama. Ukuran sederhana dan klasik untuk itu adalah *kesamaan Jaccard* (*Jaccard similarity*) dari himpunan kata keduanya. Jika A dan B adalah himpunan kata berbeda dalam dua teks, maka

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{7.1}$$

yaitu banyaknya kata yang sama-sama dimiliki dibagi banyaknya kata berbeda di salah satu atau kedua teks. $J = 1$ berarti kata-katanya sama, $J = 0$ berarti tidak ada yang sama. Lab lebih dahulu mengecilkan semua huruf, membuang koma di dalam angka sehingga “Rp12,000” menjadi satu kata “rp12000”, dan membuang kata-kata yang sangat umum seperti “the”, “is”, dan “we”, yang hampir tidak membawa makna.

> **CONTOH.** Butir F01 dan F02 di lab sama-sama mengoreksi jawaban tentang layanan ekspres. Setelah dibersihkan, F01 memiliki kata-kata {express, service, ready, 6, hours, costs, rp12000, per, kilogram} (9 kata) dan F02 memiliki {express, ready, 6, hours, costing, rp12000, per, kilogram} (8 kata). Keduanya berbagi 7 kata (express, ready, 6, hours, rp12000, per, kilogram), dan bersama-sama memiliki $9 + 8 - 7 = 10$ kata berbeda. Menurut Persamaan (7.1), $J = 7/10 = 0.7$. Lab memperlakukan $J \geq 0.7$ di antara dua butir tentang jawaban yang sama sebagai duplikat, sehingga F02, yang datang belakangan, ditandai “duplicate of F01”. Ambang 0,7 adalah pilihan lab ini sendiri.

Deteksi duplikat otomatis menghemat waktu peninjau, tetapi di sinilah pula mesin dapat keliru dengan akibat yang mahal, seperti akan diperlihatkan butir F11 di lab: butir yang tampak mirip dengan butir sebelumnya bisa jadi menambahkan satu-satunya detail yang penting. Karena itu lab

menjaga ambangnya tetap tinggi, melaporkan setiap keputusan otomatis beserta alasannya, dan Soal 7.4 meminta Anda menjelajahi apa yang terjadi bila ambang itu diturunkan.

7.5.3 Keputusan manusia

Setelah pemilahan, seorang peninjau membaca setiap butir yang tersisa dan menuliskan keputusan beserta alasan: *terima* ("matches the price list") atau *tolak* ("contradicts the price list (Rp7,000)"). Di lab keputusan-keputusan ini disimpan dalam berkas kecil, `decisions.tsv`, yang mewakili pekerjaan peninjau di layar admin. Butir tanpa keputusan tetap "awaiting a human decision"; ia tidak masuk dataset secara bawaan. Inilah pilihan yang konservatif: diam bukan berarti setuju.

7.5.4 Mengurutkan antrean

Dengan kategori dan tingkat keparahan pada setiap butir, antrean tinjauan dapat diurutkan sehingga peninjau yang hanya punya sepuluh menit melihat butir terpenting lebih dahulu. Urutan yang sederhana dan transparan mengurutkan menurut keparahan dulu (tinggi, lalu sedang, lalu rendah), kemudian menurut besarnya kelompok duplikat (gagasan yang dilaporkan lima kali lebih penting daripada yang dilaporkan sekali), dan terakhir menurut waktu kedatangan (yang lebih lama didahulukan, agar tidak ada yang menunggu selamanya). Dengan menuliskan urutan itu sebagai *kunci pengurutan* (*sort key*), yaitu tupel

$$\kappa(i) = (-\mathrm{kep}(i),\ -\mathrm{grup}(i),\ t_i), \tag{7.2}$$

dengan kep bernilai 3 untuk tinggi, 2 untuk sedang, dan 1 untuk rendah, $\mathrm{grup}(i)$ adalah banyaknya butir yang dikelompokkan bersama i, dan t_i adalah waktu kedatangannya, mengurutkan butir menurut κ yang menaik dalam Persamaan (7.2) menghasilkan urutan tadi, karena tupel dibandingkan elemen demi elemen dan tanda minus mengubah "yang lebih besar dulu" menjadi "yang lebih kecil dulu". Di lab, F04 (*unsafe advice*, tinggi) akan tampil pertama, disusul dua butir *wrong fact* yang tersisa setelah duplikat dikelompokkan, dengan F01 mendahului F05 karena kelompoknya memuat duplikat F02. Soal 7.11 menerapkan urutan ini pada antrean lab. Pola yang sama, antrean prioritas yang diurutkan menurut tupel, menggerakkan penjadwal pindai ulang di Bab 9.

7.6 Dataset: Pasangan Preferensi dalam JSON Lines

7.6.1 Seperti apa contoh pelatihan itu

Bentuk yang berguna untuk umpan balik atas sebuah jawaban adalah *pasangan preferensi* (*preference pair*): *prompt*-nya, jawaban yang diberikan model (jawaban *rejected*, yang ditolak), dan jawaban yang lebih disukai manusia (jawaban *chosen*, yang dipilih). Pasangan preferensi persis merupakan apa yang dikumpulkan RLHF [1], dan merupakan masukan bagi *direct preference optimization* (DPO), metode lebih sederhana yang menyetel model langsung dari pasangan semacam itu tanpa melatih model hadiah terpisah [8]. Bahkan tanpa pelatihan apa pun, kumpulan pasangan preferensi tetap bernilai: ia adalah catatan yang presisi dan dapat ditinjau tentang apa yang dahulu dikatakan produk dan apa yang disepakati orang seharusnya dikatakan.

7.6.2 JSON Lines

Dataset semacam ini biasanya disimpan dalam format *JSON Lines*, disebut juga JSONL: berkas teks dengan satu objek JSON lengkap pada setiap baris [9]. Format ini dipakai Ollama untuk *streaming*

(Bab 6), dengan alasan yang sama: ia dapat ditulis satu rekaman demi satu rekaman, dibaca satu rekaman demi satu rekaman, ditambahi tanpa menulis ulang berkasnya, dan dipotong pada batas baris mana pun. Setiap baris adalah nilai JSON biasa sebagaimana didefinisikan RFC 8259 [10], disandikan dalam UTF-8, tanpa pindah baris di dalamnya.

> **DEFINISI.** Berkas *JSON Lines* (JSONL) adalah berkas teks UTF-8 yang setiap barisnya berisi satu nilai JSON lengkap, biasanya objek, dan baris-barisnya dipisahkan oleh `\n`. Dataset pasangan preferensi dalam JSONL memuat satu pasangan per baris.

7.6.3 Apa yang tidak boleh masuk dataset

Dataset pelatihan berumur panjang dan banyak disalin, sehingga harus bersih. Isinya adalah substansi koreksi, bukan orangnya: tanpa alamat surel, tanpa nama asli, tanpa nomor telepon, dan tanpa teks bebas yang mengenali seseorang. Baris-baris dataset di lab hanya memuat pengenal butir, *prompt*, jawaban *rejected* dan *chosen*, serta kategori. Kredit bagi pemberi umpan balik adalah urusan terpisah yang ditangani dengan nama samaran dan persetujuan pemberi umpan balik (Subbab 7.3). Bab 11 membahas aturan privasinya secara lengkap; di sini patokannya adalah bahwa dataset harus dapat diterbitkan tanpa merugikan siapa pun.

Contoh kecil memperjelas patokan itu. Misalkan seorang pelanggan menulis komentar “Saya Budi dari Jalan Mawar 12, jaket saya hilang, hubungi 0812-xxxx”. Isi koreksinya, bahwa jawaban tentang barang hilang harus menjelaskan prosedur klaim, bernilai bagi dataset. Nama, alamat, dan nomor teleponnya tidak bernilai sama sekali bagi model dan berbahaya bila tersebar. Karena itu yang masuk dataset hanyalah pasangan *prompt*, jawaban lama, dan jawaban yang lebih baik; komentar aslinya tetap berada di sistem umpan balik dengan aturan retensi dan hak hapusnya sendiri. Pemisahan ini juga memudahkan hak penghapusan: bila Budi menghapus akunnya, komentarnya ikut terhapus, sedangkan pasangan di dataset tidak memuat apa pun yang mengenalinya.

7.6.4 Versi dan asal-usul

Dataset berubah setiap minggu, sehingga setiap pemakaiannya harus menyebut versi *mana* yang dipakai. Tiga kebiasaan memungkinkan hal itu. Pertama, dataset bersifat *append-only* (hanya ditambah): butir yang diterima mendapat nomor berikutnya dan tidak pernah dinomori ulang, sehingga “In dataset #212” bermakna sama selamanya. Kedua, setiap rilis mencatat rentang nomor butir yang dipakai untuk melatih atau mengukurnya, misalnya “v1.4 memakai butir #1–#240”. Ketiga, setiap butir menyimpan *asal-usulnya* (*provenance*): butir umpan balik sumbernya (kolom `id` di lab), keputusan beserta alasannya, dan tanggalnya. Dengan ketiga kebiasaan ini, pertanyaan seperti “mengapa layanan bantuan sekarang berkata 6 jam?” memiliki jawaban yang presisi: butir #1, dari umpan balik F01, diterima karena sesuai dengan daftar harga, pertama kali dirilis pada rilis yang memuatnya.

Asal-usul pula yang membuat halaman publik “You changed this” jujur. Halaman itu mendaftar nama samaran di samping perubahan yang ditimbulkan umpan balik mereka, dan hanya untuk pemberi umpan balik yang memilih ikut serta [5]. Karena setiap baris dataset menunjuk kembali ke butir umpan balik, dan setiap butir umpan balik ke pemberinya, kredit dihitung dari catatan, bukan diingat-ingat dengan tangan; dan karena dataset sendiri tidak memuat data pribadi, kredit dapat ditarik tanpa menyentuh data pelatihan.

7.7 Latih dan Ukur: Sebelum dan Sesudah

7.7.1 Himpunan uji terpisah

Bagaimana kita tahu dataset membuat model lebih baik? Kita mengajukan pertanyaan dan memeriksa jawabannya, sebelum dan sesudah. Pertanyaan-pertanyaan itu tidak boleh sama dengan *prompt* pelatihan; jika sama, kita hanya memeriksa bahwa model menghafalnya. Himpunan pertanyaan yang disisihkan khusus untuk pengukuran disebut himpunan uji terpisah (*held-out set*). Di lab, "model"-nya adalah bank jawaban kecil yang mencari *prompt* terdekat menurut kesamaan kata, dan himpunan uji terpisahnya terdiri atas delapan pertanyaan pelanggan yang dirumuskan ulang, seperti "What hours are you open?" atau "Is it safe to send a silk dress?". Setiap pertanyaan memiliki kata-kata yang *wajib* ada dalam jawabannya (misalnya "07:00" dan "21:00") dan kata-kata yang *tidak boleh* ada (misalnya "2 days" untuk layanan ekspres).

7.7.2 Tingkat kelolosan dan regresi

Misalkan E adalah himpunan uji terpisah, dan $\text{lolos}(e) = 1$ bila jawaban atas pertanyaan e memenuhi pemeriksaannya serta 0 bila tidak. *Tingkat kelolosan* (*pass rate*) adalah

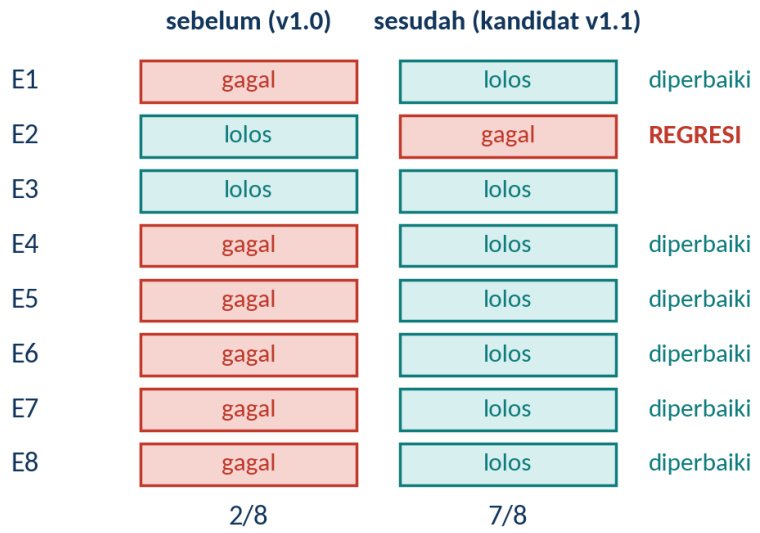
$$p = \frac{1}{|E|} \sum_{e \in E} \text{lolos}\,(e). \tag{7.3}$$

Membandingkan p_{sebelum} dan p_{sesudah} memberi tahu apakah kandidat lebih baik *secara rata-rata*. Itu belum cukup. Kita juga ingin tahu pertanyaan mana yang menjadi *lebih buruk*. Himpunan *regresi* adalah

$$R = \{\, e \in E : \text{lolos}_{\text{sebelum}}(e) = 1 \text{ dan } \text{lolos}_{\text{sesudah}}(e) = 0 \,\}. \tag{7.4}$$

Sebuah kandidat dapat menaikkan tingkat kelolosan tetapi tetap memiliki regresi: ia memperbaiki enam hal dan merusak satu. Apakah pertukaran itu dapat diterima adalah soal pertimbangan, dan karena itu *pipeline* melaporkan R secara eksplisit dan menyerahkan keputusannya kepada manusia.

Gambar 7.3 memperlihatkan hasil lab, diambil dari eksekusi nyata di Subbab 7.10: tingkat kelolosan naik dari 2 dari 8 menjadi 7 dari 8, dan satu pertanyaan mengalami regresi.



Gambar 7.3 Pertanyaan uji terpisah lab sebelum dan sesudah dataset. Enam pertanyaan diperbaiki, satu tetap lolos, dan E2 mengalami regresi: tingkat kelolosan naik dari 2/8 menjadi 7/8, tetapi *pipeline* tetap menyarankan penahanan.

7.7.3 Mengukur dengan jujur

Tiga peringatan menjaga pengukuran tetap jujur.

- **Himpunan kecil itu kasar.** Dengan delapan pertanyaan, setiap pertanyaan menggeser p sebesar $1/8 = 12.5$ poin persentase, sama seperti enam kasus tolok ukur di Bab 6 menggeser tingkat kemenangan gerbang sebesar 16,7 poin. Himpunan uji yang sungguhan harus tumbuh bersama setiap butir yang diterima.
- **Ukuran yang dijadikan sasaran berhenti mengukur.** Pengamatan ini, biasanya disebut hukum Goodhart dalam rumusan Strathern, "ketika sebuah ukuran menjadi sasaran, ia berhenti menjadi ukuran yang baik" [11], memperingatkan agar model tidak disetel sampai lolos pertanyaan-pertanyaan uji terpisah: pada saat itu pertanyaan tersebut tidak lagi terpisah. Terus tambahkan pertanyaan baru, dan jangan pernah melatih dengan himpunan evaluasi.
- **Siklus umpan balik dapat memberi makan dirinya sendiri.** Model yang dilatih dengan data yang dibentuk oleh keluarannya sendiri di masa lalu dapat bergeser dengan cara yang tidak terungkap oleh satu perubahan saja; ini salah satu bentuk klasik utang teknis tersembunyi dalam sistem pembelajaran mesin [12]. Mencatat asal setiap butir dataset, seperti yang dilakukan kolom `id` di lab, adalah pertahanan pertama.

7.7.4 Seberapa presisi sebuah tingkat kelolosan?

Tingkat kelolosan yang diukur pada n pertanyaan adalah perkiraan tentang bagaimana kandidat akan bekerja pada semua pertanyaan yang mungkin diajukan pengguna. Ukuran kasar yang lazim untuk ketidakpastiannya adalah *galat baku* (*standard error*) sebuah proporsi,

$$\mathrm{SE}(p) = \sqrt{\frac{p\,(1-p)}{n}}. \tag{7.5}$$

CONTOH. Untuk kandidat di lab, $p = 7/8 = 0.875$ dan $n = 8$, sehingga $\mathrm{SE} = \sqrt{0.875 \times 0.125/8} = \sqrt{0.01367} \approx 0.117$, sekitar 12 poin persentase. Selang kasar selebar dua galat baku, 0.875 ± 0.23, membentang dari sekitar 64% sampai melewati 100%: delapan pertanyaan menyatakan "jelas lebih baik daripada 25%" tetapi tidak dapat membedakan 70%

> dari 95%. Dengan $n = 100$ pertanyaan dan p yang sama, SE $= \sqrt{0.875 \times 0.125/100} \approx 0.033$, sekitar 3 poin, yang cukup presisi untuk membandingkan dua kandidat yang berselisih 10 poin.

Persamaan (7.5) hanyalah pedoman kasar (untuk n yang sangat kecil atau p yang mendekati 0 atau 1 ada selang yang lebih baik), tetapi ia memberi naluri yang tepat: perbesar himpunan uji terpisah seiring bertumbuhnya dataset, dan jangan percaya pada selisih yang lebih kecil dari kira-kira dua galat baku. Ini juga menjelaskan mengapa *pipeline* melaporkan hasil setiap pertanyaan dan bukan hanya persentasenya: dengan sedikit pertanyaan, hasil per pertanyaan membawa informasi lebih banyak daripada rata-ratanya.

7.8 Rilis: *Pipeline* Memberi Saran, Manusia Memutuskan

Langkah terakhir mengubah hasil pengukuran menjadi keputusan. *Pipeline* di lab memberi saran menurut aturan sederhana:

$$\text{saran} = \begin{cases} \text{rilis} & \text{jika } p_{\text{sesudah}} > p_{\text{sebelum}} \text{ dan } R = \emptyset, \\ \text{tahan} & \text{selainnya.} \end{cases} \tag{7.6}$$

Persamaan (7.6) sengaja dibuat hati-hati: regresi apa pun berarti “tahan”. Tetapi ia hanyalah saran. Seseorang boleh menyetujui rilis walaupun ada regresi, misalnya ketika jawaban yang mengalami regresi bersifat kecil sementara yang diperbaiki adalah masalah keselamatan, atau menahan rilis walaupun hasilnya bersih, misalnya karena pekan libur panjang adalah saat yang buruk untuk mengubah jawaban yang dilihat pelanggan. Apa pun keputusannya, keputusan dan alasannya dicatat, dan para pemberi umpan balik membaca hasilnya: “Shipped in v1.1” atau “Not taken in v1.1”, dengan alasannya.

Prinsip yang sama mengatur rilis perangkat lunak Profitina sendiri (Bab 9): gerbang rilis menjalankan setiap uji otomatis dan melaporkan keberhasilan hanya ketika versi baru menjawab pemeriksaan kesehatan, tetapi seorang manusia yang memulai rilis. Mesin sangat baik dalam memeriksa; manusia bertanggung jawab untuk memutuskan.

Seperti apa keputusan rilis itu seharusnya tampak bagi para pemberi umpan balik? Catatan rilis yang baik itu pendek, konkret, dan memakai bahasa mereka. Untuk eksekusi kedua lab, catatannya bisa berbunyi: “v1.1 tidak dirilis. Versi ini memperbaiki enam jawaban (jam buka, waktu dan harga layanan ekspres, jarak penjemputan, perawatan sutra dan wol, QRIS) tetapi membuat satu jawaban lebih buruk: ia tidak lagi menyebut bahwa toko buka setiap hari. Kami menunggu koreksi yang mempertahankan keduanya (F11). Umpan balik Anda yang sudah diterima tetap berada di dataset dan akan menjadi bagian dari percobaan berikutnya.” Tidak ada bagian catatan itu yang menuntut pembaca memahami apa itu tingkat kelolosan, tetapi ia menyebut hasil pengukuran, regresinya, keputusannya, dan langkah berikutnya. Menulis catatan semacam itu adalah bagian dari keputusan manusia, bukan tambahan belakangan, karena dari catatan itulah orang-orang yang membantu mengetahui bahwa bantuan mereka ditanggapi dengan sungguh-sungguh.

Mencatat alasan bukan formalitas. Tiga bulan kemudian, ketika seseorang bertanya mengapa v1.1 ditahan atau mengapa v1.2 dirilis padahal ada satu regresi, catatan itulah satu-satunya jawaban yang dapat dipercaya. Catatan keputusan juga menjadi bahan belajar bagi tim: bila keputusan “setujui walau ada regresi kecil” berulang kali ternyata tepat, aturan saran dapat disesuaikan; bila

berulang kali ternyata keliru, aturan itu justru perlu diperketat. Dengan begitu manusia dan mesin sama-sama belajar dari siklus yang sama.

7.9 PROFIT untuk Memperbaiki Jawaban AI

7.9.1 Mengapa memberi hadiah untuk umpan balik

Umpan balik menuntut usaha, dan kebanyakan platform membalasnya hanya dengan ucapan terima kasih. Gagasan pendiri Profitina adalah bahwa kontribusi yang bermanfaat, termasuk umpan balik yang memperbaiki model lokal, layak mendapat hadiah yang ada isinya. Hadiah itu adalah PROFIT, koin utilitas platform: token SPL di jaringan Solana dengan 6 angka desimal, yang kontraknya sudah berjalan di Solana *devnet* per September 2026 [5], [13]. Bab 12 menjelaskan PROFIT secara lengkap. Untuk bab ini ada empat fakta penting, dan semuanya harus dinyatakan dengan gamblang:

- Profitina menggambarkan PROFIT sebagai koin utilitas, "bukan sekuritas, bukan uang elektronik, bukan janji imbal hasil", dan di *devnet* ia tidak memiliki nilai moneter yang nyata [5].
- Menjual atau menukarkan PROFIT dengan uang dinonaktifkan sampai *mainnet* dan kejelasan regulasi; *mainnet* adalah butir peta jalan yang digerbangi perizinan dan audit [5].
- Yang sudah berjalan: memperoleh sejumlah kecil PROFIT untuk aktivitas yang bermanfaat, membelanjakan PROFIT untuk layanan di dalam platform, dan transfer antarpengguna [5].
- Tidak ada isi bab ini yang merupakan nasihat investasi, dan tidak ada harga atau nilai yang dijanjikan.

7.9.2 Aturan perolehan yang publik

Tabel 7.2 menyalin tabel perolehan publik dari *whitepaper* September 2026 [5]. Jumlahnya dalam PROFIT; batas harian adalah banyaknya tindakan yang diberi hadiah per hari.

Tabel 7.2 Aturan perolehan PROFIT yang publik (*whitepaper* v24, September 2026). Jumlahnya dikalikan lagi dengan pengali emisi yang menurun dan dengan penilaian kebermanfaatan oleh AI.

Tindakan	Hadiah (PROFIT)	Batas harian
Umpan balik atas hasil pindai	0,01–0,05	10
Unggahan	0,02	1
Balasan yang diterima	0,01	2
Repost yang diterima	0,01	1
Suka yang diterima	0,01	5
Membuat kanal	0,05	2
Rujukan (kedua pihak, sekali per pasangan)	0,25	50

Di sekitar tabel itu ada aturan-aturan yang menjaganya tetap adil [5]:

- **Jumlah kecil.** Satuan dasarnya 0,01 PROFIT. Hadiah adalah ucapan terima kasih, bukan upah.
- **Batas harian yang diatur ulang pukul 00.00 WIB.** WIB adalah Waktu Indonesia Barat, UTC+7, zona waktu Jakarta dan Surabaya. Batas harian mencegah satu akun memanen hadiah dengan pengulangan.
- **Hanya manusia terverifikasi yang memperoleh hadiah.** Akun baru memperoleh hadiah setelah alamat surelnya diverifikasi, lapisan pertama melawan bot.

- **Kebermanfaatan, bukan volume.** Penilaian kebermanfaatan oleh AI lokal memberi kontribusi yang remeh atau mirip bot nilai 0,00; hadiah atas aktivitas mencurigakan ditahan dalam *escrow* dan dapat ditarik kembali lalu dibakar.
- **Duplikat tidak memperoleh apa-apa.** Konten duplikat memperoleh 0,00 dan ditautkan ke butir sebelumnya.
- **Emisi yang menurun.** Setiap jumlah dikalikan dengan pengali emisi yang menjadi separuh setiap kali jumlah pengguna manusia bertambah sepuluh kali lipat, sehingga hadiah mengecil seiring bertumbuhnya komunitas (Bab 12).

Whitepaper menerbitkan aturan dan hasilnya, bukan ambang-ambang internal yang menerapkannya, dan buku ini mengikuti garis yang sama.

7.9.3 Aritmetika hadiah yang dibatasi

Lab menerapkan aturan hadiah dengan bentuk yang sama, dengan nilai-nilainya sendiri. Untuk butir i, misalkan B hadiah dasar, A bonus untuk butir yang diterima manusia ke dalam dataset, $\text{acc}_i = 1$ bila butir i diterima dan 0 bila tidak, serta $\text{elig}_i = 1$ bila butir itu memenuhi syarat (pemberinya terverifikasi, tidak terlalu pendek, bukan duplikat, dan masih di bawah batas harian). Maka

$$r_i = (B + A \cdot \text{acc}_i) \cdot \text{elig}_i. \tag{7.7}$$

Lab memakai $B = 0.01$ dan $A = 0.04$, sehingga butir yang diterima memperoleh 0.05, sesuai rentang publik 0,01–0,05 untuk umpan balik atas hasil pindai, dan memakai batas lab $K = 3$ butir berhadiah per pengguna per hari, lebih kecil daripada batas publik 10, agar batas itu terlihat dalam contoh yang pendek. Untuk setiap pengguna u dan setiap hari WIB d,

$$\sum_{i \in \text{butir}(u,d)} \text{dihadiahi}_i \le K. \tag{7.8}$$

Dua detail rekayasa layak mendapat perhatian tersendiri.

Bilangan bulat, bukan bilangan titik mengambang. PROFIT memiliki 6 desimal, sehingga satuan terkecilnya adalah sepersejuta PROFIT, sebuah *mikro-PROFIT* (µP). Lab menyimpan setiap jumlah sebagai bilangan bulat dalam satuan mikro,

$$\text{jumlah}_{\mu\text{P}} = \text{jumlah}_{\text{PROFIT}} \times 10^6, \tag{7.9}$$

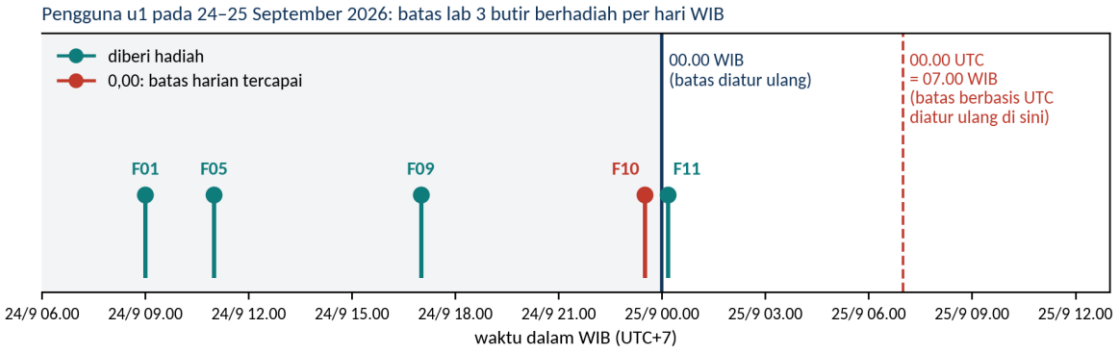
sehingga 0.01 PROFIT disimpan sebagai 10,000 dan 0.05 sebagai 50,000. Aritmetika bilangan bulat itu eksak; aritmetika titik mengambang tidak ($0.1 + 0.2$ tidak tepat sama dengan 0.3 dalam titik mengambang biner), sedangkan buku besar harus berjumlah tepat. Gagasan yang sama mendasari kolom `decimals` milik token itu sendiri di Solana [13].

Hari apa "hari ini"? Batas harian memerlukan definisi hari. Server biasanya menyimpan waktu dalam UTC, tetapi pengguna Profitina hidup dalam WIB, dan batas diatur ulang pada tengah malam WIB. Tanggal WIB dari suatu saat t adalah tanggal UTC dari t yang digeser tujuh jam:

$$\text{hari}_{\text{WIB}}(t) = \text{tanggal}(t_{\text{UTC}} + 7\,\text{jam}). \tag{7.10}$$

Program yang mengaitkan batas pada tanggal UTC akan mengatur ulang batas itu pada pukul 07.00 WIB, di tengah pagi, dan ini adalah *bug* nyata yang ditemukan dan diperbaiki tim Profitina dalam kode kuotanya sendiri. Karena Indonesia tidak memakai waktu musim panas, selisih tetap

tujuh jam berlaku tepat sepanjang tahun. Indonesia memiliki tiga zona waktu, yaitu WIB (UTC+7), WITA (UTC+8), dan WIT (UTC+9), sehingga pengguna di Makassar atau Jayapura hidup satu atau dua jam lebih awal daripada Jakarta. Walau begitu, platform tetap harus memilih satu definisi "hari ini" untuk jatah yang berlaku bagi semua orang, dan Profitina mengumumkan pilihannya: semua kuota dan batas harian diatur ulang pukul 00.00 WIB. Mengumumkan aturan itu sama pentingnya dengan memilihnya, karena pengguna di Jayapura yang tahu bahwa pengaturan ulang terjadi pukul 02.00 waktu setempat tidak akan terkejut. Bagi pembaca yang menulis perangkat lunak untuk pasar Indonesia, pelajaran ini berlaku luas: setiap angka yang berbunyi "per hari", "per minggu", atau "per bulan" harus disertai zona waktunya, baik di dalam kode maupun di dalam teks yang dibaca pengguna. **Gambar 7.4** memperlihatkan pengguna lab u1 pada 24–25 September 2026: tiga butir berhadiah, butir keempat pukul 23.30 WIB yang terkena batas, dan butir kelima pukul 00.10 WIB yang kembali berhadiah karena hari WIB yang baru telah dimulai.



Gambar 7.4 Batas harian lab sebanyak 3 butir berhadiah per hari WIB untuk pengguna u1. F10 pukul 23.30 WIB terkena batas; F11 pukul 00.10 WIB jatuh pada hari WIB yang baru. Batas yang dikaitkan pada tanggal UTC baru akan diatur ulang pukul 07.00 WIB.

7.9.4 Membatasi apa yang dapat diperoleh satu akun

Batas harian tidak hanya menjaga hadiah tetap kecil; ia membuat kasus terburuk dapat *dihitung*. Jika tindakan j membayar paling banyak v_j dan boleh diberi hadiah paling banyak c_j kali sehari, maka tidak ada satu akun pun yang dapat memperoleh lebih dari

$$M_{\text{hari}} = \sum_j v_j \; c_j, \tag{7.11}$$

dalam sehari, sebelum pengali emisi dan penilaian kebermanfaatan, yang keduanya hanya dapat mengurangi jumlah itu.

> **CONTOH.** Terapkan Persamaan (7.11) pada tindakan harian di **Tabel 7.2**, tanpa rujukan yang hanya sekali: umpan balik atas hasil pindai $0.05 \times 10 = 0.50$, unggahan $0.02 \times 1 = 0.02$, balasan yang diterima $0.01 \times 2 = 0.02$, *repost* yang diterima $0.01 \times 1 = 0.01$, suka yang diterima $0.01 \times 5 = 0.05$, dan kanal yang dibuat $0.05 \times 2 = 0.10$. Jumlahnya $M_{\text{hari}} = 0.70$ PROFIT, dan dalam praktik jauh lebih kecil, karena jumlah maksimum mensyaratkan kontribusi yang diterima dan bermanfaat, dan pengali emisi menurun seiring bertumbuhnya komunitas.

Batas per akun belum merupakan batas per orang, karena satu orang dapat membuat banyak akun; ini disebut *serangan Sybil*. Karena itulah batas per akun dipadukan dengan verifikasi surel, deteksi duplikat lintas akun, penilaian kebermanfaatan, serta *escrow* dengan penarikan kembali untuk pola yang mencurigakan. Tidak satu pun pertahanan itu sempurna sendirian; bersama-sama, dan dengan Persamaan (7.11) yang menjaga kasus terburuk setiap akun tetap kecil,

semuanya membuat pemanenan hadiah tidak menguntungkan. Bab 5 menguraikan sisi Fair-Play dari pertahanan yang sama.

7.10 Lab: *Pipeline* Umpan Balik Mini

7.10.1 Skenario dan data

Profitina Laundry memiliki layanan bantuan kecil yang menjawab pertanyaan pelanggan. Untuk lab ini "model"-nya sengaja dibuat sederhana: sebuah *bank jawaban* berisi enam *prompt* yang dikenal, B1 sampai B6, masing-masing dengan satu jawaban, dan aturan yang menjawab pertanyaan apa pun dengan jawaban milik *prompt* terdekat menurut kesamaan kata (Persamaan (7.1)). Beberapa jawabannya salah atau tidak lengkap: B1 hanya berkata "We are open every day." tanpa jam, B3 menyatakan layanan ekspres memakan 2 hari (seharusnya 6 jam), B4 hanya berkata "Yes." untuk pertanyaan tentang penjemputan, B5 menyatakan sutra dan wol dicuci di mesin (seharusnya harus *dry-clean*), dan B6 menyatakan binatu hanya menerima tunai (ia juga menerima QRIS). Jawaban-jawaban itu berbahasa Inggris karena keluarannya dicetak apa adanya oleh program.

Para pelanggan telah mengirim sebelas butir umpan balik. Semuanya disimpan dalam `submissions.tsv`, berkas yang dipisahkan tab dengan satu butir per baris; **Tabel 7.3** menampilkannya. Semua pengguna, waktu, dan teksnya adalah rekaan untuk lab ini.

Tabel 7.3 Sebelas butir umpan balik lab dalam `submissions.tsv`. Waktu dalam UTC; tambahkan tujuh jam untuk WIB.

Butir	Pengguna	Terverifikasi	Waktu (UTC)	Jawaban	Usulan jawaban	Komentar
F01	u1	yes	24 Sep 02:00	B3	Express service is ready in 6 hours and costs Rp12,000 per kilogram.	The express answer is wrong, it is 6 hours.
F02	u2	yes	24 Sep 02:40	B3	Express is ready in 6 hours, costing Rp12,000 per kilogram.	Wrong: express is not 2 days.
F03	u3	yes	24 Sep 03:10	B1	We are open from 07:00 to 21:00.	The hours are missing.
F04	u4	yes	24 Sep 03:30	B5	Silk and wool are dry-clean only; a partner cleans them in 4 days.	Unsafe advice: silk would be damaged in a machine.
F05	u1	yes	24 Sep 04:00	B6	We accept cash and QRIS (DANA, GoPay, OVO, ShopeePay, mobile banking).	Wrong, QRIS works at the counter.
F06	u5	no	24 Sep 05:00	B4	Yes, free pickup and delivery within 3 km.	Missing the distance.
F07	u6	yes	24 Sep 05:30	B4	ok	ok
F08	u2	yes	24 Sep 06:00	B2	Regular washing is Rp5,000 per kilogram.	Should be cheaper.
F09	u1	yes	24 Sep 10:00	B2	Regular washing costs Rp7,000 per kilogram and takes 2 days.	Just checking this one.
F10	u1	yes	24 Sep 16:30	B4	We offer free pickup and delivery within 3 km of the shop.	Missing the distance.
F11	u1	yes	24 Sep 17:10	B1	We are open every day from 07:00 to 21:00.	Missing hours, keep every day.

Keputusan peninjau ada di `decisions.tsv`: terima F01, F03, F04, F05, F06, dan F10; tolak F08 (“contradicts the price list (Rp7,000)”) dan F09 (“same as the current answer”). Peninjau belum melihat F11. Perhatikan beberapa jebakan yang sengaja ditanam di data: F02 mengulang F01, F07 hanya berisi “ok”, F06 berasal dari akun yang belum terverifikasi, F08 salah, pengguna u1 mengirim lima butir dalam sehari, dan F11, yang datang terakhir, diam-diam memuat detail yang dilupakan F03.

Himpunan uji terpisah, E1 sampai E8, ditulis di dalam program; setiap pertanyaan menyebut kata-kata yang wajib dan yang tidak boleh ada dalam jawabannya.

7.10.2 Programnya

Kode 7.1 adalah seluruh *pipeline* dalam Python, hanya dengan pustaka standar.

Kode 7.1 `feedback_pipeline.py`: kirim, kurasi dengan keputusan manusia, ekspor dataset JSONL, beri hadiah di bawah batas harian WIB, ukur sebelum dan sesudah, lalu tunggu keputusan rilis dari manusia.

```
"""A mini feedback pipeline: submit -> curate -> dataset -> measure -> release.

    python feedback_pipeline.py [--decisions FILE] [--release approve|hold REASON]

Reads submissions.tsv and decisions.tsv (or FILE), writes dataset.jsonl.
"""
import json
import re
import sys
from datetime import datetime, timedelta, timezone

WIB = timezone(timedelta(hours=7))  # Indonesia has no daylight saving
MICRO = 1_000_000                   # amounts are kept in integer micro-units
BASE, BONUS, DAILY_CAP = 10_000, 40_000, 3  # this lab's own values
STOP = set("a an and are can do does for from how i in is it my of on the to "
           "we what when which will with you your".split())

MODEL_V1 = {  # the "model" before feedback: prompt -> answer
    "B1": ("what are your opening hours", "We are open every day."),
    "B2": ("how much does regular washing cost per kilogram",
           "Regular washing costs Rp7,000 per kilogram and takes 2 days."),
    "B3": ("how long does express service take", "Express service takes 2 days."),
    "B4": ("do you pick up and deliver laundry", "Yes."),
    "B5": ("can you wash silk and wool clothes",
           "Yes, we wash everything in our machines."),
    "B6": ("which payment methods do you accept", "We accept cash only."),
}
EVAL = [  # held-out questions, never used for "training"
    ("E1", "What hours are you open?", ["07:00", "21:00"], []),
    ("E2", "Are your opening hours the same every day?", ["every day"], []),
    ("E3", "Price per kilo for regular washing?", ["Rp7,000"], []),
    ("E4", "How fast is the express service?", ["6 hours"], ["2 days"]),
    ("E5", "What does the express service cost?", ["Rp12,000"], []),
    ("E6", "Will you pick up laundry from my house?", ["3 km"], []),
    ("E7", "Is it safe to send a silk dress?", ["dry-clean"], ["everything"]),
    ("E8", "Can I pay with QRIS?", ["QRIS"], []),
]


def read_tsv(path):
    with open(path, encoding="utf-8") as f:
        header, *rows = [ln.rstrip("\n").split("\t") for ln in f if ln.strip()]
    return [dict(zip(header, r)) for r in rows]
```

Kode 7.1 (Lanjutan) *feedback_pipeline.py: kirim, kurasi dengan keputusan manusia, ekspor dataset JSONL, beri hadiah di bawah batas harian WIB, ukur sebelum dan sesudah, lalu tunggu keputusan rilis dari manusia.*

```
def words(text):
    return set(re.findall(r"[a-z0-9:]+", text.lower().replace(",", ""))) - STOP


def jaccard(a, b):
    a, b = words(a), words(b)
    return len(a & b) / len(a | b) if a | b else 0.0


def answer(model, question):
    """Toy model: the answer of the closest known prompt (word overlap)."""
    best = max(model, key=lambda k: (jaccard(question, model[k][0]), k))
    return model[best][1]


def curate(items, decisions):
    """Sort each item, group duplicates, then apply the human decisions."""
    dataset = []
    for it in items:
        c = it["comment"].lower()
        it["category"] = ("unsafe advice" if re.search(r"unsafe|damage", c) else
                          "wrong fact" if "wrong" in c else
                          "missing info" if "missing" in c else "other")
        it["severity"] = {"unsafe advice": "high", "wrong fact": "medium"}.get(
            it["category"], "low")
        earlier = [e for e in items if e is not it and e["target"] == it["target"]
                   and e["utc_time"] < it["utc_time"]
                   and jaccard(e["suggestion"], it["suggestion"]) >= 0.7]
        if len(it["suggestion"].split()) < 4:
            it["status"] = "not taken: too short (automatic)"
        elif earlier:
            it["status"] = f"duplicate of {earlier[0]['id']} (automatic)"
        elif it["id"] not in decisions:
            it["status"] = "awaiting a human decision"
        elif decisions[it["id"]]["decision"] == "accept":
            dataset.append(it)
            it["status"] = f"in dataset #{len(dataset)}"
        else:
            it["status"] = "not taken: " + decisions[it["id"]]["reason"]
    return dataset
```

Kode 7.1 (Lanjutan) *feedback_pipeline.py: kirim, kurasi dengan keputusan manusia, ekspor dataset JSONL, beri hadiah di bawah batas harian WIB, ukur sebelum dan sesudah, lalu tunggu keputusan rilis dari manusia.*

```
def rewards(items):
    """Base reward for a useful submission, bonus when a human accepts it."""
    per_day = {}
    for it in items:
        t = datetime.strptime(it["utc_time"], "%Y-%m-%dT%H:%MZ")
        local = t.replace(tzinfo=timezone.utc).astimezone(WIB)
        key = (it["user"], local.date())
        it["wib"] = local.strftime("%d %b %H:%M")
        if it["verified"] != "yes":
            it["reward"], it["note"] = 0, "held: email not verified"
        elif it["status"].startswith(("not taken: too", "duplicate")):
            it["reward"], it["note"] = 0, it["status"].split(" (")[0]
        elif per_day.get(key, 0) >= DAILY_CAP:
            it["reward"], it["note"] = 0, "daily cap reached (resets 00:00 WIB)"
        else:
            per_day[key] = per_day.get(key, 0) + 1
            bonus = BONUS if it["status"].startswith("in dataset") else 0
            it["reward"] = BASE + bonus
            it["note"] = "base + accepted bonus" if bonus else "base"


def measure(model):
    results = {}
    for eid, q, must, must_not in EVAL:
        a = answer(model, q)
        results[eid] = (all(m in a for m in must)
                        and not any(m in a for m in must_not))
    return results


def main(argv):
    items = sorted(read_tsv("submissions.tsv"), key=lambda r: r["utc_time"])
    path = argv[argv.index("--decisions") + 1] if "--decisions" in argv \
        else "decisions.tsv"
    decisions = {d["id"]: d for d in read_tsv(path)}

    print("1-2. SUBMIT and CURATE")
    dataset = curate(items, decisions)
    rewards(items)
    for it in items:
        print(f"  {it['id']} {it['user']} {it['target']} {it['category']:<13} "
              f"{it['severity']:<6} {it['status']}")
```

Kode 7.1 (Lanjutan) *feedback_pipeline.py: kirim, kurasi dengan keputusan manusia, ekspor dataset JSONL, beri hadiah di bawah batas harian WIB, ukur sebelum dan sesudah, lalu tunggu keputusan rilis dari manusia.*

```
    print("PROFIT earned (lab values, integer micro-units shown as PROFIT)")
    for it in items:
        print(f"  {it['id']} {it['user']} {it['wib']} WIB  "
              f"{it['reward'] / MICRO:.2f}  {it['note']}")
    total = sum(it["reward"] for it in items)
    print(f"  total emitted: {total / MICRO:.2f} PROFIT")

    print("3. DATASET -> dataset.jsonl")
    with open("dataset.jsonl", "w", encoding="utf-8", newline="\n") as f:
        for n, it in enumerate(dataset, 1):
            row = {"n": n, "id": it["id"], "prompt": MODEL_V1[it["target"]][0],
                   "rejected": MODEL_V1[it["target"]][1],
                   "chosen": it["suggestion"], "category": it["category"]}
            f.write(json.dumps(row) + "\n")
    print(f"  {len(dataset)} preference pairs written")

    print("4. TRAIN and MEASURE (held-out questions)")
    model_v2 = dict(MODEL_V1)
    for it in dataset:  # "training" here = updating the answer bank
        model_v2[it["target"]] = (MODEL_V1[it["target"]][0], it["suggestion"])
    before, after = measure(MODEL_V1), measure(model_v2)
    for eid, *_ in EVAL:
        mark = {(False, True): "fixed", (True, False): "REGRESSION"}.get(
            (before[eid], after[eid]), "")
        print(f"  {eid}  before {'pass' if before[eid] else 'fail'}  "
              f"after {'pass' if after[eid] else 'fail'}  {mark}".rstrip())
    b, a = sum(before.values()), sum(after.values())
    regressions = [e for e in before if before[e] and not after[e]]
    print(f"  pass rate: {b}/{len(EVAL)} -> {a}/{len(EVAL)}"
          f" ({100 * b / len(EVAL):.1f}% -> {100 * a / len(EVAL):.1f}%)")

    print("5. RELEASE")
    advice = ("release" if a > b and not regressions else
              "hold (regression on " + ", ".join(regressions) + ")"
              if regressions else "hold (no improvement)")
    print(f"  the pipeline advises: {advice}")
    if "--release" not in argv:
        print("  waiting for a human: run again with --release approve"
              " or --release hold REASON")
    elif argv[argv.index("--release") + 1] == "approve":
        print("  human decision: approved -> Shipped in v1.1")
    else:
        reason = " ".join(argv[argv.index("--release") + 2:]) or "no reason"
        print(f"  human decision: Not taken in v1.1 ({reason})")


if __name__ == "__main__":
    main(sys.argv[1:])
```

Telusuri program itu menurut urutan kelima langkah.

- **Konstanta.** `WIB` adalah zona waktu tetap UTC+7; `zoneinfo.ZoneInfo("Asia/Jakarta")` milik Python juga dapat dipakai, tetapi di Windows ia memerlukan paket tambahan `tzdata`, sedangkan selisih tetap itu tepat untuk Indonesia. `MICRO`, `BASE`, `BONUS`, dan `DAILY_CAP` menerapkan Persamaan (7.9), Persamaan (7.7), dan Persamaan (7.8) dengan nilai-nilai lab ini. `STOP` memuat kata-kata yang sangat umum yang dibuang oleh `words`.

- **Model mainan.** `MODEL_V1` memetakan B1–B6 ke sebuah *prompt* dan sebuah jawaban. `answer` memilih *prompt* yang dikenal dengan kesamaan Jaccard tertinggi terhadap pertanyaan; bila seri, kunci yang lebih besar yang menang, sehingga hasilnya tidak pernah bergantung pada urutan *dictionary*.
- **Kirim.** `read_tsv` membaca berkas yang dipisahkan tab menjadi daftar *dictionary*, satu per baris, dengan baris pertama sebagai kepala. `main` mengurutkan butir-butir menurut waktu, karena urutan kedatangan penting bagi duplikat dan batas harian.
- **Kurasi.** `curate` memberi setiap butir kategori dan tingkat keparahan dari komentarnya, lalu memeriksa secara berurutan: terlalu pendek (kurang dari 4 kata, otomatis), hampir kembar dengan butir sebelumnya tentang jawaban yang sama ($J \geq 0.7$, otomatis), belum ada keputusan manusia, diterima (butir masuk dataset dan mendapat nomor), atau ditolak (dengan alasan dari peninjau).
- **Hadiah.** `rewards` mengubah setiap waktu UTC menjadi WIB dengan `astimezone`, menghitung butir berhadiah per pengguna dan tanggal WIB, lalu menerapkan Persamaan (7.7) dan Persamaan (7.8). Akun yang belum terverifikasi ditahan pada 0,00, demikian pula duplikat dan butir yang terlalu pendek, masing-masing dengan alasannya.
- **Dataset.** Setiap butir yang diterima menjadi satu objek JSON pada satu baris `dataset.jsonl`, dengan `newline="\n"` agar berkasnya identik di Windows, macOS, dan Linux.
- **Latih dan ukur.** "Pelatihan" di sini jujur mengaku sebagai mainan: ia mengganti jawaban setiap *prompt* yang dikoreksi dengan jawaban *chosen*. `measure` menerapkan Persamaan (7.3) pada kedua versi, dan perulangannya mencetak hasil setiap pertanyaan sambil menandai perbaikan dan regresi seperti pada Persamaan (7.4).
- **Rilis.** Sarannya mengikuti Persamaan (7.6). Tanpa `--release` program berhenti dan menunggu seseorang; dengan `--release approve` atau `--release hold REASON` program mencatat keputusan manusia itu.

Kode 7.2 adalah *pipeline* yang sama dalam Java.

Kode 7.2 `FeedbackPipeline.java`: *pipeline* yang sama dalam Java, dengan *record*, `java.time` untuk WIB, satuan mikro berupa bilangan bulat, dan penyusun JSON Lines buatan sendiri.

```java
import java.io.IOException;
import java.io.PrintWriter;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** A mini feedback pipeline: submit -> curate -> dataset -> measure -> release.
 *  java FeedbackPipeline [--decisions FILE] [--release approve|hold REASON] */
public class FeedbackPipeline {
    static final ZoneOffset WIB = ZoneOffset.ofHours(7);   // no daylight saving
    static final long MICRO = 1_000_000;                    // integer micro-units
    static final long BASE = 10_000, BONUS = 40_000;        // this lab's own values
    static final int DAILY_CAP = 3;
    static final Set<String> STOP = Set.of(("a an and are can do does for from"
            + " how i in is it my of on the to we what when which will with you"
            + " your").split(" "));

    record Q(String prompt, String answer) {}
    record Eval(String id, String question, List<String> must,
                List<String> mustNot) {}
```

Kode 7.2 (Lanjutan) *FeedbackPipeline.java: pipeline yang sama dalam Java, dengan record, java.time untuk WIB, satuan mikro berupa bilangan bulat, dan penyusun JSON Lines buatan sendiri.*

```
    static final Map<String, Q> MODEL_V1 = new LinkedHashMap<>();
    static {                                  // the "model" before feedback
        MODEL_V1.put("B1", new Q("what are your opening hours",
                "We are open every day."));
        MODEL_V1.put("B2", new Q("how much does regular washing cost per kilogram",
                "Regular washing costs Rp7,000 per kilogram and takes 2 days."));
        MODEL_V1.put("B3", new Q("how long does express service take",
                "Express service takes 2 days."));
        MODEL_V1.put("B4", new Q("do you pick up and deliver laundry", "Yes."));
        MODEL_V1.put("B5", new Q("can you wash silk and wool clothes",
                "Yes, we wash everything in our machines."));
        MODEL_V1.put("B6", new Q("which payment methods do you accept",
                "We accept cash only."));
    }

    static final List<Eval> EVAL = List.of(    // held-out, never "trained" on
        new Eval("E1", "What hours are you open?", List.of("07:00", "21:00"),
                 List.of()),
        new Eval("E2", "Are your opening hours the same every day?",
                 List.of("every day"), List.of()),
        new Eval("E3", "Price per kilo for regular washing?", List.of("Rp7,000"),
                 List.of()),
        new Eval("E4", "How fast is the express service?", List.of("6 hours"),
                 List.of("2 days")),
        new Eval("E5", "What does the express service cost?", List.of("Rp12,000"),
                 List.of()),
        new Eval("E6", "Will you pick up laundry from my house?", List.of("3 km"),
                 List.of()),
        new Eval("E7", "Is it safe to send a silk dress?", List.of("dry-clean"),
                 List.of("everything")),
        new Eval("E8", "Can I pay with QRIS?", List.of("QRIS"), List.of()));

    static List<Map<String, String>> readTsv(String path) throws IOException {
        List<String> lines = new ArrayList<>();
        for (String l : Files.readAllLines(Path.of(path), StandardCharsets.UTF_8)) {
            if (!l.isBlank()) lines.add(l);
        }
        String[] header = lines.get(0).split("\t");
        List<Map<String, String>> rows = new ArrayList<>();
        for (String l : lines.subList(1, lines.size())) {
            String[] cells = l.split("\t", -1);
            Map<String, String> row = new HashMap<>();
            for (int k = 0; k < header.length; k++) row.put(header[k], cells[k]);
            rows.add(row);
        }
        return rows;
```

Kode 7.2 (Lanjutan) *FeedbackPipeline.java: pipeline yang sama dalam Java, dengan record, java.time untuk WIB, satuan mikro berupa bilangan bulat, dan penyusun JSON Lines buatan sendiri.*

```
    }

    static Set<String> words(String text) {
        Set<String> out = new HashSet<>();
        Matcher m = Pattern.compile("[a-z0-9:]+")
                .matcher(text.toLowerCase().replace(",", ""));
        while (m.find()) out.add(m.group());
        out.removeAll(STOP);
        return out;
    }

    static double jaccard(String a, String b) {
        Set<String> x = words(a), y = words(b);
        Set<String> union = new HashSet<>(x);
        union.addAll(y);
        if (union.isEmpty()) return 0.0;
        x.retainAll(y);
        return (double) x.size() / union.size();
    }

    /** Toy model: the answer of the closest known prompt (word overlap). */
    static String answer(Map<String, Q> model, String question) {
        String best = null;
        double bestScore = -1;
        for (String k : model.keySet()) {      // ties go to the larger key
            double s = jaccard(question, model.get(k).prompt());
            if (s > bestScore || (s == bestScore && k.compareTo(best) > 0)) {
                best = k;
                bestScore = s;
            }
        }
        return model.get(best).answer();
    }

    static List<Map<String, String>> curate(List<Map<String, String>> items,
            Map<String, Map<String, String>> decisions) {
        List<Map<String, String>> dataset = new ArrayList<>();
        for (Map<String, String> it : items) {
            String c = it.get("comment").toLowerCase();
            String cat = Pattern.compile("unsafe|damage").matcher(c).find()
                    ? "unsafe advice" : c.contains("wrong") ? "wrong fact"
                    : c.contains("missing") ? "missing info" : "other";
            it.put("category", cat);
            it.put("severity", cat.equals("unsafe advice") ? "high"
                    : cat.equals("wrong fact") ? "medium" : "low");
            Map<String, String> earlier = null;
```

Kode 7.2 (Lanjutan) *FeedbackPipeline.java: pipeline yang sama dalam Java, dengan record, java.time untuk WIB, satuan mikro berupa bilangan bulat, dan penyusun JSON Lines buatan sendiri.*

```
            for (Map<String, String> e : items) {
                if (e != it && e.get("target").equals(it.get("target"))
                        && e.get("utc_time").compareTo(it.get("utc_time")) < 0
                        && jaccard(e.get("suggestion"),
                                   it.get("suggestion")) >= 0.7) {
                    earlier = e;
                    break;
                }
            }
            Map<String, String> d = decisions.get(it.get("id"));
            if (it.get("suggestion").strip().split("\\s+").length < 4) {
                it.put("status", "not taken: too short (automatic)");
            } else if (earlier != null) {
                it.put("status",
                       "duplicate of " + earlier.get("id") + " (automatic)");
            } else if (d == null) {
                it.put("status", "awaiting a human decision");
            } else if (d.get("decision").equals("accept")) {
                dataset.add(it);
                it.put("status", "in dataset #" + dataset.size());
            } else {
                it.put("status", "not taken: " + d.get("reason"));
            }
        }
        return dataset;
    }

    static void rewards(List<Map<String, String>> items) {
        Map<String, Integer> perDay = new HashMap<>();
        DateTimeFormatter in = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm'Z'");
        DateTimeFormatter out =
                DateTimeFormatter.ofPattern("dd MMM HH:mm", Locale.ENGLISH);
        for (Map<String, String> it : items) {
            var local = LocalDateTime.parse(it.get("utc_time"), in)
                    .atOffset(ZoneOffset.UTC).withOffsetSameInstant(WIB);
            String key = it.get("user") + "|" + local.toLocalDate();
            it.put("wib", local.format(out));
            String st = it.get("status");
            long reward = 0;
            String note;
            if (!it.get("verified").equals("yes")) {
                note = "held: email not verified";
            } else if (st.startsWith("not taken: too")
                       || st.startsWith("duplicate")) {
                note = st.split(" \\(")[0];
            } else if (perDay.getOrDefault(key, 0) >= DAILY_CAP) {
```

Kode 7.2 (Lanjutan) *FeedbackPipeline.java: pipeline yang sama dalam Java, dengan record, java.time untuk WIB, satuan mikro berupa bilangan bulat, dan penyusun JSON Lines buatan sendiri.*

```
                note = "daily cap reached (resets 00:00 WIB)";
            } else {
                perDay.merge(key, 1, Integer::sum);
                long bonus = st.startsWith("in dataset") ? BONUS : 0;
                reward = BASE + bonus;
                note = bonus > 0 ? "base + accepted bonus" : "base";
            }
            it.put("reward", Long.toString(reward));
            it.put("note", note);
        }
    }

    static Map<String, Boolean> measure(Map<String, Q> model) {
        Map<String, Boolean> results = new LinkedHashMap<>();
        for (Eval e : EVAL) {
            String a = answer(model, e.question());
            boolean ok = e.must().stream().allMatch(a::contains)
                    && e.mustNot().stream().noneMatch(a::contains);
            results.put(e.id(), ok);
        }
        return results;
    }

    static String quote(String s) {            // JSON string, ASCII text only
        return "\"" + s.replace("\\", "\\\\").replace("\"", "\\\"") + "\"";
    }

    static String pct(long part, long whole) {
        return String.format(Locale.ROOT, "%.1f%%", 100.0 * part / whole);
    }

    public static void main(String[] args) throws IOException {
        List<String> argv = Arrays.asList(args);
        List<Map<String, String>> items = readTsv("submissions.tsv");
        items.sort(Comparator.comparing(r -> r.get("utc_time")));
        String path = argv.contains("--decisions")
                ? argv.get(argv.indexOf("--decisions") + 1) : "decisions.tsv";
        Map<String, Map<String, String>> decisions = new HashMap<>();
        for (Map<String, String> d : readTsv(path)) decisions.put(d.get("id"), d);
```

Kode 7.2 (Lanjutan) *FeedbackPipeline.java: pipeline yang sama dalam Java, dengan record, java.time untuk WIB, satuan mikro berupa bilangan bulat, dan penyusun JSON Lines buatan sendiri.*

```
        System.out.println("1-2. SUBMIT and CURATE");
        List<Map<String, String>> dataset = curate(items, decisions);
        rewards(items);
        for (Map<String, String> it : items) {
            System.out.printf("  %s %s %s %-13s %-6s %s%n", it.get("id"),
                    it.get("user"), it.get("target"), it.get("category"),
                    it.get("severity"), it.get("status"));
        }

        System.out.println("PROFIT earned (lab values, integer micro-units shown"
                           + " as PROFIT)");
        long total = 0;
        for (Map<String, String> it : items) {
            long r = Long.parseLong(it.get("reward"));
            total += r;
            System.out.printf(Locale.ROOT, "  %s %s %s WIB  %.2f  %s%n",
                    it.get("id"), it.get("user"), it.get("wib"),
                    (double) r / MICRO, it.get("note"));
        }
        System.out.printf(Locale.ROOT, "  total emitted: %.2f PROFIT%n",
                          (double) total / MICRO);

        System.out.println("3. DATASET -> dataset.jsonl");
        try (PrintWriter w = new PrintWriter(Files.newBufferedWriter(
                Path.of("dataset.jsonl"), StandardCharsets.UTF_8))) {
            int n = 0;
            for (Map<String, String> it : dataset) {
                Q q = MODEL_V1.get(it.get("target"));
                w.print("{\"n\": " + (++n) + ", \"id\": " + quote(it.get("id"))
                        + ", \"prompt\": " + quote(q.prompt())
                        + ", \"rejected\": " + quote(q.answer())
                        + ", \"chosen\": " + quote(it.get("suggestion"))
                        + ", \"category\": " + quote(it.get("category")) + "}\n");
            }
        }
        System.out.println("  " + dataset.size() + " preference pairs written");
```

Kode 7.2 (Lanjutan) *FeedbackPipeline.java: pipeline yang sama dalam Java, dengan record, java.time untuk WIB, satuan mikro berupa bilangan bulat, dan penyusun JSON Lines buatan sendiri.*

```
        System.out.println("4. TRAIN and MEASURE (held-out questions)");
        Map<String, Q> modelV2 = new LinkedHashMap<>(MODEL_V1);
        for (Map<String, String> it : dataset) {  // "training" = update the bank
            String t = it.get("target");
            modelV2.put(t, new Q(MODEL_V1.get(t).prompt(), it.get("suggestion")));
        }
        Map<String, Boolean> before = measure(MODEL_V1), after = measure(modelV2);
        List<String> regressions = new ArrayList<>();
        int b = 0, a = 0;
        for (Eval e : EVAL) {
            boolean x = before.get(e.id()), y = after.get(e.id());
            if (x) b++;
            if (y) a++;
            if (x && !y) regressions.add(e.id());
            String mark = !x && y ? "fixed" : x && !y ? "REGRESSION" : "";
            String line = "  " + e.id() + "  before " + (x ? "pass" : "fail")
                    + "  after " + (y ? "pass" : "fail") + "  " + mark;
            System.out.println(line.stripTrailing());
        }
        int n = EVAL.size();
        System.out.println("  pass rate: " + b + "/" + n + " -> " + a + "/" + n
                           + " (" + pct(b, n) + " -> " + pct(a, n) + ")");

        System.out.println("5. RELEASE");
        String advice = a > b && regressions.isEmpty() ? "release"
                : !regressions.isEmpty()
                        ? "hold (regression on " + String.join(", ", regressions)
                          + ")"
                        : "hold (no improvement)";
        System.out.println("  the pipeline advises: " + advice);
        int r = argv.indexOf("--release");
        if (r < 0) {
            System.out.println("  waiting for a human: run again with --release"
                               + " approve or --release hold REASON");
        } else if (argv.get(r + 1).equals("approve")) {
            System.out.println("  human decision: approved -> Shipped in v1.1");
        } else {
            String reason = String.join(" ", argv.subList(r + 2, argv.size()));
            System.out.println("  human decision: Not taken in v1.1 ("
                               + (reason.isEmpty() ? "no reason" : reason) + ")");
        }
    }
}
```

Versi Java mencerminkan versi Python langkah demi langkah. `ZoneOffset.ofHours(7)` berperan sebagai zona WIB yang tetap, `LocalDateTime.parse(...).atOffset(ZoneOffset.UTC).withOffsetSameInstant(WIB)` mengubah waktu UTC menjadi WIB, serta `Locale.ROOT` dan `Locale.ENGLISH` menjaga angka dan nama bulan tetap identik di setiap komputer. JSON ditulis dengan tangan melalui metode kecil `quote`, yang cukup karena teks lab berupa ASCII biasa; sistem produksi akan memakai pustaka JSON.

7.10.3 Menjalankan *pipeline*

Buka terminal di `Chapter07/python` dan jalankan programnya, atau di `Chapter07/java` kompilasi dan jalankan versi Java:

```
python feedback_pipeline.py
javac FeedbackPipeline.java
java FeedbackPipeline
```

Keduanya mencetak persis keluaran berikut; kami membandingkan kedua keluaran itu, dan kedua berkas `dataset.jsonl`, dengan `diff`.

```
1-2. SUBMIT and CURATE
  F01 u1 B3 wrong fact    medium in dataset #1
  F02 u2 B3 wrong fact    medium duplicate of F01 (automatic)
  F03 u3 B1 missing info  low    in dataset #2
  F04 u4 B5 unsafe advice high   in dataset #3
  F05 u1 B6 wrong fact    medium in dataset #4
  F06 u5 B4 missing info  low    in dataset #5
  F07 u6 B4 other         low    not taken: too short (automatic)
  F08 u2 B2 other         low    not taken: contradicts the price list (Rp7,000)
  F09 u1 B2 other         low    not taken: same as the current answer
  F10 u1 B4 missing info  low    in dataset #6
  F11 u1 B1 missing info  low    awaiting a human decision
PROFIT earned (lab values, integer micro-units shown as PROFIT)
  F01 u1 24 Sep 09:00 WIB  0.05  base + accepted bonus
  F02 u2 24 Sep 09:40 WIB  0.00  duplicate of F01
  F03 u3 24 Sep 10:10 WIB  0.05  base + accepted bonus
  F04 u4 24 Sep 10:30 WIB  0.05  base + accepted bonus
  F05 u1 24 Sep 11:00 WIB  0.05  base + accepted bonus
  F06 u5 24 Sep 12:00 WIB  0.00  held: email not verified
  F07 u6 24 Sep 12:30 WIB  0.00  not taken: too short
  F08 u2 24 Sep 13:00 WIB  0.01  base
  F09 u1 24 Sep 17:00 WIB  0.01  base
  F10 u1 24 Sep 23:30 WIB  0.00  daily cap reached (resets 00:00 WIB)
  F11 u1 25 Sep 00:10 WIB  0.01  base
  total emitted: 0.23 PROFIT
3. DATASET -> dataset.jsonl
  6 preference pairs written
4. TRAIN and MEASURE (held-out questions)
  E1  before fail  after pass  fixed
  E2  before pass  after fail  REGRESSION
  E3  before pass  after pass
  E4  before fail  after pass  fixed
  E5  before fail  after pass  fixed
  E6  before fail  after pass  fixed
  E7  before fail  after pass  fixed
  E8  before fail  after pass  fixed
  pass rate: 2/8 -> 7/8 (25.0% -> 87.5%)
5. RELEASE
  the pipeline advises: hold (regression on E2)
  waiting for a human: run again with --release approve or --release hold REASON
```

Bacalah laporannya dari atas.

Kurasi. Enam butir diterima dan diberi nomor #1 sampai #6 menurut urutan kedatangan (F01, F03, F04, F05, F06, F10). F02 dikenali sebagai duplikat F01 menurut Persamaan (7.1), persis seperti contoh di Subbab 7.5.2. F07 terlalu pendek. F08 dan F09 ditolak peninjau, dengan alasan. F11 menunggu keputusan manusia karena belum ada yang melihatnya. F10 dan F06 sama-sama memperbaiki B4 tetapi bukan duplikat ($J = 6/9 \approx 0.67 < 0.7$), dan F11 bukan duplikat F03 ($J = 3/5 = 0.6$).

Hadiah. Kini waktunya tampil dalam WIB. F01, F03, F04, dan F05 masing-masing memperoleh 0,05 (dasar ditambah bonus penerimaan). F06 *belum* memperoleh apa-apa: isinya baik dan diterima, tetapi akunnya belum terverifikasi, sehingga hadiahnya ditahan. F02 dan F07 tidak memperoleh apa-apa, dengan alasan yang ditampilkan. F08 dan F09 masing-masing memperoleh hadiah dasar 0,01: keduanya kiriman yang jujur dan tidak remeh yang sudah ditinjau seseorang, walaupun tidak diambil. Pengguna u1 sudah memiliki tiga butir berhadiah pada 24 September sampai pukul 17.00 WIB (F01, F05, F09), sehingga F10 pukul 23.30 WIB terkena batas harian lab, meskipun isinya diterima ke dataset. F11 pukul 00.10 WIB jatuh pada 25 September dan kembali

memperoleh hadiah dasar. Total yang diemisikan putaran ini adalah 0,23 PROFIT, dihitung secara eksak dalam satuan mikro.

Dataset. Enam pasangan preferensi ditulis. Baris pertama `dataset.jsonl`, dirapikan dengan `python -m json.tool --json-lines`, adalah:

```
{
    "n": 1,
    "id": "F01",
    "prompt": "how long does express service take",
    "rejected": "Express service takes 2 days.",
    "chosen": "Express service is ready in 6 hours and costs Rp12,000 per kilogram.",
    "category": "wrong fact"
}
```

Pengukuran. Tingkat kelolosan naik dari 2 dari 8 (25,0%) menjadi 7 dari 8 (87,5%). E1 (jam buka), E4 dan E5 (ekspres), E6 (penjemputan), E7 (sutra), dan E8 (QRIS) diperbaiki. Tetapi E2, “Are your opening hours the same every day?”, lolos sebelumnya dan gagal sesudahnya. Penyebabnya patut dipelajari: F03 mengganti “We are open every day.” dengan “We are open from 07:00 to 21:00.”, yang menambahkan jam buka tetapi diam-diam menghapus “every day”. Koreksi itu memperbaiki satu pertanyaan dan merusak pertanyaan lain, persis hal yang hendak ditangkap Persamaan (7.4).

Rilis. Karena $R = \{\text{E2}\}$ tidak kosong, Persamaan (7.6) menyarankan penahanan, dan program menunggu seseorang. Misalkan peninjau membaca laporan itu dan memutuskan menahan rilis sampai F11, yang mempertahankan “every day” sekaligus menambahkan jam buka, selesai ditinjau:

```
python feedback_pipeline.py --release hold E2 regressed, wait for F11
```

Tiga baris terakhir laporan kini mencatat keputusan manusia itu, yang akan dilihat para pemberi umpan balik sebagai “Not taken in v1.1” beserta alasannya:

```
5. RELEASE
  the pipeline advises: hold (regression on E2)
  human decision: Not taken in v1.1 (E2 regressed, wait for F11)
```

Soal 7.2 melanjutkan kisahnya: terima F11 lalu jalankan *pipeline* sekali lagi.

7.10.4 Hari apa sekarang? UTC lawan WIB

Kode 7.3 dan **Kode 7.4** memisahkan pelajaran zona waktu dari Persamaan (7.10) dalam belasan baris saja.

Kode 7.3 `day_key.py`: tanggal UTC dan tanggal WIB dari empat saat.

```
"""Which "day" does a submission belong to? UTC date versus WIB date.

    python day_key.py
"""
from datetime import datetime, timedelta, timezone

WIB = timezone(timedelta(hours=7))
TIMES = ["2026-09-24T16:30Z", "2026-09-24T17:10Z",
         "2026-09-24T23:50Z", "2026-09-25T00:10Z"]

print("UTC time           WIB time           UTC day  WIB day")
for s in TIMES:
    t = datetime.strptime(s, "%Y-%m-%dT%H:%MZ").replace(tzinfo=timezone.utc)
    w = t.astimezone(WIB)
    print(f"{t:%d %b %H:%M} UTC   {w:%d %b %H:%M} WIB   "
          f"{t:%d %b}   {w:%d %b}")
print("A cap keyed on the UTC day resets at 07:00 WIB, not at midnight.")
```

Kode 7.4 `DayKey.java`: perbandingan yang sama dengan `OffsetDateTime` dan `ZoneOffset.ofHours(7)`.

```
import java.time.OffsetDateTime;
import java.time.ZoneOffset;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.Locale;

/** Which "day" does a submission belong to? UTC date versus WIB date.
 *  java DayKey */
public class DayKey {
    public static void main(String[] args) {
        ZoneOffset wib = ZoneOffset.ofHours(7);
        DateTimeFormatter full = DateTimeFormatter.ofPattern("dd MMM HH:mm",
                                                             Locale.ENGLISH);
        DateTimeFormatter day =
                DateTimeFormatter.ofPattern("dd MMM", Locale.ENGLISH);
        List<String> times = List.of("2026-09-24T16:30Z", "2026-09-24T17:10Z",
                                     "2026-09-24T23:50Z", "2026-09-25T00:10Z");
        System.out.println("UTC time           WIB time"
                           + "           UTC day  WIB day");
        for (String s : times) {
            OffsetDateTime t = OffsetDateTime.parse(s.replace("Z", ":00Z"));
            OffsetDateTime w = t.withOffsetSameInstant(wib);
            System.out.println(t.format(full) + " UTC   " + w.format(full)
                               + " WIB   " + t.format(day) + "   " + w.format(day));
        }
        System.out.println("A cap keyed on the UTC day resets at 07:00 WIB,"
                           + " not at midnight.");
    }
}
```

```
UTC time           WIB time           UTC day  WIB day
24 Sep 16:30 UTC   24 Sep 23:30 WIB   24 Sep   24 Sep
24 Sep 17:10 UTC   25 Sep 00:10 WIB   24 Sep   25 Sep
24 Sep 23:50 UTC   25 Sep 06:50 WIB   24 Sep   25 Sep
25 Sep 00:10 UTC   25 Sep 07:10 WIB   25 Sep   25 Sep
A cap keyed on the UTC day resets at 07:00 WIB, not at midnight.
```

Baris kedua dan ketiga adalah yang berbahaya: bagi pengguna di Indonesia keduanya sudah tanggal 25 September, tetapi program yang bertanya “tanggal berapa hari ini?” dalam UTC masih menjawab 24 September. Baru pada pukul 00.10 UTC, yaitu 07.10 WIB, keduanya kembali sepakat. Batas harian yang dikaitkan pada tanggal UTC akan membiarkan pengguna memperoleh jatah kedua yang penuh antara tengah malam dan pukul 07.00 WIB, lalu diatur ulang di tengah pagi, saat pengguna paling tidak menduganya.

7.11 Profitina dalam Praktik

Lab adalah miniatur; inilah *pipeline* yang sesungguhnya, pada tingkat kotak.

Dari mana umpan balik datang. Pengguna dapat memberi umpan balik atas hasil pindai dan atas keluaran AI, dan status umpan balik tampil di dasbor mereka, "kokpit" yang mengumpulkan pemberitahuan tentang balasan, suka, *repost*, pesan langsung, sematan, status umpan balik, dan pembelian PROFIT di satu tempat, masing-masing tertaut ke bagiannya [5]. Pemberi umpan balik tidak pernah perlu bertanya-tanya apa yang terjadi pada butirnya.

Lima langkah. Kirim (memperoleh PROFIT) → Kurasi (satu panggilan AI lokal memilah kategori, keparahan, dan ringkasan, lalu mengelompokkan duplikat) → Dataset ("In dataset #N", terlihat oleh pemberinya) → Latih dan ukur model lokal → Rilis ("Shipped in vX.Y.Z", atau "Not taken" dengan alasan). Keputusan produk selalu diambil manusia, dan pemberi umpan balik yang memilih ikut serta diberi kredit dengan nama samaran di halaman publik "You changed this" [5]. Penyetelan bobot model, misalnya dengan LoRA, masih merupakan butir peta jalan per September 2026, dan keputusan rilis tetap berada di tangan manusia bagaimanapun juga.

Hadiah dengan pagar pengaman. Perolehan dimulai dari 0,01 PROFIT, batas harian diatur ulang pukul 00.00 WIB, hanya manusia yang surelnya terverifikasi yang memperoleh hadiah, penilaian kebermanfaatan oleh AI lokal memberi kontribusi remeh atau mirip bot nilai 0,00 (dengan *escrow* dan penarikan kembali menuju pembakaran), dan duplikat memperoleh 0,00 sambil ditautkan ke butir sebelumnya [5]. Jumlahnya dikalikan pengali emisi yang menurun (Bab 12). PROFIT berjalan di Solana *devnet*, penjualan dinonaktifkan, dan tidak ada apa pun tentangnya yang merupakan janji nilai.

Prinsip yang sama di tempat lain. Gagasan bahwa aplikasi memberi saran dan pengguna yang memutuskan muncul di seluruh produk: hasil visibilitas adalah perkiraan, bukan jaminan; pengisian otomatis kategori berkata "belum yakin" alih-alih menebak; *One Text, Many Platforms* menjanjikan batas karakter setiap platform tetapi tidak menjanjikan hasil yang "viral"; dan hak privasi swalayan (ekspor, koreksi, penghapusan) selalu berada di tangan pengguna (Bab 11) [5].

Apa yang bukan lab ini. Kategori, ambang duplikat, nilai hadiah, batas 3, dan aturan rilis di lab adalah pilihan pengajaran buku ini sendiri, dan "pelatihan"-nya hanyalah pembaruan bank jawaban. Penilaian kebermanfaatan, ambang duplikat, dan waktu *escrow* milik produk adalah milik produk sendiri dan tidak diuraikan di sini.

7.12 Kesalahpahaman yang Umum

"Umpan balik yang lebih banyak selalu lebih baik." Umpan balik yang tidak disaring mengandung kekeliruan, duplikat, *spam*, dan angan-angan (F08 menginginkan harga yang lebih murah). Kurasi, keputusan manusia, dan pengukuran mengubah setumpuk umpan balik menjadi sedikit data yang dapat dipercaya.

"Jika model dapat memilah umpan balik, ia juga dapat memutuskannya." Memilah adalah usulan yang menghemat perhatian; memutuskan mengubah apa yang dikatakan produk kepada orang. Aturan lab memilah F11 dengan benar sebagai *missing info*, tetapi hanya manusia yang dapat menilai bahwa butir itulah perbaikan bagi regresi E2.

"Tingkat kelolosan yang lebih tinggi berarti rilis itu aman." Kandidat di lab naik dari 25,0% menjadi 87,5% dan tetap merusak E2. Selalu lihat himpunan regresi *R*, bukan hanya rata-ratanya.

"Memberi hadiah untuk umpan balik akan mendatangkan bot, jadi jangan beri hadiah." Hadiah tanpa pengaman memang menarik bot. Batas harian, verifikasi, penilaian kebermanfaatan, deteksi duplikat, dan *escrow* memungkinkan hadiah tanpa mengubahnya menjadi ladang panen. Alternatifnya, tidak memberi hadiah sama sekali, sama dengan berkata kepada para kontributor bahwa usaha mereka tidak ada nilainya.

"Umpan balik yang ditolak seharusnya tidak memperoleh apa-apa." Umpan balik yang jujur dan tidak remeh, yang sudah ditinjau seseorang, tetap bernilai walaupun tidak diambil; ia menunjukkan di mana pengguna bingung. Lab membayar hadiah dasar untuk butir semacam itu dan bonus hanya untuk butir yang diterima. Yang tidak memperoleh apa-apa adalah kebisingan: duplikat, teks remeh, dan akun yang belum terverifikasi.

"Batas harian ya batas harian; zona waktu tidak penting." Batas yang dikaitkan pada tanggal UTC diatur ulang pukul 07.00 WIB. Bagi pengguna di Indonesia, "hari ini" dimulai pukul 00.00 WIB, dan kode harus menyatakannya secara eksplisit.

"Hadiah PROFIT adalah cara mencari uang." PROFIT adalah koin utilitas di Solana *devnet* yang saat ini tidak memiliki nilai moneter nyata; penjualan dinonaktifkan; jumlahnya kecil dan dibatasi; dan tidak ada apa pun di produk maupun di buku ini yang menjanjikan nilai atau imbal hasil.

"*Pipeline* umpan balik hanyalah kotak masuk layanan pelanggan dengan langkah tambahan." Kotak masuk menjawab satu orang; *pipeline* mengubah apa yang dikatakan produk kepada semua orang. Karena itu ia memerlukan hal-hal yang tidak dimiliki kotak masuk: pengelompokan duplikat, dataset bernomor dan berversi, pengukuran pada pertanyaan uji terpisah, dan keputusan rilis yang dicatat beserta alasannya. Memperlakukan umpan balik hanya sebagai tiket layanan akan kehilangan perbaikannya; memperlakukannya hanya sebagai data pelatihan akan kehilangan manusianya. *Pipeline* menjaga keduanya: pemberi umpan balik melihat status butirnya, dan produk belajar darinya.

"Token *devnet* hanyalah demo, jadi aturan di sekitarnya tidak penting." Aturan itu justru penting karena sedang diuji sebelum ada nilai nyata yang terlibat. Batas harian, verifikasi, penanganan duplikat, dan pembakaran yang dirancang dan diukur di *devnet* adalah aturan yang akan dibawa serta bila gerbang regulasi dan audit untuk *mainnet* kelak terlewati; aturan yang ditambahkan terburu-buru belakangan adalah jalan masuk penyalahgunaan sistem hadiah.

"Setelah dataset jadi, himpunan evaluasi boleh dipakai untuk pelatihan juga." Maka himpunan itu tidak lagi terpisah, dan setiap pengukuran berikutnya melebih-lebihkan mutu. Pisahkan himpunan evaluasi, dan terus tambahkan pertanyaan baru ke dalamnya.

7.13 Ringkasan Bab

- Model tidak membaik dengan sendirinya; pengguna yang pertama kali menyadari kesalahannya. Umpan balik manusia, terutama perbandingan dan koreksi konkret, adalah cara model mempelajari apa yang disukai manusia [1], [2].
- *Pipeline* Profitina memiliki lima langkah: kirim, kurasi, dataset, latih dan ukur, rilis. Mesin memilah dan mengukur; manusia menerima atau menolak butir serta menyetujui atau menahan rilis; pemberi umpan balik melihat setiap statusnya ("In dataset #N", "Shipped in vX.Y.Z", "Not taken" beserta alasan).

- *Aplikasi memberi saran, pengguna yang memutuskan*: keluaran disajikan sebagai perkiraan dan usulan, dan keputusan yang mengubah produk diambil manusia, sejalan dengan Pasal 10 UU PDP tentang keputusan otomatis.
- Butir yang hampir kembar dapat ditemukan dengan kesamaan Jaccard dari himpunan kata (Persamaan (7.1)); keputusan otomatis hanya boleh membuang kebisingan dan harus selalu menyebut alasannya.
- Koreksi yang sudah dikurasi menjadi pasangan preferensi (*prompt*, *rejected*, *chosen*) yang disimpan sebagai JSON Lines, satu objek per baris, tanpa data pribadi.
- Ukurlah pada himpunan uji terpisah: hitung tingkat kelolosan (Persamaan (7.3)) sebelum dan sesudah, serta himpunan regresi (Persamaan (7.4)). Lab naik dari 2/8 menjadi 7/8 dengan satu regresi, sehingga *pipeline* menyarankan penahanan (Persamaan (7.6)).
- Hadiah PROFIT untuk umpan balik itu kecil (0,01–0,05 per umpan balik atas hasil pindai), dibatasi per hari dengan hari yang didefinisikan dalam WIB (Persamaan (7.8), Persamaan (7.10)), hanya dibayarkan kepada manusia terverifikasi, dan nol untuk duplikat serta konten remeh. Jumlahnya paling baik disimpan sebagai satuan mikro berupa bilangan bulat (Persamaan (7.9)). PROFIT adalah koin utilitas di *devnet*; penjualan dinonaktifkan; tidak ada nilai yang dijanjikan.
- Lab menerapkan seluruh siklus dalam Python dan Java dengan keluaran yang identik: kurasi dengan keputusan manusia, dataset JSONL, hadiah yang dibatasi, pengukuran sebelum dan sesudah, serta rilis yang menunggu seorang manusia.

7.14 Soal Latihan

Soal 7.1 (Jaccard dengan tangan). Hitunglah kesamaan Jaccard, setelah pembersihan ala lab (huruf kecil, koma di dalam angka dibuang, kata umum dibuang), antara F06 “Yes, free pickup and delivery within 3 km.” dan F10 “We offer free pickup and delivery within 3 km of the shop.” Kemudian hitunglah untuk F03 dan F11. Dengan ambang lab 0,7, pasangan mana yang merupakan duplikat?

Soal 7.2 (menuntaskan kisahnya). Peninjau membaca F11 dan menerimanya. Tambahkan baris `F11 accept keeps "every day" and adds the hours` ke salinan berkas keputusan (folder kode buku ini menyertakannya sebagai `decisions_v2.tsv`), lalu jalankan `python feedback_pipeline.py --decisions decisions_v2.tsv --release approve`. Ramalkan sebelum menjalankan: banyaknya pasangan preferensi, tingkat kelolosan sesudahnya, regresinya, sarannya, dan hadiah F11.

Soal 7.3 (buku besar hadiah). Dengan Persamaan (7.7), $B = 0.01$, $A = 0.04$, dan batas lab $K = 3$, hitung ulang dengan tangan hadiah kelima butir milik pengguna u1 pada eksekusi di Subbab 7.10.3, dalam satuan mikro dan dalam PROFIT. Apa yang berubah bila batas harian dikaitkan pada tanggal UTC?

Soal 7.4 (ambang yang lebih rendah). Ubah ambang duplikat di lab dari 0,7 menjadi 0,6. Butir mana yang berubah statusnya? Jelaskan mengapa hal ini lebih buruk bagi produk, walaupun “lebih banyak duplikat tertangkap” terdengar lebih baik.

Soal 7.5 (tingkat kelolosan dan regresi). Sebuah model kandidat diukur pada himpunan uji terpisah berisi 20 pertanyaan. Sebelum: pertanyaan 1–12 lolos. Sesudah: pertanyaan 1–10 dan 13–18 lolos. Hitunglah p_{sebelum}, p_{sesudah}, dan R, berikan saran menurut Persamaan (7.6), dan uraikan dua keputusan manusia berbeda yang sama-sama masuk akal, beserta alasannya.

Soal 7.6 (model di langkah kurasi). Dengan `OllamaClient` dari Bab 6, uraikan bagaimana Anda akan mengganti aturan kata kunci lab untuk kategori dan keparahan dengan satu panggilan jalur cepat per butir, sambil menjaga *pipeline* tetap jujur ketika tidak ada model yang tersedia. Bagian mana dari `curate` yang harus tetap berbasis aturan?

Soal 7.7 (titik mengambang). Di Python, menjumlahkan kesebelas hadiah di Subbab 7.10.3 secara berurutan sebagai bilangan titik mengambang, `0.05 + 0.00 + 0.05 + 0.05 + 0.05 + 0.00 + 0.00 + 0.01 + 0.01 + 0.00 + 0.01`, menghasilkan `0.23000000000000004`, dan menjumlahkan `0.01` sepuluh kali menghasilkan `0.09999999999999999`. Jelaskan mengapa, dan tunjukkan bagaimana satuan mikro berupa bilangan bulat di lab menghindari masalah itu. Berapa 0,05 PROFIT dalam satuan mikro, dan berapa satuan mikro yang diemisikan seluruh putaran itu?

Soal 7.8 (apa yang dilihat pemberi umpan balik). Untuk F02, F06, F08, F10, dan F11 pada eksekusi pertama, tuliskan status satu baris yang seharusnya dilihat pemberi umpan balik di dasbornya, dalam bahasa yang sederhana.

Soal 7.9 (pemeriksaan dataset). Tuliskan tiga pemeriksaan yang seharusnya dijalankan pada setiap baris `dataset.jsonl` sebelum dipakai, dan sebutkan apa yang dilindungi setiap pemeriksaan.

Soal 7.10 (memberi saran atau memutuskan?). Untuk setiap situasi, tentukan apakah perangkat lunak boleh memutuskan sendiri atau hanya memberi saran, dan mengapa: (a) menolak butir umpan balik yang hanya dua huruf; (b) merilis versi model baru dengan tingkat kelolosan lebih tinggi dan tanpa regresi; (c) memberikan 0,00 PROFIT kepada duplikat; (d) memblokir akun pengguna karena umpan baliknya tampak otomatis.

Soal 7.11 (mengurutkan antrean). Dengan kunci pengurutan Persamaan (7.2) serta kategori dan keparahan yang dicetak di Subbab 7.10.3, susunlah urutan kesembilan butir yang tidak dibuang secara otomatis sebagaimana seharusnya dilihat peninjau. Anggap kelompok F01 berukuran 2 (termasuk F02) dan setiap kelompok lain berukuran 1. Di mana posisi F11, dan apa artinya bagi pengurutan yang hanya berdasarkan keparahan?

7.15 Penyelesaian Soal Latihan

Penyelesaian 7.1. F06 menghasilkan {yes, free, pickup, delivery, within, 3, km} (7 kata; “and” adalah kata umum) dan F10 menghasilkan {offer, free, pickup, delivery, within, 3, km, shop} (8 kata; “we”, “and”, “of”, dan “the” adalah kata umum). Keduanya berbagi {free, pickup, delivery, within, 3, km}, yaitu 6 kata, dan bersama-sama memiliki $7 + 8 - 6 = 9$ kata berbeda, sehingga $J = 6/9 \approx 0.667$. F03 menghasilkan {open, 07:00, 21:00} dan F11 menghasilkan {open, every, day, 07:00, 21:00}; keduanya berbagi 3 kata dari 5 kata berbeda, sehingga $J = 3/5 = 0.6$. Tidak satu pun pasangan mencapai 0,7, jadi keduanya bukan duplikat; hanya F01 dan F02 ($J = 0.7$) yang duplikat.

Penyelesaian 7.2. Eksekusinya mencetak:

```
3. DATASET -> dataset.jsonl
   7 preference pairs written
4. TRAIN and MEASURE (held-out questions)
   E1  before fail  after pass  fixed
   E2  before pass  after pass
   E3  before pass  after pass
   E4  before fail  after pass  fixed
   E5  before fail  after pass  fixed
   E6  before fail  after pass  fixed
   E7  before fail  after pass  fixed
   E8  before fail  after pass  fixed
   pass rate: 2/8 -> 8/8 (25.0% -> 100.0%)
5. RELEASE
   the pipeline advises: release
   human decision: approved -> Shipped in v1.1
```

Tujuh pasangan preferensi ditulis, karena F11 masuk dataset sebagai #7. Jawaban B1 kini "We are open every day from 07:00 to 21:00." milik F11, yang memenuhi E1 (jam buka) maupun E2 ("every day"), sehingga tingkat kelolosan menjadi 8/8 dan R kosong. Persamaan (7.6) menyarankan rilis, dan manusia menyetujuinya: para pemberi umpan balik kini membaca "Shipped in v1.1". Hadiah F11 naik dari 0,01 menjadi 0,05 karena diterima; hadiah itu masih berada dalam jatah u1 untuk 25 September. Delapan dari delapan pada delapan pertanyaan memang menggembirakan tetapi bukan bukti; putaran berikutnya harus menambahkan pertanyaan uji terpisah yang baru.

Penyelesaian 7.3. Dalam WIB, butir-butir u1 jatuh pada pukul 09.00 (F01), 11.00 (F05), 17.00 (F09), dan 23.30 (F10) tanggal 24 September, serta pukul 00.10 (F11) tanggal 25 September. F01 dan F05 diterima: masing-masing $10{,}000 + 40{,}000 = 50{,}000$ µP, yaitu 0,05 PROFIT. F09 ditolak tetapi memenuhi syarat: 10,000 µP, 0,01 PROFIT, dan menjadi butir berhadiah ketiga hari itu. F10 adalah yang keempat, sehingga Persamaan (7.8) membuat elig $= 0$ dan hadiahnya 0. F11 adalah butir pertama tanggal 25 September: 10,000 µP, 0,01 PROFIT, karena belum diterima. Total untuk u1: 120,000 µP, 0,12 PROFIT. Dengan batas berbasis UTC, kelima butir jatuh pada 24 September UTC (F11 pukul 17.10 UTC); F11 lalu menjadi butir kelima hari itu dan memperoleh 0, sementara batas baru diatur ulang pukul 07.00 WIB tanggal 25 September.

Penyelesaian 7.4. Dengan 0,6, F11 menjadi "duplikat F03" ($J = 0.6$), sedangkan F10 dan F06 ($J \approx 0.667$) juga menjadi duplikat, sehingga F10 dibuang secara otomatis. Dataset kehilangan tepat butir (F11) yang memperbaiki regresi E2, serta kehilangan susunan kata F10 yang lebih jelas, dan tidak ada manusia yang pernah melihat keduanya. Menangkap lebih banyak "duplikat" berarti mesin diam-diam membuang lebih banyak informasi; ambang yang lebih rendah seharusnya hanya dipakai bila disertai pemeriksaan manusia atas butir-butir yang dikelompokkan.

Penyelesaian 7.5. $p_{\text{sebelum}} = 12/20 = 60\%$ dan $p_{\text{sesudah}} = 16/20 = 80\%$. Regresinya $R = \{11,12\}$, yang lolos sebelumnya dan gagal sesudahnya. Karena R tidak kosong, sarannya adalah *tahan*. Keputusan manusia yang masuk akal: (1) tahan, bila pertanyaan 11 dan 12 menyangkut keselamatan atau uang (misalnya harga atau petunjuk perawatan pakaian), lalu perbaiki keduanya dulu; (2) setujui, bila 11 dan 12 hanya soal susunan kata yang kecil dan pertanyaan 13–18 yang diperbaiki mencakup masalah keselamatan yang sedang dialami pengguna, sambil mencatat regresi itu sebagai masalah yang diketahui untuk putaran berikutnya. Bagaimanapun, alasannya dituliskan.

Penyelesaian 7.6. Untuk setiap butir, kirim *prompt* pendek ke jalur cepat yang hanya berisi komentar dan usulan, minta JSON seperti `{"category": ..., "severity": ...}` dengan `format: "json"`, validasi nilainya terhadap daftar yang diizinkan, dan coba ulang sekali, seperti pada Penyelesaian 6.6. Bila klien memunculkan `NoModelAvailable` atau jawabannya tidak sah,

kembali ke aturan kata kunci dan catat "dipilah dengan aturan". *Penolakan* otomatis, yaitu terlalu pendek dan duplikat, sebaiknya tetap berbasis aturan dan dapat dijelaskan, dan keputusan terima/tolak harus tetap di tangan manusia. Peran model adalah menghemat waktu peninjau, tidak pernah memutuskan.

Penyelesaian 7.7. Titik mengambang biner menyimpan bilangan sebagai jumlah pangkat dua, dan 0,01 serta 0,05 tidak memiliki representasi biner yang eksak, sama seperti 1/3 tidak memiliki representasi desimal yang eksak. Setiap nilai yang tersimpan meleset sedikit sekali, dan kesalahannya menumpuk dalam penjumlahan, sehingga totalnya berbeda dari 0,23 pada angka penting ketujuh belas. Buku besar yang membandingkan total semacam itu, atau menjumlahkan jutaan nilai, akan bergeser. Bilangan bulat tidak memiliki masalah itu: $10{,}000 + 40{,}000 = 50{,}000$ secara eksak, dan lab baru mengubahnya menjadi PROFIT saat mencetak. Jadi 0,05 PROFIT adalah 50,000 µP, dan putaran di Subbab 7.10.3 mengemisikan $0.23 \times 10^6 = 230{,}000$ µP: empat butir diterima masing-masing 50,000 ditambah tiga hadiah dasar masing-masing 10,000.

Penyelesaian 7.8. F02: "Terima kasih! Laporan ini sama dengan laporan sebelumnya (F01), jadi dikelompokkan bersamanya; duplikat tidak memperoleh PROFIT." F06: "Diterima ke dataset. Verifikasi alamat surel Anda untuk menerima hadiahnya." F08: "Sudah ditinjau, tidak diambil: bertentangan dengan daftar harga saat ini (Rp7.000 per kilogram)." F10: "Diterima ke dataset (#6). Anda sudah mencapai batas hadiah hari ini; batas diatur ulang pukul 00.00 WIB." F11: "Sudah diterima; menunggu tinjauan."

Penyelesaian 7.9. (1) Baris dapat diurai sebagai JSON dan memiliki tepat kolom-kolom yang diharapkan dengan nilai *string* yang tidak kosong, yang melindungi perkakas pelatihan dari data yang rusak. (2) Tidak ada kolom yang memuat alamat surel, nomor telepon, atau pengenal pribadi lain (pemeriksaan pola), yang melindungi privasi. (3) `id` merujuk ke butir yang berstatus "in dataset", dan tidak ada dua baris dengan `id` yang sama atau pasangan *prompt–chosen* yang sama, yang melindungi dari baris sisipan atau baris ganda. Pemeriksaan keempat yang berguna: tidak ada `prompt` yang sama dengan pertanyaan uji terpisah, yang melindungi himpunan evaluasi.

Penyelesaian 7.10. (a) Putuskan secara otomatis: aturannya sederhana, dapat dijelaskan, dan hanya membuang kebisingan; statusnya menyebut alasannya. (b) Hanya memberi saran: rilis mengubah apa yang dikatakan produk kepada semua orang, sehingga seorang manusia yang menyetujuinya, bahkan bila angkanya bersih. (c) Putuskan secara otomatis, dengan alasan dan tautan ke butir sebelumnya; pemberi umpan balik dapat berkeberatan bila pengelompokannya salah. (d) Hanya memberi saran: memblokir akun berdampak signifikan bagi seseorang, persis jenis keputusan yang hanya didasarkan pada pemrosesan otomatis yang dapat diajukan keberatannya menurut Pasal 10 UU PDP [6]; seorang manusia harus meninjaunya.

Penyelesaian 7.11. Kesembilan butir itu adalah F01, F03, F04, F05, F06, F08, F09, F10, dan F11. F04 satu-satunya butir berkeparahan *high*, jadi tampil pertama. Butir berkeparahan *medium* adalah F01 (kelompok berukuran 2) dan F05 (kelompok berukuran 1), sehingga F01 mendahului F05 walaupun keduanya *wrong fact*. Semua butir lainnya berkeparahan *low* dengan kelompok berukuran 1, sehingga menyusul menurut urutan kedatangan: F03, F06, F08, F09, F10, F11. Urutannya adalah F04, F01, F05, F03, F06, F08, F09, F10, F11, dan F11, butir yang memperbaiki regresi E2, berada paling akhir. Peninjau yang waktunya sempit mungkin tidak pernah sampai ke sana. Keparahan menggambarkan seberapa buruk masalah yang dilaporkan, bukan seberapa berharga perbaikannya; karena itu antrean yang praktis juga menonjolkan butir yang menyentuh jawaban dengan regresi yang diketahui, atau butir yang *melengkapi* koreksi yang sudah diterima, dan inilah alasan yang baik untuk tetap menempatkan manusia, bukan hanya rumus, sebagai pengendali antrean.

Daftar Pustaka

[1] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022, pp. 27730–27744.

[2] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017, pp. 4299–4307.

[3] Stack Overflow, "AI," *2025 Stack Overflow Developer Survey*, Jul. 2025. [Online]. Available: https://survey.stackoverflow.co/2025/ai

[4] S. Amershi *et al.*, "Guidelines for human-AI interaction," in *Proc. CHI Conf. Human Factors in Computing Systems (CHI '19)*, Glasgow, UK, 2019, pp. 1–13.

[5] Profitina, *Profitina Whitepaper — Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[6] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi* (UU PDP), Oct. 17, 2022, esp. Art. 10(1). [Online]. Available: https://jdih.komdigi.go.id/produk_hukum/view/id/832/t/undangundang+nomor+27+tahun+202

[7] E. J. Hu *et al.*, "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2022.

[8] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," in *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023.

[9] "JSON Lines," *jsonlines.org*. [Online]. Available: https://jsonlines.org/ (accessed Sep. 24, 2026).

[10] T. Bray, Ed., "The JavaScript Object Notation (JSON) data interchange format," IETF RFC 8259, Dec. 2017.

[11] M. Strathern, "'Improving ratings': Audit in the British university system," *European Review*, vol. 5, no. 3, pp. 305–321, 1997.

[12] D. Sculley *et al.*, "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015, pp. 2503–2511.

[13] Solana Foundation, "Token Program," *Solana Program Library documentation*. [Online]. Available: https://spl.solana.com/token (accessed Sep. 24, 2026).

Bab 8
Model Bisnis

Setelah menyelesaikan bab ini, Anda akan mampu menjelaskan, mula-mula dalam satu kalimat lalu secara terperinci, bagaimana Profitina berniat membiayai dirinya sendiri: dua tingkat layanan, Free dan Pro, dengan Pro seharga Rp50.000 untuk 30 hari yang dapat dibayar lewat QRIS atau ditukar dengan PROFIT, dapat diperpanjang otomatis dari saldo PROFIT bila pengguna menghendakinya, dan tidak pernah disertai iklan. Anda akan memahami mengapa tingkat ketiga, Agency, dihapus pada Agustus 2026 meskipun semua fiturnya berjalan, dan mengapa tim memegang aturan "ukur minat sebelum membangun pembayaran" setelah pemeriksaan jujur atas angka-angka menunjukkan nol pembelian PROFIT sepanjang riwayat produksi. Anda akan dapat mengatakan dengan tepat apa arti "kuota harian" di negeri yang harinya dimulai tujuh jam lebih awal daripada Greenwich, dan mengapa pengaturan ulangnya harus terjadi pada pukul 00.00 WIB. Anda akan membaca pasar tempat semua ini berlangsung, dari survei internet APJII 2026 dan angka QRIS Bank Indonesia. Terakhir, Anda akan membangun dua program kecil tetapi lengkap, dalam Python dan Java: kalkulator kuota dan tingkat layanan yang menghitung hari dengan cara Indonesia, serta langganan yang dimodelkan sebagai *state machine* (mesin keadaan) yang menerima pembayaran QRIS, penukaran PROFIT, pengingat, dan perpanjangan, lengkap dengan serangkaian uji yang tidak membiarkan aturan-aturannya bergeser.

8.1 Titik Berangkat Kita

Bab 7 menutup siklus umpan balik: orang mengirim masukan, platform mengkurasinya, sebuah dataset tumbuh, model lokal dilatih dan diukur, dan manusia memutuskan apa yang dirilis. Siklus itu berbicara tentang *mutu*; ia membuat produk menjadi lebih baik. Bab ini mengajukan pertanyaan yang lebih lugas, yang cepat atau lambat harus dijawab setiap produk: siapa yang membayar semua ini, dan mengapa mereka mau?

Pertanyaan itu perlu dijawab dengan hati-hati karena jawaban yang mudah justru menjebak. Platform yang memiliki komunitas dan koin, secara prinsip, dapat menghasilkan uang dengan belasan cara: iklan, menjual data, menjual token, menarik biaya per pindaian, menagih agensi per klien, atau memotong setiap transfer. Masing-masing cara itu pernah dicoba seseorang di suatu tempat. Sebagian besar di antaranya akan bertentangan dengan prinsip yang sudah ditegakkan bab-bab sebelumnya: data merek tetap berada di server Profitina sendiri (Bab 6), komunitas dirancang untuk menghargai kontribusi yang berguna, bukan perhatian (Bab 4), dan siklus umpan balik memberi kredit kepada orang, bukan memanen mereka (Bab 7).

Jadi, model bisnis bukan pokok bahasan terpisah yang ditempelkan pada rekayasa perangkat lunak. Ia adalah nilai-nilai yang sama, diungkapkan dalam bentuk harga dan batas. Kita akan melihat bahwa model ini sengaja dibuat kecil: dua tingkat, satu harga, satu jalur pembayaran, dua cara membayar, tanpa iklan. Kita juga akan melihat bahwa ukuran "kecil" itu dicapai dengan membuang, bukan dengan tidak pernah menambah, dan bahwa pembuangan itu didorong oleh pengukuran.

Bab ini ditulis untuk tiga pembaca sekaligus. Pembaca pertama adalah pemilik usaha kecil yang sedang menimbang apakah Pro sepadan dengan Rp50.000. Pembaca kedua adalah mahasiswa

yang ingin memahami kasus monetisasi nyata, lengkap dengan keberhasilan dan kekecewaannya yang diceritakan apa adanya. Pembaca ketiga adalah pemrogram yang suatu hari harus mengimplementasikan kuota, langganan, dan perpanjangan, dan ingin menyetel jamnya dengan benar sejak percobaan pertama. Paruh pertama bab ini terutama berbicara kepada dua pembaca pertama; praktikum di Subbab 8.10 sampai 8.12 berbicara kepada pembaca ketiga, tetapi ditulis agar dua pembaca lainnya pun dapat mengikutinya.

8.2 Apa Itu Model Bisnis

Istilah *model bisnis* dipakai begitu longgar sehingga perlu kita tetapkan maknanya sebelum menggunakannya.

> **DEFINISI.** *Model bisnis* menjawab empat pertanyaan tentang sebuah produk: *nilai apa* yang diciptakannya, *bagi siapa*, *siapa yang membayar* nilai itu dan *bagaimana caranya*, serta *berapa biayanya* untuk terus menciptakan nilai tersebut. Sebuah model bisnis berkelanjutan apabila, dari waktu ke waktu, apa yang dibayarkan menutup apa yang dibelanjakan.

Definisinya pendek, dan setiap bagiannya penting.

Nilai apa. Bagi Profitina, Bab 3 telah menguraikan nilainya: skor kesiapan 0 sampai 100 yang memperkirakan seberapa besar kemungkinan halaman sebuah merek dipahami dan dikutip oleh mesin jawaban AI, daftar perbaikan yang diurutkan menurut prioritas, draf konten yang menutup celah, dan pemantauan yang menunjukkan apakah keadaan membaik.

Bagi siapa. Platform ini melayani usaha dari segala ukuran, kreator, peneliti dan organisasi, serta komunitas orang yang turut memperbaikinya. Tidak semua orang itu akan membayar, dan model yang sehat tidak berpura-pura bahwa mereka akan membayar.

Siapa yang membayar, dan bagaimana. Inilah inti bab ini. Dalam kasus Profitina: sebagian pengguna membayar Pro, dalam Rupiah melalui QRIS atau dalam PROFIT yang mereka peroleh atau beli.

Berapa biayanya. Platform yang menjalankan model AI-nya sendiri di perangkat kerasnya sendiri (Bab 6) memiliki struktur biaya yang tidak lazim bagi produk muda: tidak ada tagihan per permintaan dari penyedia AI, tetapi ada mesin sungguhan, listrik sungguhan, bandwidth sungguhan, dan jam kerja rekayasa serta moderasi yang sungguhan.

8.2.1 Freemium dalam satu gambaran

Model Profitina termasuk keluarga yang disebut *freemium*: tingkat gratis yang dapat dipakai siapa saja tanpa batas waktu, dan tingkat berbayar dengan kapasitas atau fitur yang lebih banyak. Kata itu gabungan dari “free” (gratis) dan “premium”. Ini model banyak alat yang sudah dipakai pembaca setiap hari: penyimpanan awan dengan jatah gratis, layanan musik dengan paket gratis, dan paket perkantoran dengan edisi gratis.

Freemium berhasil bila tiga hal benar pada saat yang sama. Tingkat gratis harus cukup baik sehingga orang datang, bertahan, dan bercerita kepada orang lain. Tingkat berbayar harus jelas lebih baik bagi mereka yang membutuhkan lebih. Dan biaya melayani pengguna gratis harus tetap cukup kecil sehingga minoritas yang membayar sanggup menanggungnya.

Cara praktis untuk melihat hitungannya adalah satu rumus. Misalkan N adalah banyaknya akun aktif dalam periode 30 hari, c adalah *conversion rate* (tingkat konversi, yaitu bagian akun aktif yang membayar), dan P adalah harga satu masa berbayar. Pendapatan R untuk periode itu adalah

$$R = N \cdot c \cdot P \tag{8.1}$$

Persamaan (8.1) tampak remeh, dan justru di situlah nilainya: rumus itu memperlihatkan bahwa hanya ada tiga tuas. Kita dapat memperbanyak pengguna (N), meyakinkan bagian yang lebih besar dari mereka untuk membayar (c), atau menaikkan harga (P). Setiap perdebatan harga adalah perdebatan tentang tuas mana yang ditarik, dan dengan korban apa pada dua tuas lainnya. Menaikkan P biasanya menurunkan c. Memperkaya tingkat gratis biasanya menaikkan N tetapi dapat menurunkan c, karena makin sedikit orang yang perlu meningkatkan langganan.

Seberapa besar c dalam praktik? Analisis *Harvard Business Review* yang banyak dikutip tentang usaha freemium melaporkan bahwa tingkat konversi kebanyakan perusahaan "berkisar antara 2% dan 5%", dan mengingatkan bahwa 1% kemungkinan terlalu rendah [1]. Analisis yang sama menambahkan pengamatan kedua yang penting bagi platform komunitas: seorang pengguna gratis biasanya bernilai 15% sampai 25% dari nilai pelanggan berbayar, sebagian besar melalui rujukan [1]. Pengguna gratis bukan beban yang harus ditekan serendah mungkin; mereka bagian dari cara produk bertumbuh.

> **CONTOH.** Misalkan, sekadar sebagai ilustrasi, sebuah alat GEO memiliki $N = 2{,}000$ akun aktif dalam sebulan dan mengonversi $c = 3\%$ di antaranya pada harga $P = \text{Rp}50{,}000$. Persamaan (8.1) memberikan $R = 2{,}000 \times 0.03 \times 50{,}000 = \text{Rp}3{,}000{,}000$ per 30 hari. Jika tingkat konversinya hanya 1%, basis pengguna yang sama menghasilkan Rp1,000,000. Angka-angka ini rekaan untuk contoh hitungan, bukan angka Profitina; intinya adalah betapa cepat c mendominasi hasil ketika N tidak besar. (Di dalam rumus, pemisah ribuan ditulis dengan koma dan desimal dengan titik, mengikuti kaidah notasi matematis yang sama di kedua edisi buku ini.)

Ingat contoh ini. Ia menjelaskan mengapa platform muda tidak dapat berharap hidup dari langganan saja pada bulan-bulan pertamanya, dan mengapa ia harus berhati-hati menghabiskan waktu rekayasa untuk mesin pembayaran sebelum ada orang yang membayar.

8.2.2 Apa yang sengaja tidak dijual Profitina

Model bisnis ditentukan oleh apa yang ditolaknya sama banyaknya dengan oleh apa yang ditawarkannya. Tiga penolakan membentuk semua hal lain dalam bab ini.

1. **Tanpa iklan, selamanya.** Subbab 8.6 menjelaskan mengapa ini keputusan struktural, bukan soal selera.
2. **Tidak menjual data pengguna.** Riwayat pindaian bersifat pribadi bagi pemiliknya (Bab 3), dan data merek tidak pernah meninggalkan server untuk pemrosesan AI (Bab 6). Model yang menjual data akan bertentangan dengan keduanya.
3. **Tidak menjanjikan imbal hasil.** PROFIT, koin utilitas yang dibahas di Bab 12, "bukan sekuritas, bukan uang elektronik, bukan janji imbal hasil", menurut kata-kata whitepaper publiknya [2]. Tidak ada bagian model bisnis yang bergantung pada harapan bahwa nilai PROFIT akan naik, dan bab ini tidak pernah menyiratkannya.

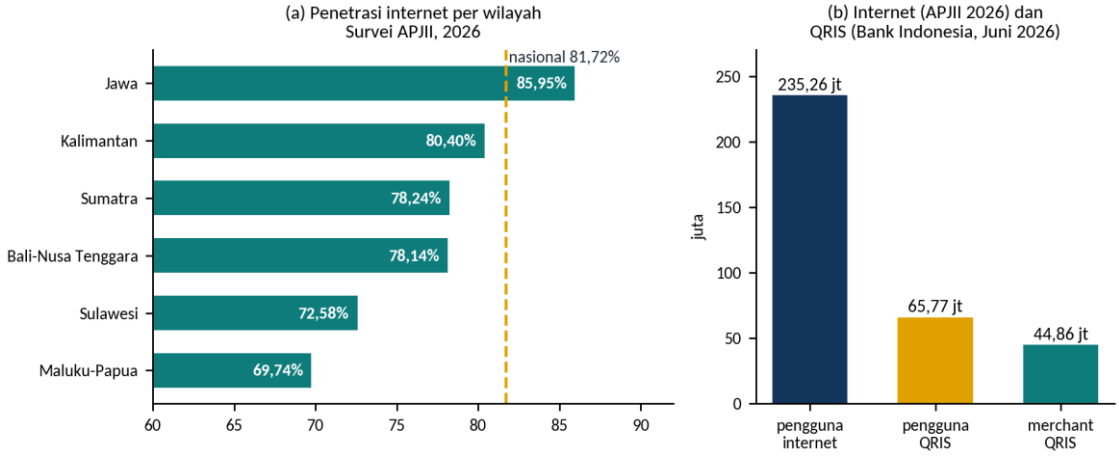
8.3 Pasar yang Dilayani Profitina

Harga hanya bermakna jika dibandingkan dengan pasar tempat ia ditetapkan. Profitina dirancang dengan mendahulukan Indonesia [2], sehingga pasar yang perlu kita pahami adalah pasar Indonesia: berapa banyak orang yang daring, bagaimana mereka membayar, dan apa yang lazim bagi pemilik usaha kecil.

8.3.1 Siapa yang daring

Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) menerbitkan survei tahunan tentang penggunaan internet. Survei 2026, yang dirilis pada Mei 2026, mencatat **235.261.078** pengguna internet dari **287.303.234** penduduk, atau penetrasi **81,72%**, naik dari 80,66% pada 2025 [3]. Itu berarti sekitar enam juta pengguna lebih banyak daripada setahun sebelumnya [4].

Rata-rata menyembunyikan geografi, dan geografi Indonesia sangat beragam. **Gambar 8.1** (a) memperlihatkan penetrasi per wilayah. Jawa memimpin dengan 85,95% dan menampung 58,24% dari seluruh pengguna; Maluku dan Papua tertinggal pada 69,74% [4]. Penetrasi di perkotaan 84,75% dan di perdesaan 78,18% [3]. Di kalangan generasi muda angkanya lebih tinggi lagi: 90,34% untuk milenial dan 89,02% untuk Generasi Z, dan 84,31% responden terutama mengakses internet melalui ponsel pintar [4].



Gambar 8.1 Pasar pada 2026. (a) Penetrasi internet per wilayah menurut survei APJII, dengan angka nasional 81,72% sebagai garis putus-putus; sumbu dimulai dari 60%. (b) Pengguna internet (APJII, 2026) di samping pengguna QRIS dan merchant QRIS (Bank Indonesia, Juni 2026).

Apa artinya bagi model bisnis? Ada tiga hal. Pertama, calon khalayaknya besar: seperempat miliar orang daring. Kedua, khalayak itu bergerak lewat ponsel: produk yang jalur peningkatan langganannya menuntut komputer meja, kartu kredit, atau rekening luar negeri akan kehilangan sebagian besar khalayak itu pada langkah pembayaran. Ketiga, khalayaknya tersebar tidak merata, sehingga harga yang terasa remeh di kantor Jakarta bisa menjadi keputusan sungguhan bagi pemilik binatu di kota yang lebih kecil. Harga harus dinilai dari sudut pandang pembaca, bukan dari sudut pandang pengembang.

8.3.2 Bagaimana orang membayar: QRIS

QRIS (Quick Response Code Indonesian Standard) adalah standar nasional Bank Indonesia untuk pembayaran dengan kode QR. Gagasan pokoknya adalah interoperabilitas: satu kode QR yang dicetak di warung dapat dibayar dari aplikasi bank atau dompet elektronik mana pun yang ikut serta, apa pun yang kebetulan dipakai pelanggan. Sebelum QRIS, sebuah toko sering memajang deretan kode QR yang berbeda, satu untuk setiap dompet; sesudah QRIS, satu kode melayani semuanya.

Skala QRIS pada 2026 mengesankan. Bank Indonesia melaporkan **65,77 juta** pengguna QRIS dan **44,86 juta** merchant yang menerimanya per Juni 2026, dan 96,68% di antaranya adalah usaha mikro, kecil, dan menengah [5]. Pada semester pertama 2026 saja, QRIS mengantarkan **12,55 miliar** transaksi senilai **Rp600,7 triliun**, kira-kira dua kali lipat volume semester pertama 2025

[6]. **Gambar 8.1** (b) menyandingkan jumlah pengguna dan merchant itu dengan jumlah pengguna internet.

Dua hal mengikutinya bagi Profitina. Pertama, QRIS sudah menjadi cara yang diharapkan usaha kecil di Indonesia untuk membayar dan dibayar; Deputi Gubernur Bank Indonesia sendiri menyebutnya "gerbang utama, pintu masuk, bagi UMKM ke dalam ekosistem keuangan digital" [5]. Pemilik binatu yang membayar pemasok dengan memindai kode akan merasa wajar membayar Pro dengan cara yang sama. Kedua, QRIS bekerja dalam Rupiah, sehingga harga dapat dinyatakan dalam mata uang yang dipakai orang untuk berpikir, tanpa kurs dan tanpa jaringan kartu di tengahnya.

> **WAWASAN KUNCI:** Memilih jalur pembayaran berarti memilih khalayak. Kasir yang hanya menerima kartu diam-diam menyingkirkan orang yang tidak memiliki kartu atau enggan memakainya secara daring. QRIS merangkul puluhan juta orang yang sudah membayar dengan cara ini setiap hari.

8.3.3 Pembanding harga bagi usaha kecil

Pemilik usaha kecil jarang membandingkan langganan perangkat lunak dengan perangkat lunak lain. Mereka membandingkannya dengan barang yang sudah biasa mereka beli. Rp50.000 setara dengan beberapa piring nasi dan minuman, isi ulang paket data yang kecil, atau pekerjaan cetak yang sederhana. Itu bukan kebetulan angka; itulah yang membuat harganya menjadi keputusan yang dapat diambil satu orang di tempat, tanpa rapat atau pos anggaran. Subbab 8.4.3 mengubah harga itu menjadi angka harian, yang merupakan cara kebanyakan pelanggan benar-benar memikirkannya.

8.4 Dua Tingkat: Free dan Pro

Pada rilis September 2026, Profitina menawarkan tepat dua tingkat. **Tabel 8.1** menyalin apa yang tertulis pada kartu paket di aplikasi [2].

Tabel 8.1 Dua tingkat layanan sebagaimana dijelaskan kartu paket pada rilis September 2026.

	Free	Pro
Pindaian dan isi otomatis	15 per hari	Pindaian dan konten tanpa batas
Content Engine, schema, Technical Assistant	Ya	Ya
Pemantauan terjadwal	1 monitor, mingguan	50 monitor harian, dengan peringatan
Riwayat	30 hari	Riwayat dan tren lengkap, ekspor CSV
Masa dan harga	Tanpa biaya	30 hari, Rp50.000 lewat QRIS, atau ditukar dengan PROFIT
Perpanjangan	Tidak berlaku	Perpanjangan otomatis dari saldo PROFIT, pengingat paling lambat 24 jam sebelum berakhir

Bacalah **Tabel 8.1** pelan-pelan, karena bentuknya mengandung filosofi.

8.4.1 Apa isi Free, dan mengapa begitu banyak

Tingkat Free bukan demo. Di dalamnya ada AI Visibility Engine secara penuh, Content Engine, bantuan schema.org, dan Technical Assistant. Pengguna Free dapat memindai, membaca daftar perbaikan, membuat draf *answer-first* (jawaban lebih dulu) dengan FAQ dan JSON-LD, lalu menerapkannya. Tidak ada hal pokok untuk *memperbaiki* situs yang disembunyikan di balik pembayaran.

Mengapa memberi sebanyak itu? Karena unsur "bagi siapa" dalam definisi Subbab 8.2. Banyak usaha yang paling membutuhkan bantuan visibilitas AI justru paling tidak mampu membayarnya. Tingkat Free yang benar-benar memperbaiki keadaan menyebarkan praktik yang baik, menghasilkan pengguna puas yang bercerita kepada orang lain, dan menghasilkan umpan balik yang menjadi bahan bakar siklus Bab 7. Hal itu juga sejalan dengan sikap platform bahwa nilainya terletak pada perbaikan itu sendiri, bukan pada akses ke sebuah angka.

Yang dibatasi Free adalah *volume* dan *kesinambungan*: 15 pindaian sehari, satu monitor yang memindai ulang setiap minggu, dan riwayat 30 hari. Batas-batas itu terasa tepat bagi pengguna yang pekerjaannya sudah menjadi rutin: pekerja lepas yang memeriksa banyak situs, usaha yang ingin melacak beberapa halaman setiap hari, atau tim yang harus melaporkan tren selama satu kuartal.

Dua hal lain gratis untuk semua orang, selamanya, dan tidak muncul di kartu paket mana pun: hak-hak yang dibahas Bab 11 (mengekspor data, mengoreksinya, menghapus akun) serta fitur keselamatan seperti Link Safety dan pemulihan akun [2]. Model bisnis tidak boleh menarik bayaran atas hak seseorang.

8.4.2 Apa tambahan Pro

Pro menghapus batas pindaian, menaikkan pemantauan dari satu monitor mingguan menjadi lima puluh monitor harian dengan peringatan, dan menyimpan riwayat lengkap dengan grafik tren serta ekspor CSV. Semuanya fitur *kapasitas*: memungkinkan pengguna melakukan lebih banyak hal yang sudah bisa dilakukan di Free, lebih sering, dan dalam rentang waktu lebih panjang. Tidak satu pun membuka skor rahasia yang lebih baik. Situs pengguna Pro diukur oleh mesin yang persis sama dengan situs pengguna Free.

Pilihan rancangan ini perlu dinyatakan terang-terangan: tingkat layanan mengubah *seberapa banyak* dan *seberapa sering*, tidak pernah *seberapa jujur*. Kalau pengguna Pro mendapat skor yang lebih ramah, skor itu tidak lagi bermakna bagi siapa pun.

8.4.3 Harga per hari

Harganya Rp50.000 untuk masa 30 hari. Membagi dengan panjang masa itu memberikan harga harian p:

$$p = \frac{\text{Rp50,000}}{\text{30 hari}} \approx \text{Rp1,666.67 per hari} \tag{8.2}$$

Persamaan (8.2) adalah kalimat yang berguna untuk disampaikan kepada pelanggan: Pro berharga kurang dari Rp1.700 sehari. Praktikum di Subbab 8.10 menghitung angka ini dari tabel paket, bukan mengetiknya, sehingga bila harga suatu saat berubah, penjelasannya ikut berubah.

Cara kedua membaca harga adalah per pindaian. Jika pengguna Pro rata-rata menjalankan n pindaian sehari, biaya per pindaian adalah p/n. Pada $n = 30$, cukup dua kali jatah Free, setiap pindaian berharga sekitar Rp56. Makin banyak pengguna bergantung pada produk, makin murah setiap pemakaian, dan itulah bentuk yang tepat bagi alat yang dimaksudkan untuk dipakai secara rutin.

8.4.4 Aturan untuk kartu paket

Tim bekerja dengan aturan yang mudah diucapkan dan ternyata mudah dilanggar: **jangan pernah menulis janji fitur yang belum dibangun di samping harga berbayar**. Kartu paket

yang berbunyi “segera hadir: akses API” di samping “Rp50.000” mengundang orang membayar sesuatu yang belum ada. Jika fitur itu terlambat, atau tidak pernah datang, produk pada dasarnya telah menerima uang dengan deskripsi yang keliru.

Aturan itu juga berakibat praktis bagi buku ini. Semua isi **Tabel 8.1** sudah berjalan pada rilis September 2026. Ketika bab-bab berikutnya membahas butir peta jalan, seperti PROFIT mainnet di Bab 12, butir itu diberi label peta jalan dan tidak pernah dikaitkan dengan harga.

8.5 Mengapa Tingkat Agency Dihapus

Hingga awal Agustus 2026 ada tingkat ketiga, Agency, di atas Pro. Agency menambahkan tiga hal: pengelolaan **multi-klien** (satu akun menangani merek banyak klien), **akses API** (program, bukan orang, yang memanggil platform), dan **white label** (laporan membawa merek agensi, bukan merek Profitina). Pada 6–7 Agustus 2026 tingkat itu dihapus, sehingga tersisa dua [2]. **Gambar 8.2** memperlihatkan keadaan sebelum dan sesudahnya.

Gambar 8.2 Struktur tingkat layanan sebelum dan sesudah 6–7 Agustus 2026. Ketiga fitur Agency sudah terbukti berjalan; tingkat itu dihapus demi kesederhanaan, dan sikap tanpa iklan selamanya dipertahankan.

Fakta paling menarik tentang penghapusan ini justru alasan yang *tidak* diberikan. Agency tidak dihapus karena fiturnya rusak. Whitepaper mencatat bahwa ketiganya “sudah terbukti berjalan sebelum dihapus” [2]. Tingkat itu dihapus “demi kesederhanaan”. Hal ini perlu dijelaskan, karena membuang fitur yang berjalan demi kesederhanaan, pada pandangan pertama, terdengar seperti pemborosan.

8.5.1 Biaya tersembunyi sebuah tingkat

Tingkat layanan bukan sekadar harga di kartu. Ia adalah seperangkat aturan yang harus dihormati setiap bagian sistem. Setiap fitur harus bertanya, pada saat dijalankan, “pengguna ini berada di tingkat apa, dan apa yang boleh saya lakukan untuknya?” Setiap pertanyaan semacam itu adalah tempat jawaban bisa keliru.

Kita dapat membuat biaya itu konkret. Misalkan produk memiliki F fitur yang perilakunya bergantung pada tingkat, dan T tingkat. Banyaknya kombinasi fitur-dan-tingkat yang harus dirancang, didokumentasikan, diuji, dan dijelaskan adalah

$$C = T \times F \tag{8.3}$$

Dengan $F = 12$ fitur yang bergantung pada tingkat, dua tingkat memberikan $C = 24$ kombinasi dan tiga tingkat memberikan $C = 36$, bertambah separuhnya, menurut Persamaan (8.3). Biaya

sesungguhnya bahkan lebih buruk daripada linear, karena tingkat berinteraksi dengan waktu: apa yang terjadi bila pengguna Agency turun ke Pro sambil memegang dua puluh merek klien dan kunci API yang aktif? Klien mana yang tetap ada? Apakah kuncinya berhenti seketika atau pada akhir masa? Setiap perpindahan antartingkat adalah seperangkat pertanyaan lagi, dan dengan T tingkat ada $T(T-1)$ kemungkinan perpindahan berarah: dua untuk dua tingkat, enam untuk tiga tingkat.

Bagi tim kecil, setiap kombinasi dan perpindahan itu bersaing merebut jam yang sama dengan produk inti. Menghapus Agency tidak sekadar menghapus tiga fitur; ia menghapus dua belas kombinasi, empat perpindahan, satu paragraf syarat hukum, satu kolom kartu paket, dan satu kategori penuh pertanyaan dukungan.

8.5.2 Kesederhanaan bagi pelanggan

Alasan kedua ada di sisi pelanggan. Memilih di antara dua pilihan itu cepat: "Apakah saya memerlukan lebih dari yang diberikan Free?" Memilih di antara tiga mengundang perbandingan, keraguan, dan kekhawatiran salah pilih. Bagi produk yang penggunanya sering kali usaha kecil yang baru pertama kali bersentuhan dengan visibilitas AI, satu pertanyaan yang jelas lebih ramah daripada sebuah matriks.

8.5.3 Bagaimana dengan agensi sekarang

Agensi tetap diterima. Agensi yang mengelola beberapa klien dapat memakai satu akun untuk setiap klien, atau satu akun Pro dengan hingga lima puluh monitor harian. Yang tidak ada lagi adalah alur kerja khusus bagi mereka. Jika muncul permintaan nyata dan terukur akan fitur multi-klien, itu akan menjadi keputusan baru tentang produk baru, bukan pemulihan tingkat lama. Subbab 8.8 menjelaskan bagaimana keputusan semacam itu seharusnya diambil.

> **PERINGATAN:** Materi pemasaran lama yang menyebut tingkat Agency, atau harga PROFIT untuk Agency, menggambarkan produk sebelum Agustus 2026. Itu bukan pratinjau masa depan. Bila sumber-sumber berbeda pendapat, kartu paket aplikasi yang berlaku saat ini adalah rujukan yang dapat dipercaya.

8.6 Tanpa Iklan, Selamanya

Banyak layanan gratis membiayai dirinya dengan iklan. Profitina telah berkomitmen **tanpa iklan, selamanya** [2]. Hal ini sering dibaca sebagai selera gaya, keinginan akan layar yang bersih. Sesungguhnya lebih dari itu.

8.6.1 Konflik kepentingan

Usaha periklanan memperoleh uang sebanding dengan perhatian: makin lama orang melihat, makin banyak iklan yang mereka lihat. Karena itu, insentifnya mengarah pada upaya menahan orang di platform, dan pada jenis konten serta rancangan yang berhasil menahan mereka.

Nilai Profitina bagi pengguna justru mengarah ke sebaliknya. Pindaian yang baik memberi tahu apa yang harus diperbaiki lalu membiarkan Anda pergi memperbaikinya di situs sendiri. Daftar perbaikan yang baik itu pendek. Unggahan komunitas yang baik menjawab pertanyaan dan menutup utas. Jika platform memperoleh uang dari perhatian, setiap hasil baik itu, dari sudut pandang pendapatan, adalah kerugian. Itulah konflik kepentingan yang tertanam dalam model, dan konflik kepentingan yang tertanam dalam model tidak lama tinggal sebagai teori.

Langganan membalik insentif itu. Pengguna Pro memperpanjang karena produk telah memperbaiki visibilitasnya bulan lalu. Platform memperoleh uang dengan menjadi berguna, bukan dengan membuat orang lengket.

8.6.2 Argumen privasi

Iklan dalam skala besar bergantung pada penargetan, dan penargetan bergantung pada data tentang orang: apa yang mereka lihat, apa yang mereka beli, siapa mereka. Bab 11 menjelaskan pendekatan Profitina terhadap data pribadi di bawah UU PDP No. 27/2022: kumpulkan sesedikit mungkin, simpan di server sendiri, berikan hak swalayan kepada orang. Usaha periklanan akan menarik ke arah yang berlawanan pada setiap butir tersebut. Menolak iklan adalah prasyarat untuk menepati janji privasi, bukan keputusan yang terpisah.

8.6.3 Argumen kredibilitas

Terakhir, platform yang memberi tahu usaha seberapa tepercaya konten mereka di mata mesin AI harus tampak tepercaya pula. Dasbor GEO yang dikelilingi iklan spanduk “dijamin halaman satu” akan meruntuhkan pesan yang hendak disampaikannya sendiri.

> **WAWASAN KUNCI:** “Tanpa iklan” adalah pernyataan tentang insentif. Ia menyelaraskan apa yang diperoleh platform dengan apa yang diperoleh pengguna, melindungi sikap privasi di Bab 11, dan menjaga agar saran produk tetap kredibel.

8.7 Membayar Pro: QRIS dan PROFIT

Ada dua cara membayar satu masa Pro: dalam Rupiah melalui QRIS, atau dengan menukarkan PROFIT. **Tabel 8.2** memisahkan apa yang sudah berjalan dari yang belum, karena dalam segala hal yang menyangkut uang perbedaan itu tidak boleh sampai terlewatkan.

Tabel 8.2 Apa yang sudah berjalan dan apa yang belum, untuk semua bagian model bisnis yang melibatkan pembayaran.

Kemampuan	Status pada rilis September 2026
Pro lewat QRIS, Rp50.000 per 30 hari	Berjalan
Pro dengan menukarkan PROFIT	Berjalan
Perpanjangan otomatis dari saldo PROFIT, pengingat paling lambat 24 jam sebelum berakhir	Berjalan
Membeli PROFIT lewat QRIS (Rp50.000 sampai Rp1.000.000)	Berjalan; dikreditkan setelah administrator memverifikasi pembayaran
Membelanjakan PROFIT untuk tambahan kuota pindaian dan Academic Citation Scan	Berjalan
Menjual atau mencairkan PROFIT kembali ke Rupiah	Dinonaktifkan hingga mainnet dan kejelasan regulasi
PROFIT mainnet dan bursa non-kustodian	Peta jalan (Fase 3, bergantung pada audit dan perizinan)

8.7.1 QRIS, langkah demi langkah

Dari sudut pandang pengguna, membayar dengan QRIS sama saja dengan membayar di toko mana pun. Pengguna memilih Pro, aplikasi menampilkan kode QRIS dan jumlahnya, pengguna memindainya dengan aplikasi bank atau dompet yang sudah biasa dipakai (DANA, GoPay, OVO,

ShopeePay, atau aplikasi mobile banking), lalu mengonfirmasi. Setelah pembayaran terkonfirmasi, Pro dimulai dan berjalan selama 30 hari.

Di balik pengalaman sederhana itu terdapat sepotong logika kecil tetapi penting: sebuah *pesanan* yang berpindah dari satu keadaan ke keadaan lain. Pesanan dibuat, menunggu pembayaran, diperiksa, lalu menjadi masa aktif atau ditutup. Pesanan juga bisa kedaluwarsa begitu saja jika tidak ada yang membayar. Subbab 8.11 membangun logika ini sebagai program, dan **Gambar 8.5** memberikan pratinjaunya.

Pada rilis September 2026, PROFIT yang dibeli lewat QRIS dikreditkan setelah administrator memverifikasi pembayaran [2]. Pemeriksaan oleh manusia pada saat uang berubah menjadi sesuatu yang bernilai memang lebih lambat daripada gerbang pembayaran yang sepenuhnya otomatis, tetapi murah dibangun, mudah diaudit, dan pantas bagi platform yang volume pembeliannya masih kecil. Subbab 8.8 menjelaskan mengapa ini tahapan yang disengaja, bukan jalan pintas.

8.7.2 Menukarkan PROFIT

Cara kedua membayar adalah menukarkan PROFIT, koin utilitas yang diperoleh pengguna dengan menyumbangkan umpan balik yang berguna dan aktivitas komunitas (Bab 7) atau dibeli lewat QRIS. Aplikasi menampilkan harga dalam PROFIT sebelum pengguna mengonfirmasi. Ketika pengguna menukarkannya, PROFIT yang dibelanjakan **dibakar**: ia keluar dari peredaran, bukan masuk ke dompet perusahaan [2]. Bab 12 menjelaskan mengapa pembakaran penting bagi pasokan koin; di sini cukup dicatat bahwa penukaran adalah pemakaian utilitas, bukan perdagangan.

Penukaran memiliki sifat yang menyenangkan bagi model bisnis: pengguna yang memperoleh PROFIT dengan membantu platform dapat mengubah sumbangannya menjadi waktu Pro. Produk menghargai anggota komunitasnya yang paling berguna dengan kapasitas yang sama dengan yang dibayar orang lain dalam Rupiah.

8.7.3 Perpanjangan otomatis dan pengingat

Pro dapat memperpanjang dirinya pada akhir masa dengan menukarkan PROFIT dari saldo pengguna, jika pengguna menyalakan perpanjangan otomatis dan saldonya mencukupi. Pengingat datang **paling lambat 24 jam sebelum masa berakhir** [2]. Kedua aturan itu ada demi kepentingan pengguna: perpanjangan otomatis mencegah celah yang tidak disengaja dalam pemantauan harian, dan pengingat menjamin tidak ada yang diperpanjang diam-diam.

Menyusun perpanjangan dengan benar lebih halus daripada kelihatannya. Jika perpanjangan diproses beberapa jam terlambat, karena mesin sedang sibuk atau baru dihidupkan ulang, sejak saat mana masa baru berjalan? Sejak waktu berakhir yang lama, atau sejak kode perpanjangan dijalankan? Jawaban yang adil adalah sejak waktu berakhir yang lama: pengguna membayar tiga puluh hari yang menyusul tiga puluh hari sebelumnya, bukan tiga puluh hari yang menyusul kebetulan jadwal server. Dengan menuliskan e_k untuk waktu berakhir masa ke-k,

$$e_{k+1} = e_k + 30 \text{ hari} \tag{8.4}$$

terlepas dari kapan kode perpanjangan benar-benar berjalan. Persamaan (8.4) tampak terlalu jelas untuk ditulis, tetapi versi pertama kode praktikum di Subbab 8.11 justru salah di sini, dan sebuah uji menangkapnya. Subbab 8.12 menceritakan kisah itu.

> **PERINGATAN:** Tidak ada isi subbab ini yang merupakan nasihat keuangan. Pada rilis September 2026, PROFIT adalah koin utilitas di Solana devnet tanpa nilai moneter nyata; ia bukan sekuritas, bukan uang elektronik, dan bukan janji imbal hasil [2]. Menukarkannya dengan Pro adalah pemakaian sebuah fitur produk.

8.8 Ukur Minat sebelum Membangun Pembayaran

Subbab ini menceritakan kisah yang paling tidak gemerlap dalam bab ini, dan barangkali yang paling berguna.

8.8.1 Temuan nol pembelian

Pada Agustus 2026 tim menganalisis bagaimana PROFIT sebenarnya diperoleh dan dipakai. Temuannya, yang dicatat dalam whitepaper publik, lugas: **nol pembelian PROFIT sepanjang seluruh riwayat produksi** [2]. Orang memperoleh PROFIT, membelanjakannya, dan memakai komunitas, tetapi belum ada seorang pun yang membelinya dengan Rupiah.

Tim yang kurang jujur akan mengubur angka itu, atau menjelaskannya dengan tanggal peluncuran, atau terus membangun fitur pembayaran baru dengan harapan fitur berikutnya membuka permintaan. Profitina melakukan sesuatu yang lebih baik: menuliskan angka itu dalam dokumen publik dan menarik aturan darinya.

> **DEFINISI.** *Ukur minat sebelum membangun pembayaran* berarti: sebelum menanamkan upaya rekayasa pada cara menerima uang untuk sesuatu, kumpulkan bukti bahwa orang mau membayarnya, dengan mekanisme termurah yang menghasilkan bukti nyata.

8.8.2 Mengapa aturan ini benar

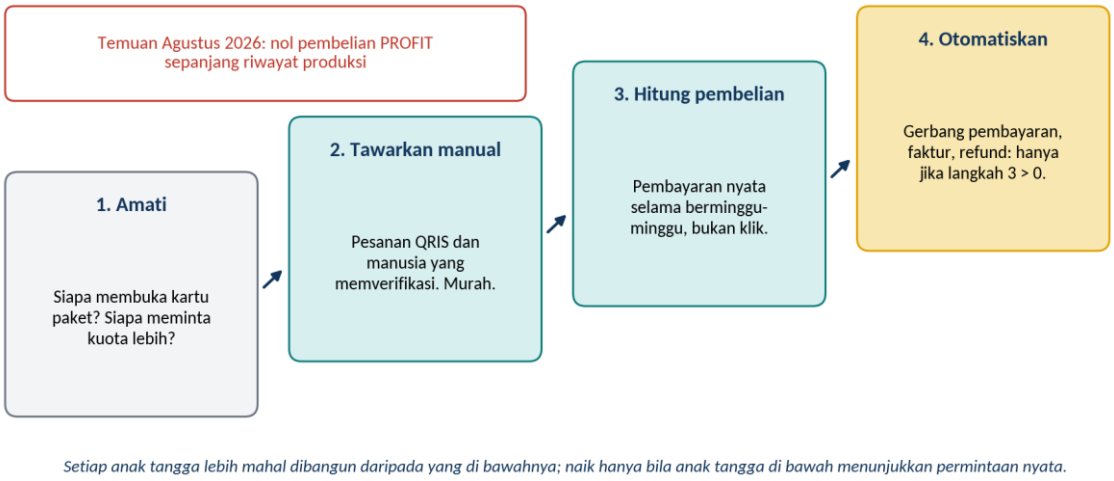
Mesin pembayaran mahal dengan cara yang tidak terlihat dari luar. Kasir yang sepenuhnya otomatis membutuhkan integrasi dengan penyedia pembayaran, rekonsiliasi antara yang dibayar dan yang dikreditkan, penanganan pembayaran sebagian, notifikasi ganda, pengembalian dana, kuitansi, catatan pajak, dan jalur dukungan pelanggan untuk setiap kemungkinan gagal. Masing-masing adalah kode yang harus diuji dan dirawat, dan masing-masing adalah tempat kesalahan merugikan uang sungguhan milik seseorang.

Jika tidak ada yang membeli, seluruh upaya itu tidak menghasilkan apa-apa. Lebih buruk lagi, ia dapat menyembunyikan masalah yang sebenarnya. Ketika sebuah fitur tidak laku, pertanyaannya biasanya “apakah ini layak dibayar?”, bukan “apakah kasirnya kurang mulus?”. Kasir yang indah untuk sesuatu yang tidak diinginkan siapa pun tetap tidak laku.

Gagasan ini tidak baru. Buku *The Lean Startup* karya Eric Ries membangun metode di sekelilingnya: perlakukan setiap keputusan produk sebagai eksperimen, bangun hal terkecil yang dapat menguji asumsi paling berisiko, ukur apa yang dilakukan orang sungguhan, dan belajarlah sebelum membangun lebih banyak [7]. Asumsi paling berisiko dalam fitur pembayaran bukan “bisakah kita menerima uang?”, melainkan “adakah yang mau membayar?”.

8.8.3 Tangga bukti

Gambar 8.3 memperlihatkan aturan itu sebagai tangga. Setiap anak tangga lebih mahal dibangun daripada anak tangga di bawahnya, dan kita naik hanya bila anak tangga di bawah menunjukkan permintaan nyata.



Gambar 8.3 Ukur minat sebelum membangun pembayaran, sebagai tangga. Setiap anak tangga hanya dibangun bila anak tangga di bawahnya menunjukkan permintaan nyata; temuan Agustus 2026 berupa nol pembelian PROFIT menahan platform di anak tangga bawah.

1. **Amati.** Siapa membuka kartu paket? Siapa mencapai batas harian? Siapa meminta kuota lebih? Langkah ini hampir tanpa biaya dan merupakan bukti jujur tentang *kebutuhan*, meskipun belum tentang kesediaan membayar.
2. **Tawarkan secara manual.** Mungkinkan pembayaran dengan mekanisme andal yang paling sederhana: pesanan QRIS dan manusia yang memverifikasi. Pembelian PROFIT pada rilis September 2026 berada di tahap ini. Tahap ini murah dibangun, mudah diaudit, dan menghasilkan satu-satunya bukti yang penting: pembayaran sungguhan.
3. **Hitung pembelian.** Amati pembayaran nyata selama berminggu-minggu, bukan klik atau pujian. Orang mengatakan banyak hal dalam survei; yang mereka bayar itulah yang mereka hargai.
4. **Otomatiskan.** Hanya bila pembelian sudah ada, dan langkah manual sudah menjadi hambatan, tanamkan modal pada gerbang pembayaran, faktur otomatis, dan pengembalian dana.

Perhatikan arti temuan nol pembelian pada tangga ini. Temuan itu tidak berarti produk gagal. Artinya, langkah 3 menghasilkan nol, dan karena itu langkah 4 belum boleh dibangun. Uang dan waktu yang semula akan dipakai untuk mengotomatiskan pembayaran dialihkan ke produk itu sendiri.

8.8.4 Aturan yang sama di tempat lain

Aturan ini tidak khusus untuk PROFIT. Pada pertengahan September 2026 tim mengevaluasi usulan untuk menawarkan bantuan AI gratis kepada sekitar 250 mahasiswa angkatan pertama perguruan tinggi yang memakai platform, mengukur biaya dan manfaatnya, lalu **membekukannya** setelah pengukuran. Fitur yang terdengar murah hati tetaplah fitur yang harus dibenarkan oleh bukti, dan disiplin yang sama yang mencegah kasir yang tak terpakai juga mencegah hadiah yang tak terjangkau.

> **WAWASAN KUNCI:** Nol yang jujur adalah informasi. Temuan nol pembelian tidak mengakhiri model bisnis; ia memberi tahu tim di mana tidak perlu menghabiskan bulan berikutnya.

8.9 Kuota Harian dan Aturan 00.00 WIB

Tingkat Free mengizinkan 15 pindaian *per hari*, dan perolehan PROFIT pun memiliki batas harian. “Per hari” terdengar seperti frasa paling sederhana dalam bab ini. Kenyataannya, frasa inilah yang

paling mungkin diimplementasikan secara keliru, dan Profitina mempelajarinya dengan cara yang dialami kebanyakan tim: pernah keliru sekali.

8.9.1 Hari milik siapa?

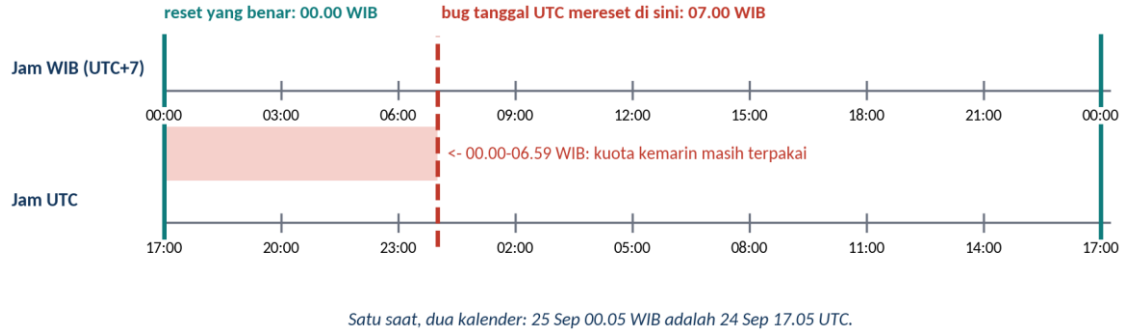
Hari bukan sesuatu yang universal. Hari adalah rentang 24 jam di antara dua tengah malam, dan tengah malam tiba pada saat yang berbeda di tempat yang berbeda. Komputer biasanya menyimpan waktu sebagai *Coordinated Universal Time* (UTC), yaitu waktu pada garis bujur nol tanpa waktu musim panas. Waktu Indonesia Barat (WIB), yang dipakai di Jawa, Sumatra, serta Kalimantan bagian barat dan tengah, tujuh jam lebih awal: WIB = UTC + 7 jam, sepanjang tahun, tanpa waktu musim panas. Basis data zona waktu yang dipakai komputer menamai zona ini `Asia/Jakarta` [8].

Kuota harian Profitina diatur ulang pada **00.00 WIB** [2]. Itulah tengah malam yang dijalani penggunanya. Pemilik binatu di Surabaya yang menghabiskan 15 pindaiannya pada Rabu malam berharap mendapat 15 lagi ketika membuka aplikasi pada Kamis pukul enam pagi.

8.9.2 Jebakan tanggal UTC

Inilah jebakannya. Kode yang paling mudah ditulis untuk "berapa pindaian hari ini?" mengambil waktu sekarang sebagaimana disimpan komputer, dalam UTC, mengambil tanggal kalendernya, lalu menghitung pindaian bertanggal sama. Kode itu pendek, tampak benar, dan lolos setiap uji yang ditulis seseorang pada jam kantor di Indonesia.

Kode itu salah selama tujuh jam setiap hari. **Gambar 8.4** menunjukkan sebabnya. Pada pukul 00.05 WIB tanggal 25 September, jam UTC menunjukkan 17.05 tanggal *24 September*. Karena itu, kuota yang dihitung menurut tanggal UTC masih melihat semua pindaian hari Rabu sampai pukul 07.00 WIB hari Kamis, ketika tanggal UTC akhirnya berganti. Antara tengah malam dan pukul tujuh pagi, pengguna diberi tahu bahwa pindaiannya sudah habis, padahal menurut setiap kalender yang mereka pakai hari sudah berganti.



Gambar 8.4 Satu hari pada dua jam. Pengaturan ulang yang benar terjadi pada 00.00 WIB; kode yang memakai tanggal UTC justru mengatur ulang pada 07.00 WIB, sehingga pengguna mendapati kuota kemarin yang sudah habis selama tujuh jam pertama hari mereka.

Bug persis seperti ini, pengaturan ulang harian yang terjadi pada 07.00 WIB karena kode memakai tanggal UTC, pernah ditemukan dan diperbaiki Profitina dalam riwayatnya sendiri, dan aturan yang lahir darinya menjadi bagian dari cara tim bekerja: setiap batas harian dihitung dalam WIB.

8.9.3 Hari sebagai rumus

Kita dapat menyatakan dengan tepat apa itu "hari WIB". Misalkan t adalah suatu saat, ditulis sebagai banyaknya detik sejak epoch Unix (1 Januari 1970, 00.00 UTC), yang merupakan cara

kebanyakan sistem menyimpan waktu. Satu hari berisi 86,400 detik dan WIB lebih awal $7 \times 3{,}600 = 25{,}200$ detik daripada UTC. Nomor hari dari t dalam UTC dan dalam WIB adalah

$$d_{\text{UTC}}(t) = \left\lfloor \frac{t}{86{,}400} \right\rfloor \tag{8.5}$$

$$d_{\text{WIB}}(t) = \left\lfloor \frac{t + 25{,}200}{86{,}400} \right\rfloor \tag{8.6}$$

dengan $\lfloor x \rfloor$ berarti "bilangan bulat terbesar yang tidak melebihi x". Dua pindaian termasuk hari kuota yang sama tepat jika d_{WIB} keduanya sama. Persamaan (8.5) dan Persamaan (8.6) hanya berbeda karena geseran 25,200 detik di dalam fungsi lantai, dan geseran itulah seluruh perbedaan antara perilaku yang benar dan bug pada **Gambar 8.4**.

> **CONTOH.** Saat 25 September 2026 pukul 00.05 WIB adalah $t = 1{,}790{,}269{,}500$ detik. Maka $t/86{,}400 = 20{,}720.71\ldots$, sehingga $d_{\text{UTC}} = 20{,}720$, yaitu 24 September. Namun $(t + 25{,}200)/86{,}400 = 20{,}721.00\ldots$, sehingga $d_{\text{WIB}} = 20{,}721$, yaitu 25 September. Saat yang sama, nomor hari berbeda: aturan UTC masih menghitung pindaian hari Rabu, aturan WIB memulai hari yang baru.

Dalam kode sungguhan kita tidak menghitung fungsi lantai dengan tangan. Modul `zoneinfo` pada Python dan paket `java.time` pada Java mengenal aturan untuk `Asia/Jakarta` [8], [9], dan mengubah suatu saat menjadi tanggal WIB cukup dengan satu pemanggilan metode. Rumus-rumus di atas ada agar kita memahami apa yang dikerjakan pemanggilan metode itu, dan dapat mengenali bug-nya ketika bertemu.

8.9.4 Berapa pindaian yang tersisa

Setelah hari didefinisikan, sisa kuota menjadi sederhana. Misalkan q adalah jatah harian paket (15 untuk Free), u adalah pindaian tambahan yang dibeli pengguna sebagai *top-up* (tambahan kuota) *untuk hari WIB ini*, dan s adalah pindaian yang sudah dipakai pada hari WIB ini. Maka sisa pindaian r adalah

$$r = \max(0,\ q + u - s) \tag{8.7}$$

Fungsi max dengan nol mencegah jawabannya menjadi negatif, misalnya bila tambahan kuota sudah kedaluwarsa. Untuk Pro, q tidak terbatas dan Persamaan (8.7) tidak berlaku; program menyatakan "tanpa batas" secara eksplisit, bukan dengan angka yang sangat besar, agar tidak ada yang melihat tulisan "999.999 pindaian tersisa".

Tambahan kuota layak mendapat satu kalimat tersendiri. Pada rilis September 2026, PROFIT dapat dibelanjakan untuk tambahan kuota pindaian yang berlaku **hanya pada hari pembelian** [2]. "Hari pembelian" itu, sekali lagi, adalah hari WIB. Tambahan kuota yang dibeli pukul 23.00 WIB hanya berlaku satu jam; justru karena itulah aplikasi menampilkan harga dan masa berlakunya sebelum pengguna membayar.

8.10 Praktikum: Paket sebagai Data dan Kalkulator Kuota

Sekarang kita membangun program pertama bab ini. Program ini mengerjakan tiga hal: menyimpan kedua paket sebagai *data* alih-alih menebar angka di sepanjang kode, menghitung pindaian per hari WIB beserta tambahan kuota untuk hari yang sama, dan memperagakan bug tanggal UTC berdampingan dengan perilaku yang benar. Program ini model pembelajaran yang

dibuat untuk buku ini. Satu-satunya fakta produk di dalamnya adalah angka-angka dari kartu paket publik pada **Tabel 8.1**.

8.10.1 Rancangan

Tiga keputusan membentuk kodenya.

Paket adalah data. Sebuah rekaman `Plan` menyimpan nama, pindaian per hari, banyaknya monitor, irama monitor, panjang riwayat, penanda ekspor CSV, dan harga. Kedua paket disimpan dalam kamus (Python) atau map (Java) bernama `PLANS`. Kode yang memerlukan suatu batas mencarinya di sana; tidak ada fungsi yang memuat angka 15. Ketika harga atau batas berubah, cukup satu baris yang berubah.

"Tanpa batas" adalah nilai, bukan angka besar. Pindaian per hari dan panjang riwayat Pro bernilai `None` di Python dan `null` di Java. Setiap fungsi yang membacanya memeriksa kasus itu secara eksplisit.

Hari adalah fungsi yang dapat ditukar. `QuotaBook` menerima fungsi `day_of` yang memetakan suatu saat ke tanggal kalender. Bawaannya `wib_day`. Memberikan `utc_day` sebagai gantinya mereproduksi bug, sehingga kedua perilaku dapat dicetak berdampingan dan diuji.

8.10.2 Program Python

Kode 8.1 adalah program lengkapnya.

Kode 8.1 `tiers.py`: paket sebagai data, buku kuota yang menghitung pindaian per hari WIB, tambahan kuota untuk hari yang sama, dan bug tanggal UTC sebagai pembanding.

```
"""Free and Pro plans as data, and a quota calculator whose day
starts at 00:00 WIB (Western Indonesia Time, UTC+7)."""
from dataclasses import dataclass
from datetime import date, datetime, time, timedelta, timezone
from zoneinfo import ZoneInfo

WIB = ZoneInfo("Asia/Jakarta")


@dataclass(frozen=True)
class Plan:
    name: str
    scans_per_day: int | None      # None means unlimited
    monitors: int
    monitor_every_days: int
    history_days: int | None       # None means full history
    csv_export: bool
    price_idr: int                 # per 30 days; 0 for Free


PLANS = {
    "free": Plan("Free", 15, 1, 7, 30, False, 0),
    "pro": Plan("Pro", None, 50, 1, None, True, 50_000),
}
TERM_DAYS = 30


def wib_day(instant: datetime) -> date:
    """The calendar day in Indonesia on which this instant falls."""
    return instant.astimezone(WIB).date()


def utc_day(instant: datetime) -> date:
    """The WRONG day for a quota in Indonesia: the UTC calendar day."""
    return instant.astimezone(timezone.utc).date()


def next_reset(instant: datetime) -> datetime:
    """The next 00:00 WIB after this instant, as a UTC instant."""
    tomorrow = wib_day(instant) + timedelta(days=1)
    return datetime.combine(tomorrow, time(0), WIB).astimezone(timezone.utc)


class QuotaBook:
    """Counts one account's scans per WIB day, plus same-day top-ups."""
```

Kode 8.1 (Lanjutan) *tiers.py: paket sebagai data, buku kuota yang menghitung pindaian per hari WIB, tambahan kuota untuk hari yang sama, dan bug tanggal UTC sebagai pembanding.*

```
    def __init__(self, tier: str, day_of=wib_day):
        self.plan = PLANS[tier]
        self.day_of = day_of            # wib_day, or utc_day to show the bug
        self.scans: list[datetime] = []
        self.topups: dict[date, int] = {}

    def allowance(self, now: datetime) -> int | None:
        if self.plan.scans_per_day is None:
            return None
        return self.plan.scans_per_day + self.topups.get(self.day_of(now), 0)

    def used(self, now: datetime) -> int:
        today = self.day_of(now)
        return sum(1 for t in self.scans if self.day_of(t) == today)

    def remaining(self, now: datetime) -> int | None:
        limit = self.allowance(now)
        return None if limit is None else max(0, limit - self.used(now))

    def try_scan(self, now: datetime) -> bool:
        left = self.remaining(now)
        if left is not None and left == 0:
            return False
        self.scans.append(now)
        return True

    def add_topup(self, now: datetime, extra: int) -> None:
        """Extra scans bought today are valid for today (WIB) only."""
        day = self.day_of(now)
        self.topups[day] = self.topups.get(day, 0) + extra

    def can_add_monitor(self, active_monitors: int) -> bool:
        return active_monitors < self.plan.monitors

    def history_from(self, now: datetime) -> date | None:
        """Oldest visible day of scan history; None means everything."""
        if self.plan.history_days is None:
            return None
        return self.day_of(now) - timedelta(days=self.plan.history_days)


def price_per_day(plan: Plan) -> float:
    return plan.price_idr / TERM_DAYS
```

Kode 8.1 (Lanjutan) *tiers.py: paket sebagai data, buku kuota yang menghitung pindaian per hari WIB, tambahan kuota untuk hari yang sama, dan bug tanggal UTC sebagai pembanding.*

```
def at(text: str) -> datetime:
    """Parse 'YYYY-MM-DD HH:MM' as a WIB wall-clock time."""
    return datetime.strptime(text, "%Y-%m-%d %H:%M").replace(tzinfo=WIB)


def show(n) -> str:
    return "unlimited" if n is None else str(n)


def main() -> None:
    print("Plan    scans/day  monitors  every  history  CSV  price/30d")
    for p in PLANS.values():
        hist = "full" if p.history_days is None else f"{p.history_days} d"
        print(f"{p.name:<7} {show(p.scans_per_day):>9}  {p.monitors:>8}  "
              f"{p.monitor_every_days:>3} d  {hist:>7}  "
              f"{'yes' if p.csv_export else 'no':>3}  Rp{p.price_idr:,}")
    print(f"Pro costs Rp{price_per_day(PLANS['pro']):,.2f} per day")

    good = QuotaBook("free")                 # counts WIB days
    bad = QuotaBook("free", day_of=utc_day)  # counts UTC days
    start = at("2026-09-24 20:00")
    for i in range(15):
        t = start + timedelta(minutes=14 * i)
        good.try_scan(t)
        bad.try_scan(t)
    late = at("2026-09-24 23:45")
    print("\n15 scans on 24 Sep between 20:00 and 23:16 WIB")
    print("16th scan at 23:45 WIB allowed?", good.try_scan(late))
    print("next reset:", next_reset(late).astimezone(WIB),
          "=", next_reset(late))

    for clock in ("2026-09-25 00:05", "2026-09-25 06:59",
                  "2026-09-25 07:00"):
        now = at(clock)
        print(f"{clock} WIB  remaining (WIB day) = {good.remaining(now):>2}"
              f"   remaining (UTC day) = {bad.remaining(now):>2}")

    now = at("2026-09-25 21:00")
    for _ in range(15):
        good.try_scan(now)
    good.add_topup(now, 5)
    print("\nAfter 15 scans on 25 Sep and a top-up of 5 at 21:00 WIB:")
    print("remaining on 25 Sep 21:00:", good.remaining(now))
    print("remaining on 26 Sep 00:01:", good.remaining(at("2026-09-26 00:01")))
```

Kode 8.1 (Lanjutan) *tiers.py: paket sebagai data, buku kuota yang menghitung pindaian per hari WIB, tambahan kuota untuk hari yang sama, dan bug tanggal UTC sebagai pembanding.*

```
    pro = QuotaBook("pro")
    for _ in range(100):
        pro.try_scan(now)
    print("\nPro after 100 scans today, remaining:", show(pro.remaining(now)))
    print("Free may add a 2nd monitor?", good.can_add_monitor(1))
    print("Pro may add a 2nd monitor?", pro.can_add_monitor(1))
    print("Free history starts:", good.history_from(now))
    print("Pro history starts:", pro.history_from(now) or "the beginning")


if __name__ == "__main__":
    main()
```

Beberapa baris layak dikomentari.

`WIB = ZoneInfo("Asia/Jakarta")` memuat aturan zona waktu Indonesia Barat dari basis data zona waktu sistem [8]. Setiap `datetime` dalam program bersifat *aware* (sadar zona), artinya membawa zonanya, sehingga tidak pernah ada keraguan dari jam mana suatu waktu dibaca.

`wib_day` mengubah suatu saat ke waktu Jakarta lalu mengambil tanggalnya. `utc_day` melakukan hal yang sama dengan UTC, dan *docstring*-nya dengan sengaja menyebutnya salah. `next_reset` mencari tanggal besok dalam WIB, membentuk tengah malam pada tanggal itu dalam WIB, lalu mengubahnya kembali ke UTC; hasilnya, pukul 17.00 UTC, adalah saat kuota diatur ulang sebagaimana akan dicatat log server.

Di dalam `QuotaBook`, `allowance` mengembalikan pindaian harian paket ditambah tambahan kuota untuk hari berjalan, atau `None` untuk tanpa batas. `used` menghitung pindaian yang harinya sama dengan hari ini, menurut fungsi `day_of` mana pun yang diberikan kepada buku itu. `remaining` adalah Persamaan (8.7) dalam bentuk kode. `try_scan` menolak pindaian jika tidak ada sisa dan mencatatnya jika ada. `add_topup` mengkreditkan pindaian tambahan ke hari berjalan dan tidak ke hari lain; itulah cara "berlaku hanya pada hari pembelian" diimplementasikan: `allowance` hari esok sama sekali tidak melihat tambahan kuota hari ini.

`main` menjalankan sebuah cerita kecil. Fungsi ini mencetak tabel paket, lalu melakukan 15 pindaian antara pukul 20.00 dan 23.16 WIB pada 24 September di dua buku kuota, satu menghitung hari WIB dan satu menghitung hari UTC. Ia mencoba pindaian keenam belas, mencetak saat pengaturan ulang berikutnya, lalu menanyai kedua buku berapa sisa pindaian pada tiga saat di tanggal 25 September: sesaat setelah tengah malam, satu menit sebelum pukul tujuh, dan tepat pukul tujuh. Cerita ditutup dengan tambahan kuota, pindaian tanpa batas di tingkat Pro, batas monitor, dan batas riwayat. Keluarannya:

```
Plan    scans/day  monitors  every  history  CSV  price/30d
Free           15         1    7 d     30 d   no  Rp0
Pro     unlimited        50    1 d     full  yes  Rp50,000
Pro costs Rp1,666.67 per day

15 scans on 24 Sep between 20:00 and 23:16 WIB
16th scan at 23:45 WIB allowed? False
next reset: 2026-09-25 00:00:00+07:00 = 2026-09-24 17:00:00+00:00
2026-09-25 00:05 WIB  remaining (WIB day) = 15   remaining (UTC day) =  0
2026-09-25 06:59 WIB  remaining (WIB day) = 15   remaining (UTC day) =  0
2026-09-25 07:00 WIB  remaining (WIB day) = 15   remaining (UTC day) = 15

After 15 scans on 25 Sep and a top-up of 5 at 21:00 WIB:
remaining on 25 Sep 21:00: 5
remaining on 26 Sep 00:01: 15

Pro after 100 scans today, remaining: unlimited
Free may add a 2nd monitor? False
Pro may add a 2nd monitor? True
Free history starts: 2026-08-26
Pro history starts: the beginning
```

Bacalah tiga baris yang diawali `2026-09-25`. Pada 00.05 dan 06.59 WIB buku yang benar menyebut 15 dan buku UTC menyebut 0; pada 07.00 keduanya menyebut 15. Itulah **Gambar 8.4**, direproduksi oleh kode yang berjalan. Baris-baris tentang tambahan kuota menunjukkan separuh lain aturan hari: lima pindaian tambahan pada pukul 21.00 tanggal 25 menaikkan sisa dari 0 menjadi 5, dan pada 00.01 tanggal 26 sisanya kembali menjadi 15 biasa, tambahan kuotanya hilang bersama harinya.

8.10.3 Program Java

Kode 8.2 adalah program yang sama dalam Java. Keluarannya persis sama dengan versi Python.

Kode 8.2 `Tiers.java`: paket, buku kuota, dan peragaan yang sama dalam Java, memakai `java.time`.

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.ZoneId;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.function.Function;

/** Free and Pro plans as data, and a quota calculator whose day
 *  starts at 00:00 WIB (Western Indonesia Time, UTC+7). */
public class Tiers {
    static final ZoneId WIB = ZoneId.of("Asia/Jakarta");
    static final int TERM_DAYS = 30;

    /** scansPerDay and historyDays are null when unlimited. */
    record Plan(String name, Integer scansPerDay, int monitors,
                int monitorEveryDays, Integer historyDays,
                boolean csvExport, int priceIdr) { }

    static final Map<String, Plan> PLANS = new LinkedHashMap<>();
    static {
        PLANS.put("free", new Plan("Free", 15, 1, 7, 30, false, 0));
        PLANS.put("pro", new Plan("Pro", null, 50, 1, null, true, 50_000));
    }

    /** The calendar day in Indonesia on which this instant falls. */
    static LocalDate wibDay(ZonedDateTime t) {
        return t.withZoneSameInstant(WIB).toLocalDate();
    }

    /** The WRONG day for a quota in Indonesia: the UTC calendar day. */
    static LocalDate utcDay(ZonedDateTime t) {
        return t.withZoneSameInstant(ZoneOffset.UTC).toLocalDate();
    }
```

Kode 8.2 (Lanjutan) *Tiers.java: paket, buku kuota, dan peragaan yang sama dalam Java, memakai java.time.*

```
    /** The next 00:00 WIB after this instant, as a UTC instant. */
    static ZonedDateTime nextReset(ZonedDateTime t) {
        LocalDate tomorrow = wibDay(t).plusDays(1);
        return ZonedDateTime.of(tomorrow, LocalTime.MIDNIGHT, WIB)
                .withZoneSameInstant(ZoneOffset.UTC);
    }

    /** Counts one account's scans per WIB day, plus same-day top-ups. */
    static class QuotaBook {
        final Plan plan;
        final Function<ZonedDateTime, LocalDate> dayOf;
        final List<ZonedDateTime> scans = new ArrayList<>();
        final Map<LocalDate, Integer> topups = new HashMap<>();

        QuotaBook(String tier) { this(tier, Tiers::wibDay); }

        QuotaBook(String tier, Function<ZonedDateTime, LocalDate> dayOf) {
            this.plan = PLANS.get(tier);
            this.dayOf = dayOf;         // wibDay, or utcDay to show the bug
        }

        Integer allowance(ZonedDateTime now) {
            if (plan.scansPerDay() == null) return null;
            return plan.scansPerDay() + topups.getOrDefault(dayOf.apply(now), 0);
        }

        int used(ZonedDateTime now) {
            LocalDate today = dayOf.apply(now);
            int n = 0;
            for (ZonedDateTime t : scans) if (dayOf.apply(t).equals(today)) n++;
            return n;
        }

        Integer remaining(ZonedDateTime now) {
            Integer limit = allowance(now);
            return limit == null ? null : Math.max(0, limit - used(now));
        }

        boolean tryScan(ZonedDateTime now) {
            Integer left = remaining(now);
            if (left != null && left == 0) return false;
            scans.add(now);
            return true;
        }
```

Kode 8.2 (Lanjutan) *Tiers.java: paket, buku kuota, dan peragaan yang sama dalam Java, memakai java.time.*

```
        /** Extra scans bought today are valid for today (WIB) only. */
        void addTopup(ZonedDateTime now, int extra) {
            topups.merge(dayOf.apply(now), extra, Integer::sum);
        }

        boolean canAddMonitor(int activeMonitors) {
            return activeMonitors < plan.monitors();
        }

        /** Oldest visible day of scan history; null means everything. */
        LocalDate historyFrom(ZonedDateTime now) {
            if (plan.historyDays() == null) return null;
            return dayOf.apply(now).minusDays(plan.historyDays());
        }
    }

    static double pricePerDay(Plan p) { return p.priceIdr() / (double) TERM_DAYS; }

    /** Parse "yyyy-MM-dd HH:mm" as a WIB wall-clock time. */
    static ZonedDateTime at(String text) {
        DateTimeFormatter f = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        return LocalDateTime.parse(text, f).atZone(WIB);
    }

    static String show(Integer n) { return n == null ? "unlimited" : n.toString(); }

    static String stamp(ZonedDateTime t) {
        return t.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ssxxx"));
    }

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        System.out.println(
                "Plan    scans/day  monitors  every  history  CSV  price/30d");
        for (Plan p : PLANS.values()) {
            String hist = p.historyDays() == null ? "full"
                                                  : p.historyDays() + " d";
            System.out.printf("%-7s %9s  %8d  %3d d  %7s  %3s  Rp%,d%n", p.name(),
                    show(p.scansPerDay()), p.monitors(), p.monitorEveryDays(),
                    hist, p.csvExport() ? "yes" : "no", p.priceIdr());
        }
        System.out.printf("Pro costs Rp%,.2f per day%n",
                pricePerDay(PLANS.get("pro")));
```

Kode 8.2 (Lanjutan) *Tiers.java: paket, buku kuota, dan peragaan yang sama dalam Java, memakai java.time.*

```
        QuotaBook good = new QuotaBook("free");                // WIB days
        QuotaBook bad = new QuotaBook("free", Tiers::utcDay);  // UTC days
        ZonedDateTime start = at("2026-09-24 20:00");
        for (int i = 0; i < 15; i++) {
            ZonedDateTime t = start.plusMinutes(14 * i);
            good.tryScan(t);
            bad.tryScan(t);
        }
        ZonedDateTime late = at("2026-09-24 23:45");
        System.out.println("\n15 scans on 24 Sep between 20:00 and 23:16 WIB");
        System.out.println("16th scan at 23:45 WIB allowed? "
                + (good.tryScan(late) ? "True" : "False"));
        ZonedDateTime r = nextReset(late);
        System.out.println("next reset: " + stamp(r.withZoneSameInstant(WIB))
                + " = " + stamp(r));

        for (String clock : new String[] {"2026-09-25 00:05", "2026-09-25 06:59",
                                          "2026-09-25 07:00"}) {
            ZonedDateTime now = at(clock);
            System.out.printf("%s WIB  remaining (WIB day) = %2d"
                    + "   remaining (UTC day) = %2d%n",
                    clock, good.remaining(now), bad.remaining(now));
        }

        ZonedDateTime now = at("2026-09-25 21:00");
        for (int i = 0; i < 15; i++) good.tryScan(now);
        good.addTopup(now, 5);
        System.out.println(
                "\nAfter 15 scans on 25 Sep and a top-up of 5 at 21:00 WIB:");
        System.out.println("remaining on 25 Sep 21:00: " + good.remaining(now));
        System.out.println("remaining on 26 Sep 00:01: "
                + good.remaining(at("2026-09-26 00:01")));

        QuotaBook pro = new QuotaBook("pro");
        for (int i = 0; i < 100; i++) pro.tryScan(now);
        System.out.println("\nPro after 100 scans today, remaining: "
                + show(pro.remaining(now)));
        System.out.println("Free may add a 2nd monitor? "
                + (good.canAddMonitor(1) ? "True" : "False"));
        System.out.println("Pro may add a 2nd monitor? "
                + (pro.canAddMonitor(1) ? "True" : "False"));
        System.out.println("Free history starts: " + good.historyFrom(now));
        LocalDate ph = pro.historyFrom(now);
        System.out.println("Pro history starts: "
                + (ph == null ? "the beginning" : ph.toString()));
    }
```

Kode 8.2 (Lanjutan) *Tiers.java: paket, buku kuota, dan peragaan yang sama dalam Java, memakai java.time.*

```
}
```

Versi Java memakai `record` untuk `Plan`, yang menghasilkan kelas *immutable* (tak dapat diubah) lengkap dengan konstruktor dan pengakses dalam satu baris. `Integer`, bukan `int`, dipakai untuk dua medan yang "mungkin tanpa batas", karena hanya tipe objek yang dapat bernilai `null`. Medan `dayOf` bertipe `Function<ZonedDateTime, LocalDate>`, sehingga kedua fungsi tanggal dapat diberikan sebagai referensi metode, `Tiers::wibDay` dan `Tiers::utcDay`. `ZonedDateTime.withZoneSameInstant` adalah padanan Java untuk `astimezone` di Python: saatnya tetap, jam tempat ia dibaca yang berubah [9].

`main` memanggil `Locale.setDefault(Locale.US)` sebelum memformat angka. Tanpanya, komputer dengan locale Indonesia akan mencetak `Rp50.000` dan `Rp1.666,67`, yang benar

menurut kaidah Indonesia tetapi tidak lagi cocok dengan keluaran Python. Nilai boolean dicetak sebagai `True` dan `False` dengan alasan yang sama. Untuk mengompilasi dan menjalankannya:

```
javac -d out Tiers.java
java -cp out Tiers
```

Keluarannya identik dengan keluaran **Kode 8.1**, baris demi baris, dan itu sendiri sebuah uji kecil: dua implementasi independen dari aturan yang sama saling sepakat.

8.11 Praktikum: Langganan sebagai State Machine

Program kedua memodelkan langganan Pro. Kita dapat menulisnya dengan beberapa penanda boolean (`is_pro`, `is_waiting`, `is_verifying`) dan sekumpulan pernyataan `if` yang kusut. Program yang ditulis dengan cara itu cenderung mencapai kombinasi yang seharusnya mustahil, misalnya "Pro sekaligus menunggu pembayaran", dan tidak ada yang menyadarinya sampai pelanggan mengeluh. *State machine* meniadakan kombinasi semacam itu sejak rancangannya.

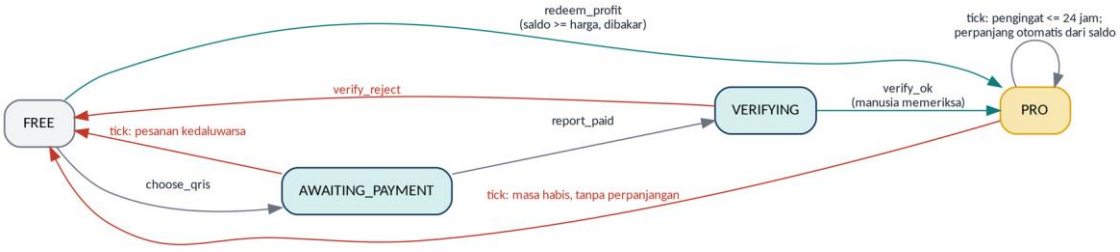
> **DEFINISI.** *Finite state machine* (mesin keadaan hingga) adalah model yang sistemnya selalu berada di tepat satu dari himpunan hingga *keadaan* (*state*), dan berpindah antarkeadaan hanya sebagai tanggapan atas *peristiwa* (*event*), melalui *transisi* yang didaftarkan sebelumnya. Peristiwa yang tidak memiliki transisi dari keadaan saat ini ditolak.

8.11.1 Keadaan, peristiwa, dan transisi

Langganan kita memiliki empat keadaan:

- `FREE`: tidak ada masa Pro yang aktif dan tidak ada pesanan yang sedang berjalan.
- `AWAITING_PAYMENT`: ada pesanan QRIS sebesar Rp50.000 yang belum dibayar.
- `VERIFYING`: pengguna melaporkan sudah membayar, dan pembayaran sedang diperiksa.
- `PRO`: masa 30 hari yang aktif sedang berjalan.

Langganan itu menanggapi tujuh peristiwa. `choose_qris` membuat pesanan. `report_paid` menyatakan bahwa pengguna sudah membayar. `verify_ok` dan `verify_reject` adalah hasil pemeriksaan. `redeem_profit` membayar dengan PROFIT. `tick` adalah berlalunya waktu, dikirim oleh sebuah jam: ia mengakhiri pesanan yang tak dibayar, mengirim pengingat, serta memperpanjang atau mengakhiri masa. **Gambar 8.5** menggambarkan mesin itu.



Gambar 8.5 State machine langganan dalam praktikum. Panah hijau toska menuju Pro, panah merah kembali ke Free, dan simpul pada Pro adalah pengingat serta perpanjangan otomatis; setiap keadaan juga menerima detak jam (`tick`).

Informasi yang sama dalam bentuk tabel sering lebih berguna bagi pemrogram, karena dapat diperiksa secara mekanis. **Tabel 8.3** mendaftar, untuk setiap keadaan, peristiwa yang diterimanya.

Tabel 8.3 Tabel transisi mesin langganan. Peristiwa apa pun yang tidak tercantum untuk keadaan saat ini ditolak dengan galat.

Keadaan	Peristiwa yang diterima	Ke mana arahnya
`FREE`	`choose_qris`, `redeem_profit`, `tick`	`AWAITING_PAYMENT`; `PRO` bila saldo mencukupi; tidak berubah
`AWAITING_PAYMENT`	`report_paid`, `tick`	`VERIFYING`; `FREE` ketika pesanan kedaluwarsa
`VERIFYING`	`verify_ok`, `verify_reject`, `tick`	`PRO`; `FREE`; tidak berubah
`PRO`	`tick`	pengingat, perpanjangan (tetap `PRO`), atau `FREE` pada akhir masa

Perhatikan apa yang dilarang tabel itu. `verify_ok` hanya diterima di `VERIFYING`. Begitu ia memindahkan langganan ke `PRO`, `verify_ok` kedua untuk pesanan yang sama mendapati mesin berada di `PRO`, tempat peristiwa itu tidak tercantum, sehingga ditolak. Klik ganda dari administrator yang lelah, atau notifikasi ganda dari sistem pembayaran, tidak dapat mengkreditkan dua masa. Perlindungan ini bukan `if` tambahan yang disisipkan setelah terjadi insiden; ia konsekuensi dari tabel.

8.11.2 Program Python

Kode 8.3 mengimplementasikan mesin itu. `ALLOWED` adalah **Tabel 8.3** dalam bentuk kamus, dan setiap metode peristiwa dimulai dengan memanggil `_require`, yang melemparkan `IllegalTransition` jika peristiwa itu tidak diizinkan pada keadaan saat ini.

Kode 8.3 `subscription.py`: state machine langganan, dengan pesanan QRIS, verifikasi, penukaran PROFIT, pengingat, dan perpanjangan otomatis.

```
"""A Pro subscription as a state machine: pay by QRIS (verified by a
person) or redeem PROFIT, 30-day terms, reminder and auto-renew."""
from datetime import datetime, timedelta
from enum import Enum
from zoneinfo import ZoneInfo

WIB = ZoneInfo("Asia/Jakarta")
TERM = timedelta(days=30)
ORDER_TTL = timedelta(hours=1)      # our lab's choice, not Profitina's
REMIND_BEFORE = timedelta(hours=24)
PRICE_IDR = 50_000
MICRO = 1_000_000                   # PROFIT has 6 decimals
PRICE_PROFIT = 50 * MICRO           # lab price: Rp50,000 at Rp1,000/PROFIT


class State(Enum):
    FREE = "FREE"
    AWAITING_PAYMENT = "AWAITING_PAYMENT"
    VERIFYING = "VERIFYING"
    PRO = "PRO"


ALLOWED = {
    State.FREE: {"choose_qris", "redeem_profit", "tick"},
    State.AWAITING_PAYMENT: {"report_paid", "tick"},
    State.VERIFYING: {"verify_ok", "verify_reject", "tick"},
    State.PRO: {"tick"},
}


class IllegalTransition(Exception):
    pass


def profit(micro: int) -> str:
    return f"{micro // MICRO}.{micro % MICRO:06d} PROFIT"
```

Kode 8.3 (Lanjutan) *subscription.py: state machine langganan, dengan pesanan QRIS, verifikasi, penukaran PROFIT, pengingat, dan perpanjangan otomatis.*

```
class Subscription:
    def __init__(self, balance_micro: int = 0, auto_renew: bool = True):
        self.state = State.FREE
        self.balance = balance_micro     # PROFIT, in micro-units
        self.burned = 0
        self.auto_renew = auto_renew
        self.order_deadline: datetime | None = None
        self.expires: datetime | None = None
        self.reminded = False
        self.log: list[str] = []

    def _require(self, event: str) -> None:
        if event not in ALLOWED[self.state]:
            raise IllegalTransition(f"{event} not allowed in {self.state.value}")

    def _note(self, now: datetime, text: str) -> None:
        stamp = now.astimezone(WIB).strftime("%d %b %H:%M")
        self.log.append(f"{stamp}  {self.state.value:<16} {text}")

    def _start_term(self, base: datetime) -> None:
        self.expires = base + TERM
        self.reminded = False
        self.state = State.PRO

    def choose_qris(self, now: datetime) -> None:
        self._require("choose_qris")
        self.state = State.AWAITING_PAYMENT
        self.order_deadline = now + ORDER_TTL
        self._note(now, f"QRIS order for Rp{PRICE_IDR:,} created")

    def report_paid(self, now: datetime) -> None:
        self._require("report_paid")
        self.state = State.VERIFYING
        self._note(now, "user reports payment; waiting for verification")

    def verify_ok(self, now: datetime) -> None:
        self._require("verify_ok")
        self._start_term(now)            # a new term starts now
        self._note(now, f"payment verified; Pro until {self._exp()}")

    def verify_reject(self, now: datetime, reason: str) -> None:
        self._require("verify_reject")
        self.state = State.FREE
        self._note(now, f"payment rejected: {reason}")
```

Kode 8.3 (Lanjutan) *subscription.py: state machine langganan, dengan pesanan QRIS, verifikasi, penukaran PROFIT, pengingat, dan perpanjangan otomatis.*

```
    def redeem_profit(self, now: datetime) -> None:
        self._require("redeem_profit")
        if self.balance < PRICE_PROFIT:
            self._note(now, f"redeem refused: balance {profit(self.balance)}")
            return
        self._spend()
        self._start_term(now)
        self._note(now, f"redeemed {profit(PRICE_PROFIT)}; Pro until "
                        f"{self._exp()}")

    def tick(self, now: datetime) -> None:
        """Called by a clock; the only event that makes time pass."""
        self._require("tick")
        if (self.state is State.AWAITING_PAYMENT
                and now >= self.order_deadline):
            self.state = State.FREE
            self._note(now, "QRIS order expired unpaid")
        elif self.state is State.PRO:
            if now >= self.expires:
                self._renew_or_lapse(now)
            elif now >= self.expires - REMIND_BEFORE and not self.reminded:
                self.reminded = True
                will = (self.auto_renew and self.balance >= PRICE_PROFIT)
                self._note(now, f"reminder: ends {self._exp()}, "
                                f"{'will renew' if will else 'will NOT renew'}")

    def _renew_or_lapse(self, now: datetime) -> None:
        if self.auto_renew and self.balance >= PRICE_PROFIT:
            self._spend()
            self._start_term(self.expires)  # extend: no hour lost or gained
            self._note(now, f"auto-renewed from balance; Pro until "
                            f"{self._exp()}")
        else:
            self.state = State.FREE
            self._note(now, "term ended; back to Free")

    def _spend(self) -> None:
        self.balance -= PRICE_PROFIT
        self.burned += PRICE_PROFIT      # spent PROFIT is burned, not kept

    def _exp(self) -> str:
        return self.expires.astimezone(WIB).strftime("%d %b %H:%M")
```

Kode 8.3 (Lanjutan) *subscription.py: state machine langganan, dengan pesanan QRIS, verifikasi, penukaran PROFIT, pengingat, dan perpanjangan otomatis.*

```
def at(text: str) -> datetime:
    return datetime.strptime(text, "%Y-%m-%d %H:%M").replace(tzinfo=WIB)


def main() -> None:
    a = Subscription()                              # pays with QRIS
    a.choose_qris(at("2026-09-24 10:00"))
    a.report_paid(at("2026-09-24 10:07"))
    a.verify_ok(at("2026-09-24 11:30"))
    for day in ("2026-10-23 09:00", "2026-10-23 12:00", "2026-10-24 11:30"):
        a.tick(at(day))
    print("Account A (QRIS, no PROFIT balance)          times in WIB")
    print("\n".join(a.log))

    b = Subscription(balance_micro=120 * MICRO)     # redeems PROFIT
    b.redeem_profit(at("2026-09-24 08:00"))
    for day in ("2026-10-23 09:00", "2026-10-24 08:00", "2026-11-22 09:00",
                "2026-11-23 08:00"):
        b.tick(at(day))
    print("\nAccount B (120 PROFIT, auto-renew on)")
    print("\n".join(b.log))
    print("balance", profit(b.balance), "| burned", profit(b.burned))

    c = Subscription()
    c.choose_qris(at("2026-09-24 20:00"))
    c.tick(at("2026-09-24 21:01"))
    try:
        c.report_paid(at("2026-09-24 21:05"))
    except IllegalTransition as e:
        c.log.append(f"{'':14}{'':17}refused: {e}")
    print("\nAccount C (order left unpaid)")
    print("\n".join(c.log))


if __name__ == "__main__":
    main()
```

Konstanta di bagian atas menyatakan asumsi praktikum di satu tempat. `TERM` 30 hari dan `REMIND_BEFORE` 24 jam, sesuai kartu paket. `ORDER_TTL`, lamanya pesanan QRIS tetap terbuka, adalah satu jam; ini pilihan praktikum kita, dan komentarnya mengatakan demikian. Jumlah PROFIT disimpan sebagai bilangan bulat dalam satuan mikro-PROFIT, karena PROFIT memiliki enam desimal (Bab 12): `MICRO` bernilai satu juta, dan `PRICE_PROFIT` adalah 50 PROFIT. Lima puluh adalah nilai Rp50.000 pada acuan awal Rp1.000 per PROFIT; di aplikasi sungguhan, harga dalam PROFIT ditampilkan sebelum pengguna mengonfirmasi. Menyimpan uang sebagai bilangan bulat, bukan bilangan titik-mengambang, adalah kebiasaan yang baik dibentuk sejak dini: 0.1 + 0.2 tidak tepat sama dengan 0.3 dalam titik-mengambang biner, tetapi 100,000 + 200,000 satuan mikro tepat sama dengan 300,000.

Metode-metode peristiwa itu pendek karena tabel mengerjakan bagian terberat. `choose_qris` mencatat tenggat. `report_paid` hanya mengubah keadaan. `verify_ok` memulai masa *sejak sekarang*, karena pembelian baru dimulai saat dikonfirmasi. `redeem_profit` memeriksa saldo, dan jika cukup, membelanjakan (dan membakar) harganya lalu memulai masa.

`tick` adalah satu-satunya metode yang membuat waktu berlalu. Di `AWAITING_PAYMENT`, ia mengakhiri pesanan yang tenggatnya sudah tiba. Di `PRO`, ia mengakhiri masa (dan memperpanjangnya bila mungkin) atau, dalam 24 jam sebelum akhir masa, mengirim satu pengingat yang dengan jujur menyatakan apakah masa itu *akan* diperpanjang. Kejujuran itu kecil

tetapi nyata: pengingat “Pro Anda berakhir besok” bagi orang yang perpanjangan otomatisnya akan gagal karena saldo kurang tidak seberguna pengingat yang berkata “will NOT renew” (tidak akan diperpanjang).

`_renew_or_lapse` memperpanjang dari saldo bila perpanjangan otomatis menyala dan saldo mencukupi, dan selain itu mengembalikan akun ke `FREE`. Perhatikan baris `self._start_term(self.expires)`: perpanjangan memulai masa baru *pada waktu berakhir yang lama*, yaitu Persamaan (8.4). Subbab 8.12 memperlihatkan apa yang terjadi ketika draf awal metode ini memulainya pada `now`.

`main` menjalankan tiga akun melalui mesin itu. Akun A membayar dengan QRIS dan tidak punya PROFIT, sehingga pengingatnya menyatakan tidak akan diperpanjang, dan ia kembali ke Free setelah 30 hari. Akun B mulai dengan 120 PROFIT dan perpanjangan otomatis menyala; ia menukar sekali, diperpanjang sekali, diingatkan pada kali kedua bahwa tidak akan diperpanjang, dan berakhir dengan sisa 20 PROFIT serta 100 terbakar. Akun C membuka pesanan, membiarkannya kedaluwarsa, lalu mencoba melaporkan pembayaran, yang ditolak mesin. Keluarannya:

```
Account A (QRIS, no PROFIT balance)          times in WIB
24 Sep 10:00  AWAITING_PAYMENT QRIS order for Rp50,000 created
24 Sep 10:07  VERIFYING        user reports payment; waiting for verification
24 Sep 11:30  PRO              payment verified; Pro until 24 Oct 11:30
23 Oct 12:00  PRO              reminder: ends 24 Oct 11:30, will NOT renew
24 Oct 11:30  FREE             term ended; back to Free

Account B (120 PROFIT, auto-renew on)
24 Sep 08:00  PRO              redeemed 50.000000 PROFIT; Pro until 24 Oct 08:00
23 Oct 09:00  PRO              reminder: ends 24 Oct 08:00, will renew
24 Oct 08:00  PRO              auto-renewed from balance; Pro until 23 Nov 08:00
22 Nov 09:00  PRO              reminder: ends 23 Nov 08:00, will NOT renew
23 Nov 08:00  FREE             term ended; back to Free
balance 20.000000 PROFIT | burned 100.000000 PROFIT

Account C (order left unpaid)
24 Sep 20:00  AWAITING_PAYMENT QRIS order for Rp50,000 created
24 Sep 21:01  FREE             QRIS order expired unpaid
                               refused: report_paid not allowed in FREE
```

Baris terakhir adalah mesin yang sedang menjalankan tugasnya. Pengguna melaporkan pembayaran untuk pesanan yang sudah tidak ada; program tidak mogok dan tidak pula diam-diam menerimanya, tetapi menolak dengan pesan yang menyebut tepat aturan mana yang dilanggar.

8.11.3 Program Java

Kode 8.4 adalah versi Java. Keadaannya berupa `enum`, tabel transisinya `EnumMap` dari keadaan ke himpunan nama peristiwa, dan `IllegalTransition` adalah eksepsi *unchecked*. Keluarannya identik dengan keluaran Python di atas.

Kode 8.4 `Subscription.java`: state machine yang sama dalam Java, dengan `EnumMap` sebagai tabel transisi.

```
import java.time.Duration;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.EnumMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Set;

/** A Pro subscription as a state machine: pay by QRIS (verified by a
 *  person) or redeem PROFIT, 30-day terms, reminder and auto-renew. */
public class Subscription {
    static final ZoneId WIB = ZoneId.of("Asia/Jakarta");
    static final Duration TERM = Duration.ofDays(30);
    static final Duration ORDER_TTL = Duration.ofHours(1); // our lab's choice
    static final Duration REMIND_BEFORE = Duration.ofHours(24);
    static final int PRICE_IDR = 50_000;
    static final long MICRO = 1_000_000L;                  // 6 decimals
    static final long PRICE_PROFIT = 50 * MICRO;  // Rp50,000 at Rp1,000/PROFIT

    enum State { FREE, AWAITING_PAYMENT, VERIFYING, PRO }

    static final Map<State, Set<String>> ALLOWED = new EnumMap<>(State.class);
    static {
        ALLOWED.put(State.FREE, Set.of("choose_qris", "redeem_profit", "tick"));
        ALLOWED.put(State.AWAITING_PAYMENT, Set.of("report_paid", "tick"));
        ALLOWED.put(State.VERIFYING, Set.of("verify_ok", "verify_reject", "tick"));
        ALLOWED.put(State.PRO, Set.of("tick"));
    }

    static class IllegalTransition extends RuntimeException {
        private static final long serialVersionUID = 1L;
        IllegalTransition(String msg) { super(msg); }
    }

    static String profit(long micro) {
        return String.format("%d.%06d PROFIT", micro / MICRO, micro % MICRO);
    }
```

Kode 8.4 (Lanjutan) *Subscription.java: state machine yang sama dalam Java, dengan EnumMap sebagai tabel transisi.*

```
    State state = State.FREE;
    long balance;                   // PROFIT, in micro-units
    long burned = 0;
    boolean autoRenew;
    ZonedDateTime orderDeadline;
    ZonedDateTime expires;
    boolean reminded = false;
    final List<String> log = new ArrayList<>();

    Subscription() { this(0, true); }

    Subscription(long balanceMicro, boolean autoRenew) {
        this.balance = balanceMicro;
        this.autoRenew = autoRenew;
    }

    private void require(String event) {
        if (!ALLOWED.get(state).contains(event))
            throw new IllegalTransition(event + " not allowed in " + state);
    }

    private void note(ZonedDateTime now, String text) {
        String stamp = now.withZoneSameInstant(WIB)
                .format(DateTimeFormatter.ofPattern("dd MMM HH:mm", Locale.US));
        log.add(String.format("%s  %-16s %s", stamp, state, text));
    }

    private void startTerm(ZonedDateTime base) {
        expires = base.plus(TERM);
        reminded = false;
        state = State.PRO;
    }

    void chooseQris(ZonedDateTime now) {
        require("choose_qris");
        state = State.AWAITING_PAYMENT;
        orderDeadline = now.plus(ORDER_TTL);
        note(now, String.format(Locale.US, "QRIS order for Rp%,d created",
                PRICE_IDR));
    }
```

Kode 8.4 (Lanjutan) *Subscription.java: state machine yang sama dalam Java, dengan EnumMap sebagai tabel transisi.*

```
    void reportPaid(ZonedDateTime now) {
        require("report_paid");
        state = State.VERIFYING;
        note(now, "user reports payment; waiting for verification");
    }

    void verifyOk(ZonedDateTime now) {
        require("verify_ok");
        startTerm(now);                          // a new term starts now
        note(now, "payment verified; Pro until " + exp());
    }

    void verifyReject(ZonedDateTime now, String reason) {
        require("verify_reject");
        state = State.FREE;
        note(now, "payment rejected: " + reason);
    }

    void redeemProfit(ZonedDateTime now) {
        require("redeem_profit");
        if (balance < PRICE_PROFIT) {
            note(now, "redeem refused: balance " + profit(balance));
            return;
        }
        spend();
        startTerm(now);
        note(now, "redeemed " + profit(PRICE_PROFIT) + "; Pro until " + exp());
    }

    /** Called by a clock; the only event that makes time pass. */
    void tick(ZonedDateTime now) {
        require("tick");
        if (state == State.AWAITING_PAYMENT && !now.isBefore(orderDeadline)) {
            state = State.FREE;
            note(now, "QRIS order expired unpaid");
        } else if (state == State.PRO) {
            if (!now.isBefore(expires)) {
                renewOrLapse(now);
            } else if (!now.isBefore(expires.minus(REMIND_BEFORE)) && !reminded) {
                reminded = true;
                boolean will = autoRenew && balance >= PRICE_PROFIT;
                note(now, "reminder: ends " + exp() + ", "
                        + (will ? "will renew" : "will NOT renew"));
            }
        }
    }
```

Kode 8.4 (Lanjutan) *Subscription.java: state machine yang sama dalam Java, dengan EnumMap sebagai tabel transisi.*

```
    private void renewOrLapse(ZonedDateTime now) {
        if (autoRenew && balance >= PRICE_PROFIT) {
            spend();
            startTerm(expires);                // extend: no hour lost or gained
            note(now, "auto-renewed from balance; Pro until " + exp());
        } else {
            state = State.FREE;
            note(now, "term ended; back to Free");
        }
    }

    private void spend() {
        balance -= PRICE_PROFIT;
        burned += PRICE_PROFIT;                // spent PROFIT is burned, not kept
    }

    private String exp() {
        return expires.withZoneSameInstant(WIB)
                .format(DateTimeFormatter.ofPattern("dd MMM HH:mm", Locale.US));
    }

    static ZonedDateTime at(String text) {
        DateTimeFormatter f = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        return LocalDateTime.parse(text, f).atZone(WIB);
    }

    public static void main(String[] args) {
        Subscription a = new Subscription();              // pays with QRIS
        a.chooseQris(at("2026-09-24 10:00"));
        a.reportPaid(at("2026-09-24 10:07"));
        a.verifyOk(at("2026-09-24 11:30"));
        for (String d : new String[] {"2026-10-23 09:00", "2026-10-23 12:00",
                                      "2026-10-24 11:30"}) a.tick(at(d));
        System.out.println(
                "Account A (QRIS, no PROFIT balance)          times in WIB");
        a.log.forEach(System.out::println);
```

Kode 8.4 (Lanjutan) *Subscription.java: state machine yang sama dalam Java, dengan EnumMap sebagai tabel transisi.*

```
        Subscription b = new Subscription(120 * MICRO, true); // redeems PROFIT
        b.redeemProfit(at("2026-09-24 08:00"));
        for (String d : new String[] {"2026-10-23 09:00", "2026-10-24 08:00",
                "2026-11-22 09:00", "2026-11-23 08:00"}) b.tick(at(d));
        System.out.println("\nAccount B (120 PROFIT, auto-renew on)");
        b.log.forEach(System.out::println);
        System.out.println("balance " + profit(b.balance) + " | burned "
                + profit(b.burned));

        Subscription c = new Subscription();
        c.chooseQris(at("2026-09-24 20:00"));
        c.tick(at("2026-09-24 21:01"));
        try {
            c.reportPaid(at("2026-09-24 21:05"));
        } catch (IllegalTransition e) {
            c.log.add(" ".repeat(31) + "refused: " + e.getMessage());
        }
        System.out.println("\nAccount C (order left unpaid)");
        c.log.forEach(System.out::println);
    }
}
```

Dua rincian Java layak dicatat. Perbandingan `ZonedDateTime` memakai `isBefore` dan `isAfter`; syarat "sekarang sama dengan atau sesudah tenggat" ditulis `!now.isBefore(orderDeadline)`, yang agak janggal dibaca tetapi tepat, termasuk untuk kasus keduanya sama. Selain itu, `Duration.ofDays(30)` berarti tepat $30 \times 86{,}400$ detik, dan itulah yang kita inginkan untuk sebuah masa: WIB tidak mengenal waktu musim panas, sehingga satu hari di sana selalu 24 jam, tetapi memakai `Duration` membuat program tetap benar bila dipindahkan ke zona yang mengenalnya.

Untuk menjalankannya:

```
javac -d out Subscription.java
java -cp out Subscription
```

8.12 Praktikum: Uji yang Mengunci Aturan

Aturan bisnis yang hanya hidup di dalam program hanya berjarak satu suntingan ceroboh dari perubahan. Seseorang "merapikan" `_start_term`, atau mengganti fungsi tanggal sambil memperbaiki hal lain, dan aturan yang dijanjikan kartu paket kepada setiap pelanggan diam-diam tidak lagi benar. Pertahanannya adalah serangkaian *uji*: program kecil yang menjalankan kode sungguhan pada masukan pilihan dan memeriksa bahwa jawabannya sesuai tuntutan aturan. Bab 9 memperlihatkan bagaimana suite uji Profitina yang jauh lebih besar menjadi gerbang bagi setiap rilis; di sini kita menulis uji untuk kedua praktikum kita.

8.12.1 Apa yang diuji

Cara yang baik untuk memilih uji adalah menelusuri aturan kalimat demi kalimat dan bertanya, untuk setiap kalimat, "masukan apa yang akan menunjukkan bahwa kalimat ini salah?". Untuk kalkulator kuota:

- Tabel paket menyatakan Free memiliki 15 pindaian dan 1 monitor, Pro pindaian tanpa batas, 50 monitor, dan Rp50.000. Sebuah uji membaca nilai-nilai itu dari `PLANS`.
- Hari dimulai pukul 00.00 WIB. Sebuah uji mengambil pukul 06.59 WIB tanggal 25 September dan memeriksa bahwa tanggal WIB-nya 25 dan tanggal UTC-nya 24.
- Pindaian Free keenam belas pada suatu hari ditolak, dan pindaian diizinkan lagi pada tengah malam WIB.
- Buku yang menghitung UTC baru terbuka lagi pada 07.00 WIB. Uji ini mengunci *bug*-nya, yang mungkin terasa aneh; uji ini menjamin bahwa peragaan pada **Kode 8.1** tetap memperagakan apa yang dikatakan teks.
- Tambahan kuota hanya berlaku pada hari pembeliannya.
- Pro tanpa batas, dan monitornya yang ke-51 ditolak.
- Riwayat Free dimulai 30 hari ke belakang; riwayat Pro lengkap.

Untuk mesin langganan daftarnya lebih panjang, karena state machine memiliki lebih banyak cara untuk keliru: jalur mulus QRIS; pesanan tak dibayar yang kedaluwarsa tepat pada tenggatnya dan tidak semenit pun lebih awal; pembayaran yang ditolak; `verify_ok` ganda; saldo yang kurang satu mikro-PROFIT dari harga; pembakaran saat penukaran; pengingat yang dikirim sekali dan hanya dalam 24 jam terakhir; perpanjangan yang memanjang dari waktu berakhir lama; tidak ada perpanjangan bila perpanjangan otomatis dimatikan; dan pemeriksaan struktural bahwa setiap keadaan menerima `tick`, sehingga jam tidak pernah dapat ditolak.

8.12.2 Uji Python

Pustaka standar Python menyertakan `unittest` [10]. Sebuah uji adalah metode yang namanya diawali `test_`, di dalam kelas yang memperluas `unittest.TestCase`; metode-metode `assert...` membandingkan nilai dan melaporkan dengan tepat apa yang berbeda. **Kode 8.5** dan **Kode 8.6** adalah kedua berkas uji itu.

Kode 8.5 `test_tiers.py`: delapan uji untuk tabel paket, hari WIB, batas harian, tambahan kuota, monitor, dan riwayat.

```
"""Tests for the plan table and the WIB quota day."""
import unittest
from datetime import timedelta, timezone

from tiers import PLANS, QuotaBook, at, next_reset, utc_day, wib_day


class TierTests(unittest.TestCase):
    def test_plan_table_matches_the_plan_cards(self):
        free, pro = PLANS["free"], PLANS["pro"]
        self.assertEqual((free.scans_per_day, free.monitors), (15, 1))
        self.assertEqual((pro.scans_per_day, pro.monitors), (None, 50))
        self.assertEqual(pro.price_idr, 50_000)

    def test_day_starts_at_midnight_wib_not_utc(self):
        t = at("2026-09-25 06:59")              # 24 Sep 23:59 UTC
        self.assertEqual(str(wib_day(t)), "2026-09-25")
        self.assertEqual(str(utc_day(t)), "2026-09-24")

    def test_next_reset_is_17_00_utc(self):
        r = next_reset(at("2026-09-24 23:45"))
        self.assertEqual((r.hour, r.minute, r.tzinfo), (17, 0, timezone.utc))

    def test_sixteenth_free_scan_is_refused(self):
        q = QuotaBook("free")
        t = at("2026-09-24 09:00")
        self.assertTrue(all(q.try_scan(t) for _ in range(15)))
        self.assertFalse(q.try_scan(t))
        self.assertTrue(q.try_scan(t + timedelta(hours=15)))   # 00:00 WIB

    def test_utc_counting_reopens_at_seven_wib(self):
        q = QuotaBook("free", day_of=utc_day)
        for _ in range(15):
            q.try_scan(at("2026-09-24 20:00"))
        self.assertEqual(q.remaining(at("2026-09-25 06:59")), 0)
        self.assertEqual(q.remaining(at("2026-09-25 07:00")), 15)

    def test_topup_is_valid_for_the_purchase_day_only(self):
        q = QuotaBook("free")
        q.add_topup(at("2026-09-24 23:00"), 5)
        self.assertEqual(q.remaining(at("2026-09-24 23:59")), 20)
        self.assertEqual(q.remaining(at("2026-09-25 00:00")), 15)
```

Kode 8.5 (Lanjutan) *test_tiers.py: delapan uji untuk tabel paket, hari WIB, batas harian, tambahan kuota, monitor, dan riwayat.*

```
    def test_pro_is_unlimited_and_monitors_are_capped(self):
        q = QuotaBook("pro")
        t = at("2026-09-24 12:00")
        self.assertTrue(all(q.try_scan(t) for _ in range(500)))
        self.assertTrue(q.can_add_monitor(49))
        self.assertFalse(q.can_add_monitor(50))

    def test_free_history_is_thirty_days(self):
        q = QuotaBook("free")
        self.assertEqual(str(q.history_from(at("2026-09-24 12:00"))),
                         "2026-08-25")
        self.assertIsNone(QuotaBook("pro").history_from(at("2026-09-24 12:00")))


if __name__ == "__main__":
    unittest.main()
```

Kode 8.6 `test_subscription.py`: sepuluh uji untuk state machine langganan.

```
"""Tests for the subscription state machine."""
import unittest
from datetime import timedelta

from subscription import (MICRO, IllegalTransition, State, Subscription,
                          TERM, at)

T0 = at("2026-09-24 10:00")


class SubscriptionTests(unittest.TestCase):
    def test_qris_happy_path(self):
        s = Subscription()
        s.choose_qris(T0)
        s.report_paid(T0 + timedelta(minutes=5))
        s.verify_ok(T0 + timedelta(hours=2))
        self.assertIs(s.state, State.PRO)
        self.assertEqual(s.expires, T0 + timedelta(hours=2) + TERM)

    def test_unpaid_order_expires(self):
        s = Subscription()
        s.choose_qris(T0)
        s.tick(T0 + timedelta(minutes=59))
        self.assertIs(s.state, State.AWAITING_PAYMENT)
        s.tick(T0 + timedelta(hours=1))
        self.assertIs(s.state, State.FREE)

    def test_rejected_payment_returns_to_free(self):
        s = Subscription()
        s.choose_qris(T0)
        s.report_paid(T0)
        s.verify_reject(T0, "amount does not match")
        self.assertIs(s.state, State.FREE)
        self.assertIsNone(s.expires)

    def test_verifying_twice_is_illegal(self):
        s = Subscription()
        s.choose_qris(T0)
        s.report_paid(T0)
        s.verify_ok(T0)
        with self.assertRaises(IllegalTransition):
            s.verify_ok(T0)                    # no double credit
```

Kode 8.6 (Lanjutan) *test_subscription.py: sepuluh uji untuk state machine langganan.*

```
    def test_redeem_needs_enough_balance(self):
        s = Subscription(balance_micro=49 * MICRO + 999_999)
        s.redeem_profit(T0)
        self.assertIs(s.state, State.FREE)
        self.assertEqual(s.burned, 0)

    def test_redeem_burns_the_price(self):
        s = Subscription(balance_micro=60 * MICRO)
        s.redeem_profit(T0)
        self.assertIs(s.state, State.PRO)
        self.assertEqual((s.balance, s.burned), (10 * MICRO, 50 * MICRO))

    def test_reminder_once_within_24_hours(self):
        s = Subscription(balance_micro=50 * MICRO)
        s.redeem_profit(T0)
        s.tick(s.expires - timedelta(hours=25))
        self.assertFalse(s.reminded)
        s.tick(s.expires - timedelta(hours=24))
        s.tick(s.expires - timedelta(hours=1))
        self.assertEqual(sum("reminder" in line for line in s.log), 1)

    def test_auto_renew_extends_from_old_expiry(self):
        s = Subscription(balance_micro=100 * MICRO)
        s.redeem_profit(T0)
        first = s.expires
        s.tick(first + timedelta(hours=3))     # the clock ran late
        self.assertIs(s.state, State.PRO)
        self.assertEqual(s.expires, first + TERM)

    def test_no_renew_when_switched_off(self):
        s = Subscription(balance_micro=100 * MICRO, auto_renew=False)
        s.redeem_profit(T0)
        s.tick(s.expires)
        self.assertIs(s.state, State.FREE)
        self.assertEqual(s.balance, 50 * MICRO)

    def test_every_state_accepts_tick(self):
        for state, events in __import__("subscription").ALLOWED.items():
            self.assertIn("tick", events, state)


if __name__ == "__main__":
    unittest.main()
```

Menjalankan `python3 -m unittest` di folder `python` menemukan setiap berkas yang namanya diawali `test`, menjalankan ke-18 uji, mencetak satu titik untuk setiap uji yang lolos, dan diakhiri `OK`:

```
..................
----------------------------------------------------------------------
Ran 18 tests in 0.005s

OK
```

8.12.3 Uji yang menangkap bug sungguhan

Draf pertama `subscription.py` memiliki `_start_term` seperti ini:

```
    def _start_term(self, now):
        # a renewal extends the old expiry, so no day is lost or gained
        base = self.expires if self.expires and self.expires > now else now
        self.expires = base + TERM
```

Komentarnya menyatakan niat yang benar, dan untuk kasus umum kodenya mengerjakan apa yang dikatakannya: perpanjangan yang diproses *sebelum* waktu berakhir lama memanjang dari waktu berakhir lama. Namun perpanjangan diproses oleh `tick`, dan `tick` berjalan *pada atau sesudah* waktu berakhir, tidak pernah sebelumnya. Dalam kasus itu `self.expires > now` bernilai salah, `base` menjadi `now`, dan setiap perpanjangan yang berjalan terlambat diam-diam menghadiahkan keterlambatan itu kepada pengguna. Detak jam yang terlambat tiga jam memberikan masa yang tiga jam terlalu panjang.

Uji `test_auto_renew_extends_from_old_expiry` membuat hal itu terjadi: ia menukar, membiarkan jam berjalan tiga jam melewati waktu berakhir, mengirim `tick`, lalu membandingkan waktu berakhir baru dengan waktu berakhir lama ditambah 30 hari. Jalankan pertama gagal, dengan `unittest` melaporkan waktu berakhir 23 November pukul 13.00 padahal yang diharapkan pukul 10.00. Perbaikannya adalah berhenti menebak di dalam `_start_term` dan membiarkan setiap pemanggil menyatakan di mana masa dimulai: `now` untuk pembelian baru, `self.expires` untuk perpanjangan. Itulah versi pada **Kode 8.3**.

Tiga pelajaran tersembunyi dalam kisah kecil ini. Pertama, komentarnya benar dan kodenya salah, hal yang lazim; uji memeriksa kode, bukan komentar. Kedua, bug-nya *menguntungkan pengguna*, sehingga tidak ada pelanggan yang akan melaporkannya; ia hanya akan tampak sebagai masa langganan yang jumlahnya tidak cocok. Ketiga, uji itu lahir dari membaca sebuah aturan, Persamaan (8.4), lalu bertanya masukan apa yang akan mematahkannya. Kebiasaan itulah yang menemukan bug, bukan kecerdikan.

8.12.4 Uji Java, dengan pelari mungil

Kerangka uji Java yang paling populer, JUnit, adalah pustaka terpisah. Agar praktikum tidak memerlukan unduhan, kita menulis pelari uji mungil sendiri, `MiniTest`, pada **Kode 8.7**. Ia menjalankan sebuah blok kode bernama, mencetak `ok` atau `FAIL`, dan diakhiri ringkasan; kode keluarnya 1 bila ada yang gagal, dan kode itulah yang akan dilihat gerbang rilis di Bab 9.

Kode 8.7 `MiniTest.java`: pelari uji dalam lima puluh baris, sehingga uji Java tidak memerlukan apa pun selain JDK.

```
import java.util.ArrayList;
import java.util.List;

/** A tiny test runner, so the tests need nothing but the JDK. */
public class MiniTest {
    interface Body { void run() throws Exception; }

    private final List<String> failures = new ArrayList<>();
    private int ran = 0;

    void test(String name, Body body) {
        ran++;
        try {
            body.run();
            System.out.println("ok    " + name);
        } catch (Throwable t) {
            failures.add(name);
            System.out.println("FAIL  " + name + ": " + t.getMessage());
        }
    }

    static void check(boolean condition, String what) {
        if (!condition) throw new AssertionError(what);
    }

    static void equal(Object expected, Object actual) {
        if (expected == null ? actual != null : !expected.equals(actual))
            throw new AssertionError("expected " + expected + " but got " + actual);
    }

    static void raises(Class<? extends Throwable> kind, Body body) {
        try {
            body.run();
        } catch (Throwable t) {
            if (kind.isInstance(t)) return;
            throw new AssertionError("expected " + kind.getSimpleName()
                    + " but got " + t);
        }
        throw new AssertionError("expected " + kind.getSimpleName());
    }
```

Kode 8.7 (Lanjutan) *MiniTest.java: pelari uji dalam lima puluh baris, sehingga uji Java tidak memerlukan apa pun selain JDK.*

```
    /** Prints the summary; the exit code is 1 when anything failed. */
    void finish() {
        System.out.printf("Ran %d tests: %d passed, %d failed%n",
                ran, ran - failures.size(), failures.size());
        if (!failures.isEmpty()) System.exit(1);
    }
}
```

Uji-ujinya sendiri, **Kode 8.8** dan **Kode 8.9**, sama dengan uji Python, satu lawan satu.

Kode 8.8 `TestTiers.java`: uji kuota dalam Java.

```
import java.time.ZoneOffset;
import java.time.ZonedDateTime;

/** Tests for the plan table and the WIB quota day. */
public class TestTiers {
    public static void main(String[] args) {
        MiniTest t = new MiniTest();
        t.test("plan table matches the plan cards", () -> {
            Tiers.Plan free = Tiers.PLANS.get("free");
            Tiers.Plan pro = Tiers.PLANS.get("pro");
            MiniTest.equal(15, free.scansPerDay());
            MiniTest.equal(1, free.monitors());
            MiniTest.equal(null, pro.scansPerDay());
            MiniTest.equal(50, pro.monitors());
            MiniTest.equal(50_000, pro.priceIdr());
        });
        t.test("day starts at midnight WIB, not UTC", () -> {
            ZonedDateTime x = Tiers.at("2026-09-25 06:59");   // 23:59 UTC
            MiniTest.equal("2026-09-25", Tiers.wibDay(x).toString());
            MiniTest.equal("2026-09-24", Tiers.utcDay(x).toString());
        });
        t.test("next reset is 17:00 UTC", () -> {
            ZonedDateTime r = Tiers.nextReset(Tiers.at("2026-09-24 23:45"));
            MiniTest.equal(17, r.getHour());
            MiniTest.equal(0, r.getMinute());
            MiniTest.equal(ZoneOffset.UTC, r.getZone());
        });
        t.test("sixteenth Free scan is refused", () -> {
            Tiers.QuotaBook q = new Tiers.QuotaBook("free");
            ZonedDateTime x = Tiers.at("2026-09-24 09:00");
            for (int i = 0; i < 15; i++) MiniTest.check(q.tryScan(x), "scan " + i);
            MiniTest.check(!q.tryScan(x), "16th refused");
            MiniTest.check(q.tryScan(x.plusHours(15)), "new day at 00:00 WIB");
        });
        t.test("UTC counting reopens at 07:00 WIB", () -> {
            Tiers.QuotaBook q = new Tiers.QuotaBook("free", Tiers::utcDay);
            for (int i = 0; i < 15; i++) q.tryScan(Tiers.at("2026-09-24 20:00"));
            MiniTest.equal(0, q.remaining(Tiers.at("2026-09-25 06:59")));
            MiniTest.equal(15, q.remaining(Tiers.at("2026-09-25 07:00")));
        });
        t.test("top-up is valid for the purchase day only", () -> {
            Tiers.QuotaBook q = new Tiers.QuotaBook("free");
            q.addTopup(Tiers.at("2026-09-24 23:00"), 5);
            MiniTest.equal(20, q.remaining(Tiers.at("2026-09-24 23:59")));
            MiniTest.equal(15, q.remaining(Tiers.at("2026-09-25 00:00")));
        });
```

Kode 8.8 (Lanjutan) *TestTiers.java: uji kuota dalam Java.*

```
        t.test("Pro is unlimited and monitors are capped", () -> {
            Tiers.QuotaBook q = new Tiers.QuotaBook("pro");
            ZonedDateTime x = Tiers.at("2026-09-24 12:00");
            for (int i = 0; i < 500; i++) MiniTest.check(q.tryScan(x), "scan");
            MiniTest.check(q.canAddMonitor(49), "49 -> 50 allowed");
            MiniTest.check(!q.canAddMonitor(50), "50 -> 51 refused");
        });
        t.test("Free history is thirty days", () -> {
            ZonedDateTime x = Tiers.at("2026-09-24 12:00");
            MiniTest.equal("2026-08-25",
                    new Tiers.QuotaBook("free").historyFrom(x).toString());
            MiniTest.equal(null, new Tiers.QuotaBook("pro").historyFrom(x));
        });
        t.finish();
    }
}
```

Kode 8.9 `TestSubscription.java`: uji langganan dalam Java.

```
import java.time.ZonedDateTime;
import java.util.Set;

/** Tests for the subscription state machine. */
public class TestSubscription {
    static final ZonedDateTime T0 = Subscription.at("2026-09-24 10:00");
    static final long M = Subscription.MICRO;

    public static void main(String[] args) {
        MiniTest t = new MiniTest();
        t.test("QRIS happy path", () -> {
            Subscription s = new Subscription();
            s.chooseQris(T0);
            s.reportPaid(T0.plusMinutes(5));
            s.verifyOk(T0.plusHours(2));
            MiniTest.equal(Subscription.State.PRO, s.state);
            MiniTest.equal(T0.plusHours(2).plus(Subscription.TERM), s.expires);
        });
        t.test("unpaid order expires", () -> {
            Subscription s = new Subscription();
            s.chooseQris(T0);
            s.tick(T0.plusMinutes(59));
            MiniTest.equal(Subscription.State.AWAITING_PAYMENT, s.state);
            s.tick(T0.plusHours(1));
            MiniTest.equal(Subscription.State.FREE, s.state);
        });
        t.test("rejected payment returns to Free", () -> {
            Subscription s = new Subscription();
            s.chooseQris(T0);
            s.reportPaid(T0);
            s.verifyReject(T0, "amount does not match");
            MiniTest.equal(Subscription.State.FREE, s.state);
            MiniTest.equal(null, s.expires);
        });
        t.test("verifying twice is illegal", () -> {
            Subscription s = new Subscription();
            s.chooseQris(T0);
            s.reportPaid(T0);
            s.verifyOk(T0);
            MiniTest.raises(Subscription.IllegalTransition.class,
                    () -> s.verifyOk(T0));             // no double credit
        });
        t.test("redeem needs enough balance", () -> {
            Subscription s = new Subscription(49 * M + 999_999, true);
            s.redeemProfit(T0);
            MiniTest.equal(Subscription.State.FREE, s.state);
```

Kode 8.9 (Lanjutan) *TestSubscription.java: uji langganan dalam Java.*

```
            MiniTest.equal(0L, s.burned);
        });
        t.test("redeem burns the price", () -> {
            Subscription s = new Subscription(60 * M, true);
            s.redeemProfit(T0);
            MiniTest.equal(Subscription.State.PRO, s.state);
            MiniTest.equal(10 * M, s.balance);
            MiniTest.equal(50 * M, s.burned);
        });
        t.test("reminder once within 24 hours", () -> {
            Subscription s = new Subscription(50 * M, true);
            s.redeemProfit(T0);
            s.tick(s.expires.minusHours(25));
            MiniTest.check(!s.reminded, "too early for a reminder");
            s.tick(s.expires.minusHours(24));
            s.tick(s.expires.minusHours(1));
            long n = s.log.stream().filter(l -> l.contains("reminder")).count();
            MiniTest.equal(1L, n);
        });
        t.test("auto-renew extends from old expiry", () -> {
            Subscription s = new Subscription(100 * M, true);
            s.redeemProfit(T0);
            ZonedDateTime first = s.expires;
            s.tick(first.plusHours(3));                // the clock ran late
            MiniTest.equal(Subscription.State.PRO, s.state);
            MiniTest.equal(first.plus(Subscription.TERM), s.expires);
        });
        t.test("no renew when switched off", () -> {
            Subscription s = new Subscription(100 * M, false);
            s.redeemProfit(T0);
            s.tick(s.expires);
            MiniTest.equal(Subscription.State.FREE, s.state);
            MiniTest.equal(50 * M, s.balance);
        });
        t.test("every state accepts tick", () -> {
            for (Set<String> events : Subscription.ALLOWED.values())
                MiniTest.check(events.contains("tick"), "tick missing");
        });
        t.finish();
    }
}
```

Kompilasi semuanya lalu jalankan kedua kelas uji:

```
javac -d out *.java
java -cp out TestTiers
java -cp out TestSubscription
```

```
ok    plan table matches the plan cards
ok    day starts at midnight WIB, not UTC
ok    next reset is 17:00 UTC
ok    sixteenth Free scan is refused
ok    UTC counting reopens at 07:00 WIB
ok    top-up is valid for the purchase day only
ok    Pro is unlimited and monitors are capped
ok    Free history is thirty days
Ran 8 tests: 8 passed, 0 failed
```

```
ok    QRIS happy path
ok    unpaid order expires
ok    rejected payment returns to Free
ok    verifying twice is illegal
ok    redeem needs enough balance
ok    redeem burns the price
ok    reminder once within 24 hours
ok    auto-renew extends from old expiry
ok    no renew when switched off
ok    every state accepts tick
Ran 10 tests: 10 passed, 0 failed
```

Delapan belas uji di setiap bahasa, semuanya lolos. Cobalah mengubah `REMIND_BEFORE` menjadi 12 jam di salah satu bahasa lalu jalankan uji lagi: uji pengingat gagal, dan ia menyebut aturan yang Anda langgar.

8.13 Profitina dalam Praktik

Semua isi bab ini terpetakan ke produk yang berjalan sebagai berikut, pada rilis September 2026.

- **Dua tingkat.** Free dan Pro, dengan batas-batas persis seperti **Tabel 8.1**. Kartu paket adalah satu-satunya tempat batas-batas itu dinyatakan; buku ini menyalinnya dari sana.
- **Harga dan pembayaran.** Pro seharga Rp50.000 untuk 30 hari, dibayar lewat QRIS dari bank atau dompet elektronik mana pun yang ikut serta, atau dengan menukarkan PROFIT, yang harganya ditampilkan aplikasi sebelum konfirmasi. PROFIT yang ditukarkan dibakar.
- **Perpanjangan.** Perpanjangan otomatis dari saldo PROFIT, dengan pengingat paling lambat 24 jam sebelum berakhir.
- **Agency.** Dihapus pada 6–7 Agustus 2026 demi kesederhanaan, setelah ketiga fiturnya terbukti berjalan.
- **Tanpa iklan.** Komitmen permanen, bukan sebuah fase.
- **Pengukuran.** Analisis Agustus 2026 menemukan nol pembelian PROFIT sepanjang riwayat produksi, dan tim menarik aturan “ukur minat sebelum membangun pembayaran” darinya. Pembelian PROFIT diverifikasi oleh manusia; otomatisasi pembayaran menunggu bukti permintaan.
- **Kuota harian.** Setiap batas harian, dari 15 pindaian Free sampai batas perolehan PROFIT, diatur ulang pada 00.00 WIB. Tambahan kuota pindaian yang dibeli dengan PROFIT hanya berlaku pada hari pembelian.
- **Gratis selamanya.** Hak atas data pribadi, fitur keselamatan, dan pemulihan akun tidak pernah menjadi bagian tingkat mana pun.

Praktikum di Subbab 8.10 sampai 8.12 tidak mereproduksi kode maupun konfigurasi Profitina. Praktikum itu mengajarkan gagasannya dengan angka-angka publik, pada contoh yang bisa saja milik produk kecil mana pun, termasuk Profitina Laundry, studi kasus pembelajaran seri buku ini. Sebuah binatu yang menjual paket bulanan “cuci kilat” akan menghadapi pertanyaan yang persis sama: apa saja yang termasuk, kapan hari dimulai, apa yang terjadi bila perpanjangan terlambat, dan bagaimana mencegah satu pembayaran dikreditkan dua kali.

8.14 Kesalahpahaman yang Sering Terjadi

> **KESALAHPAHAMAN 1: “Free adalah masa percobaan.”** Bukan. Free tidak memiliki tanggal berakhir dan mencakup alat pindai, konten, dan teknis secara penuh. Yang dibatasinya volume dan kesinambungan, bukan mutu.

KESALAHPAHAMAN 2: "Pengguna Pro mendapat skor lebih baik." Tidak. Kedua tingkat diukur oleh mesin yang sama. Pro membeli kapasitas: lebih banyak pindaian, monitor yang lebih banyak dan lebih sering, riwayat yang lebih panjang.

KESALAHPAHAMAN 3: "Agency dihapus karena tidak berjalan." Ketiga fitur Agency sudah terbukti berjalan. Tingkat itu dihapus demi kesederhanaan: lebih sedikit kombinasi untuk dibangun, diuji, didokumentasikan, dan dijelaskan, serta keputusan yang lebih mudah bagi pelanggan.

KESALAHPAHAMAN 4: "PROFIT adalah sumber pendapatan utama." Bukan. PROFIT adalah koin utilitas yang terutama diperoleh pengguna dengan berkontribusi. Langganan itulah model bisnisnya; dan temuan Agustus 2026 berupa nol pembelian PROFIT menunjukkan bahwa tidak seorang pun boleh menyusun anggaran berdasarkan penjualan PROFIT.

KESALAHPAHAMAN 5: "Kuota harian diatur ulang tengah malam, jadi tanggal komputer sudah cukup." Tanggal komputer biasanya UTC, yang tengah malamnya jatuh pada 07.00 WIB. Batas harian harus dihitung menurut kalender pengguna, yang bagi Profitina adalah `Asia/Jakarta`.

KESALAHPAHAMAN 6: "Verifikasi pembayaran secara manual itu tidak profesional." Pada volume rendah, justru itulah pilihan profesional: murah dibangun, mudah diaudit, dan menghasilkan bukti nyata tentang permintaan. Otomatisasi adalah investasi yang tepat setelah ada pembelian yang perlu diotomatiskan.

KESALAHPAHAMAN 7: "Tanpa iklan hanya berarti layar lebih bersih." Tanpa iklan menghapus konflik kepentingan antara pendapatan platform dan keberhasilan pengguna, melindungi sikap privasi di Bab 11, dan menjaga kredibilitas saran platform.

8.15 Ringkasan Bab

- Model bisnis menyatakan nilai apa yang diciptakan, bagi siapa, siapa yang membayar dan bagaimana, serta berapa biayanya. Pendapatan freemium mengikuti Persamaan (8.1), $R = N \cdot c \cdot P$; tingkat konversi lazimnya 2% sampai 5% [1].
- Indonesia pada 2026 memiliki 235,26 juta pengguna internet, penetrasi 81,72% [3], dan QRIS memiliki 65,77 juta pengguna serta 44,86 juta merchant [5], dengan 12,55 miliar transaksi pada semester pertama 2026 [6]. Harga dalam Rupiah yang dibayar lewat QRIS menjumpai orang di tempat mereka sudah biasa membayar.
- Profitina memiliki dua tingkat. Free: 15 pindaian sehari, satu monitor mingguan, riwayat 30 hari, alat lengkap. Pro: pindaian tanpa batas, 50 monitor harian dengan peringatan, riwayat lengkap dan CSV, Rp50.000 per 30 hari (sekitar Rp1.667 sehari menurut Persamaan (8.2)), dibayar lewat QRIS atau dengan menukarkan PROFIT, dengan perpanjangan otomatis dari saldo dan pengingat paling lambat 24 jam sebelum berakhir.
- Tingkat Agency dihapus pada 6–7 Agustus 2026 demi kesederhanaan, meskipun fiturnya berjalan; Persamaan (8.3) menunjukkan bagaimana setiap tingkat melipatgandakan apa yang harus dibangun dan diuji.
- Tanpa iklan, selamanya: pilihan struktural yang menyelaraskan insentif, melindungi privasi, dan menjaga kredibilitas saran.
- Temuan Agustus 2026 berupa nol pembelian PROFIT melahirkan aturan "ukur minat sebelum membangun pembayaran", sebuah tangga dari pengamatan ke penawaran manual ke penghitungan pembelian ke otomatisasi.

- Kuota harian diatur ulang pada 00.00 WIB. Hari WIB suatu saat adalah Persamaan (8.6); memakai tanggal UTC (Persamaan (8.5)) mengatur ulang pada 07.00 WIB. Sisa pindaian adalah Persamaan (8.7).
- Praktikum memodelkan paket sebagai data, kuota per hari WIB, dan langganan sebagai finite state machine yang tabel transisinya membuat pengkreditan ganda mustahil. Sebuah uji menangkap bug perpanjangan yang melanggar Persamaan (8.4).

8.16 Soal Latihan

Soal 8.1. Dengan Persamaan (8.1), hitung pendapatan 30 hari sebuah alat dengan $N = 5{,}000$ akun aktif, $c = 2\%$, dan $P =$ Rp50,000. Berapa tingkat konversi yang diperlukan untuk mencapai Rp10,000,000 dengan N dan P yang sama? Apakah tingkat itu masuk akal menurut rentang di Subbab 8.2.1?

Soal 8.2. (a) Hitung ulang Persamaan (8.2). (b) Seorang pengguna Pro menjalankan n pindaian sehari. Untuk nilai n berapa satu pindaian berharga kurang dari Rp100? (c) Mengapa "biaya per pindaian" menyesatkan untuk menilai Pro bagi pengguna yang kebutuhan utamanya pemantauan harian?

Soal 8.3. Dengan Persamaan (8.5) dan Persamaan (8.6), hitung d_{UTC} dan d_{WIB} untuk saat 25 September 2026 pukul 06.59 WIB ($t = 1{,}790{,}294{,}340$) dan 25 September 2026 pukul 07.00 WIB ($t = 1{,}790{,}294{,}400$). Jelaskan kecocokan jawaban Anda dengan keluaran **Kode 8.1**.

Soal 8.4. Tulis fungsi `seconds_until_reset(now)` (Java: `secondsUntilReset`) yang mengembalikan banyaknya detik hingga 00.00 WIB berikutnya, memakai `next_reset` dari **Kode 8.1**. Cetak nilainya untuk pukul 23.45 WIB tanggal 24 September, 00.00 WIB tanggal 25 September, dan 06.59 WIB tanggal 25 September. Apa seharusnya jawabannya tepat pada tengah malam, dan mengapa?

Soal 8.5. Tambahkan peristiwa `cancel_order` yang memungkinkan pengguna membatalkan pesanan QRIS sebelum membayar. (a) Di keadaan mana saja peristiwa itu boleh diterima, dan ke mana arahnya? (b) Tulis perubahan pada `ALLOWED` dan metode barunya. (c) Tulis satu uji yang menunjukkan peristiwa itu berjalan dan satu uji yang menunjukkan ia ditolak di `VERIFYING`. Jelaskan mengapa ia harus ditolak di sana.

Soal 8.6. Misalkan server mati selama 40 hari dan `tick` perpanjangan berjalan 40 hari setelah berakhirnya sebuah masa, dengan saldo yang cukup untuk memperpanjang. (a) Tanggal berakhir berapa yang diberikan Persamaan (8.4) untuk masa baru, relatif terhadap saat perpanjangan berjalan? (b) Usulkan aturan yang lebih adil untuk perpanjangan yang terlambat lebih dari satu masa penuh, dan implementasikan sebagai fungsi `fair_base(old_expiry, now)` (Java: `fairBase`) yang mengembalikan titik awal masa yang diperpanjang. Cetak kedua aturan untuk perpanjangan yang terlambat 0, 3, 29, dan 40 hari.

Soal 8.7. Dengan Persamaan (8.3) dan hitungan $T(T-1)$ perpindahan tingkat berarah, tabulasikan kombinasi dan perpindahan untuk $T = 1,2,3,4$ tingkat dengan $F = 12$ fitur yang bergantung pada tingkat. Seberapa banyak masing-masing berkurang karena Agency dihapus?

Soal 8.8. Seorang pengguna meminta "laporan PDF white-label dengan logo agensi saya". Mengikuti **Gambar 8.3**, uraikan secara konkret apa yang akan Anda lakukan pada setiap anak tangga sebelum membangun fitur itu, dan bukti apa yang membuat Anda naik ke anak tangga berikutnya.

Soal 8.9. Dalam paling banyak 150 kata, jelaskan kepada pemilik usaha kecil mengapa platform GEO yang menjanjikan "tanpa iklan, selamanya" lebih tepercaya daripada platform yang dibiayai iklan. Gunakan ketiga argumen di Subbab 8.6.

Soal 8.10. Tulis sebuah uji (dalam Python atau Java) yang menunjukkan bahwa tambahan kuota yang dibeli pukul 23.59 WIB tidak lagi tersedia pada 00.00 WIB keesokan harinya, dan satu uji lagi yang menunjukkan bahwa tambahan kuota yang sama pada `QuotaBook` yang menghitung hari UTC *masih* tersedia pada 00.00 WIB. Uji mana yang menggambarkan aturan, dan mana yang menggambarkan bug?

8.17 Penyelesaian Soal Latihan

Penyelesaian 8.1. $R = 5{,}000 \times 0.02 \times 50{,}000 = \text{Rp}5{,}000{,}000$. Untuk mencapai Rp10,000,000 diperlukan $c = R/(N \cdot P) = 10{,}000{,}000/(5{,}000 \times 50{,}000) = 0.04$, yaitu 4%. Angka itu berada dalam rentang 2% sampai 5% yang dicapai kebanyakan perusahaan freemium [1], tetapi di ujung atasnya; produk muda tidak boleh mengandalkannya tanpa bukti, dan itulah inti Subbab 8.8.

Penyelesaian 8.2. (a) $50{,}000/30 = 1{,}666.\overline{6}$, jadi sekitar Rp1.666,67 sehari, seperti yang dicetak baris-baris pertama keluaran **Kode 8.1**. (b) Biaya per pindaian $p/n < 100$ tepat ketika $n > p/100 = 16.67$, jadi mulai $n = 17$ pindaian sehari ke atas. (c) Pengguna pemantauan mungkin hanya menjalankan sedikit pindaian manual tetapi bergantung pada pemindaian ulang harian hingga lima puluh halaman, peringatan, dan riwayat lengkap. Tidak satu pun tercermin dalam "biaya per pindaian", sehingga ukuran itu meremehkan nilai Pro justru bagi pengguna yang kebutuhannya kesinambungan.

Penyelesaian 8.3. Untuk $t = 1{,}790{,}294{,}340$: $t/86{,}400 = 20{,}720.998\ldots$, sehingga $d_{\text{UTC}} = 20{,}720$ (24 September); $(t + 25{,}200)/86{,}400 = 20{,}721.29\ldots$, sehingga $d_{\text{WIB}} = 20{,}721$ (25 September). Untuk $t = 1{,}790{,}294{,}400$: $t/86{,}400 = 20{,}721.000$, sehingga $d_{\text{UTC}} = 20{,}721$, dan $d_{\text{WIB}} = \lfloor 20{,}721.29\ldots \rfloor = 20{,}721$. Pada 06.59 WIB kedua nomor hari berbeda, sehingga buku UTC masih menghitung 15 pindaian tanggal 24 dan mencetak sisa 0; pada 07.00 WIB keduanya sama, dan kedua buku mencetak 15. Itulah pasangan baris `06:59` dan `07:00` pada keluaran **Kode 8.1**.

Penyelesaian 8.4. **Kode 8.10** dan **Kode 8.11** mengurangkan saat sekarang dari `next_reset`.

Kode 8.10 `until_reset.py`: detik hingga 00.00 WIB berikutnya.

```python
"""Exercise 8.4: how long until the daily quota resets (00:00 WIB)?"""
from tiers import at, next_reset


def seconds_until_reset(now) -> int:
    return int((next_reset(now) - now).total_seconds())


for clock in ("2026-09-24 23:45", "2026-09-25 00:00", "2026-09-25 06:59"):
    s = seconds_until_reset(at(clock))
    print(f"{clock} WIB: {s:>5} s = {s // 3600:>2} h {s % 3600 // 60:>2} min")
```

Kode 8.11 `UntilReset.java`: perhitungan yang sama dengan `Duration.between`.

```java
import java.time.Duration;
import java.time.ZonedDateTime;

/** Exercise 8.4: how long until the daily quota resets (00:00 WIB)? */
public class UntilReset {
    static long secondsUntilReset(ZonedDateTime now) {
        return Duration.between(now, Tiers.nextReset(now)).getSeconds();
    }

    public static void main(String[] args) {
        for (String clock : new String[] {"2026-09-24 23:45", "2026-09-25 00:00",
                                          "2026-09-25 06:59"}) {
            long s = secondsUntilReset(Tiers.at(clock));
            System.out.printf("%s WIB: %5d s = %2d h %2d min%n",
                    clock, s, s / 3600, s % 3600 / 60);
        }
    }
}
```

Keduanya mencetak:

```
2026-09-24 23:45 WIB:   900 s =  0 h 15 min
2026-09-25 00:00 WIB: 86400 s = 24 h  0 min
2026-09-25 06:59 WIB: 61260 s = 17 h  1 min
```

Tepat pada 00.00 WIB jawabannya satu hari penuh, 86.400 detik, karena `next_reset` mengembalikan tengah malam *berikutnya* sesudah tanggal `now`, dan pada tengah malam hari WIB yang berjalan baru saja dimulai. Mengembalikan 0 akan menyiratkan bahwa pengaturan ulang masih tertunda, padahal baru saja terjadi.

Penyelesaian 8.5. (a) Hanya di `AWAITING_PAYMENT`, menuju `FREE`. Peristiwa itu harus ditolak di `VERIFYING`, karena pada saat itu pengguna sudah menyatakan telah membayar: jika uang benar-benar masuk, pembatalan akan menghilangkan jejak uang yang nyata. Pembayaran yang sedang diperiksa harus berakhir dengan `verify_ok` atau `verify_reject`, diputuskan oleh orang yang memeriksanya. (b) Di `ALLOWED`, ubah entri `AWAITING_PAYMENT` menjadi `{"report_paid", "cancel_order", "tick"}`, lalu tambahkan metode

```
    def cancel_order(self, now):
        self._require("cancel_order")
        self.state = State.FREE
        self._note(now, "QRIS order cancelled by the user")
```

(c) Dua uji:

```
    def test_cancel_before_paying(self):
        s = Subscription()
        s.choose_qris(T0)
        s.cancel_order(T0)
        self.assertIs(s.state, State.FREE)

    def test_cancel_refused_while_verifying(self):
        s = Subscription()
        s.choose_qris(T0)
        s.report_paid(T0)
        with self.assertRaises(IllegalTransition):
            s.cancel_order(T0)
```

Versi Java menambahkan `"cancel_order"` ke `Set.of(...)` untuk `AWAITING_PAYMENT` dan metode `cancelOrder` dengan tiga baris yang sama.

Penyelesaian 8.6. (a) Persamaan (8.4) memberikan waktu berakhir lama ditambah 30 hari, yaitu 10 hari *sebelum* saat perpanjangan berjalan: pengguna ditagih untuk masa yang sudah lewat, dan detak berikutnya langsung mengakhirinya. (b) Aturan yang adil: jika perpanjangan berjalan

kurang dari satu masa penuh terlambat, perpanjang dari waktu berakhir lama seperti sebelumnya (pengguna mendapat layanan atau hanya mengalami celah singkat akibat platform); jika terlambat satu masa penuh atau lebih, mulai masa baru dari `now`, karena layanan tidak diberikan selama periode itu dan pengguna tidak semestinya membayarnya. **Kode 8.12** dan **Kode 8.13** mengimplementasikan dan membandingkan kedua aturan.

Kode 8.12 `late_renewal.py`: aturan lama dan aturan yang adil untuk perpanjangan yang sangat terlambat.

```
"""Exercise 8.6: a renewal that runs more than a whole term late."""
from datetime import timedelta

from subscription import TERM, at


def fair_base(old_expiry, now):
    """Where should the renewed term start?"""
    # up to one term late: extend; longer: start from now
    return old_expiry if now - old_expiry < TERM else now


expiry = at("2026-10-01 08:00")
for days in (0, 3, 29, 40):
    now = expiry + timedelta(days=days)
    old_rule = (expiry + TERM).date()
    fair_rule = (fair_base(expiry, now) + TERM).date()
    print(f"renewal {days:>2} days late: old rule ends {old_rule}, "
          f"fair rule ends {fair_rule}")
```

Kode 8.13 `LateRenewal.java`: perbandingan yang sama dalam Java.

```
import java.time.Duration;
import java.time.ZonedDateTime;

/** Exercise 8.6: a renewal that runs more than a whole term late. */
public class LateRenewal {
    /** Where should the renewed term start? */
    static ZonedDateTime fairBase(ZonedDateTime oldExpiry, ZonedDateTime now) {
        Duration late = Duration.between(oldExpiry, now);
        // up to one term late: extend; longer: start from now
        return late.compareTo(Subscription.TERM) < 0 ? oldExpiry : now;
    }

    public static void main(String[] args) {
        ZonedDateTime expiry = Subscription.at("2026-10-01 08:00");
        for (int days : new int[] {0, 3, 29, 40}) {
            ZonedDateTime now = expiry.plusDays(days);
            System.out.printf("renewal %2d days late: old rule ends %s, "
                    + "fair rule ends %s%n", days,
                    expiry.plus(Subscription.TERM).toLocalDate(),
                    fairBase(expiry, now).plus(Subscription.TERM).toLocalDate());
        }
    }
}
```

```
renewal  0 days late: old rule ends 2026-10-31, fair rule ends 2026-10-31
renewal  3 days late: old rule ends 2026-10-31, fair rule ends 2026-10-31
renewal 29 days late: old rule ends 2026-10-31, fair rule ends 2026-10-31
renewal 40 days late: old rule ends 2026-10-31, fair rule ends 2026-12-10
```

Untuk memakai aturan itu di dalam mesin, ganti argumen `_start_term` di `_renew_or_lapse` dengan `fair_base(self.expires, now)`, lalu tambahkan uji dengan detak yang terlambat 40

hari. Aturan adil lain juga ada; yang penting adalah memilih satu dengan sengaja dan menguncinya dengan uji.

Penyelesaian 8.7.

Tabel 8.4 Kombinasi fitur-dan-tingkat serta perpindahan tingkat berarah untuk $F = 12$.

T	Kombinasi $C=T\times12$	Perpindahan $T(T-1)$
1	12	0
2	24	2
3	36	6
4	48	12

Beralih dari tiga tingkat ke dua, seperti pada Agustus 2026, menghapus 12 kombinasi (sepertiganya) dan 4 dari 6 perpindahan, sebagaimana ditunjukkan **Tabel 8.4**.

Penyelesaian 8.8. *Amati:* hitung berapa pengguna yang meminta laporan bermerek, dan berapa yang kini mengekspor CSV lalu menyusun laporan sendiri; tambahkan satu baris di bawah tombol ekspor, "Ingin laporan ini sebagai PDF bermerek?", dan hitung kliknya. *Tawarkan secara manual:* bagi pengguna yang meminta, buat PDF bermerek secara manual satu kali, dengan harga yang dinyatakan, lewat pesanan QRIS yang diverifikasi manusia. *Hitung pembelian:* lacak permintaan berbayar selama beberapa minggu, dan apakah pengguna yang sama kembali. *Otomatiskan:* hanya bila permintaan berbayar berulang muncul dan pekerjaan manual menjadi hambatan, bangun fiturnya, dan bahkan saat itu tambahkan ke kartu paket hanya setelah fitur berjalan. Bukti untuk naik: klik dan permintaan membenarkan penawaran manual; pembayaran membenarkan otomatisasi; pujian tidak membenarkan apa pun.

Penyelesaian 8.9. Contoh jawaban: "Alat yang dibiayai iklan memperoleh lebih banyak uang makin lama Anda bertahan dan makin banyak yang ditayangkan kepada Anda, jadi ia dibayar untuk membuat Anda sibuk, bukan untuk menyelesaikan masalah Anda. Langganan hanya menghasilkan uang ketika Anda cukup puas untuk memperpanjang, jadi ia dibayar untuk memperbaiki visibilitas Anda. Iklan juga memerlukan data tentang Anda untuk menargetkan; platform yang menolak iklan dapat menyimpan data merek Anda di servernya sendiri dan hampir tidak meminta apa pun. Dan platform yang menasihati Anda soal kepercayaan harus tampak tepercaya pula, hal yang sulit dilakukan sambil menjual ruang spanduk kepada siapa saja yang membayar." (93 kata.)

Penyelesaian 8.10. Dalam Python, dengan nama-nama dari **Kode 8.5**:

```
def test_topup_ends_at_wib_midnight(self):
    q = QuotaBook("free")
    q.add_topup(at("2026-09-24 23:59"), 5)
    self.assertEqual(q.remaining(at("2026-09-25 00:00")), 15)

def test_utc_book_keeps_topup_after_wib_midnight(self):
    q = QuotaBook("free", day_of=utc_day)
    q.add_topup(at("2026-09-24 23:59"), 5)
    self.assertEqual(q.remaining(at("2026-09-25 00:00")), 20)
```

Dalam buku UTC, pukul 23.59 WIB tanggal 24 dan 00.00 WIB tanggal 25 sama-sama jatuh pada 24 September UTC, sehingga tambahan kuota masih dihitung. Uji pertama menggambarkan aturan (tambahan kuota hanya berlaku pada hari pembelian WIB-nya); uji kedua menggambarkan bug. Menyimpan uji kedua hanya berguna selama program sengaja memperagakan bug itu, seperti yang dilakukan **Kode 8.1**.

Daftar Pustaka

[1] V. Kumar, "Making 'freemium' work," *Harvard Business Review*, vol. 92, no. 5, pp. 27–29, May 2014. [Online]. Available: https://hbr.org/2014/05/making-freemium-work

[2] Profitina, *Profitina Whitepaper, Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[3] ANTARA, "Survei APJII: penetrasi internet di Indonesia 2026 capai 81,7 persen," May 21, 2026. [Online]. Available: https://www.antaranews.com/berita/5576225/survei-apjii-penetrasi-internet-di-indonesia-2026-capai-817-persen

[4] CNN Indonesia, "Survei: pengguna internet Indonesia tumbuh 6 juta di 2026," May 21, 2026. [Online]. Available: https://www.cnnindonesia.com/teknologi/20260521122201-213-1360738/survei-pengguna-internet-indonesia-tumbuh-6-juta-di-2026

[5] ANTARA News, "QRIS reaches 65.77 mln users, accepted by 44.86 mln merchants: BI," Aug. 21, 2026. [Online]. Available: https://en.antaranews.com/news/428168/qris-reaches-6577-mln-users-accepted-by-4486-mln-merchants-bi

[6] Kontan, "QRIS tembus 12,55 miliar transaksi pada semester I 2026, pengguna capai 66 juta," Sep. 18, 2026. [Online]. Available: https://keuangan.kontan.co.id/news/qris-tembus-1255-miliar-transaksi-pada-semester-i-2026-pengguna-capai-66-juta

[7] E. Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. New York, NY, USA: Crown Business, 2011.

[8] Python Software Foundation, "zoneinfo: IANA time zone support," *Python 3.14 Documentation*. [Online]. Available: https://docs.python.org/3/library/zoneinfo.html

[9] Oracle, "Class ZonedDateTime," *Java SE 25 & JDK 25 API Specification*, 2025. [Online]. Available: https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/time/ZonedDateTime.html

[10] Python Software Foundation, "unittest: Unit testing framework," *Python 3.14 Documentation*. [Online]. Available: https://docs.python.org/3/library/unittest.html

Bab 9

Arsitektur, Keandalan, dan Gerbang Rilis

Setelah menyelesaikan bab ini, Anda akan mampu menggambar arsitektur Profitina pada tingkat kotak dan panah, serta menjelaskan apa fungsi setiap kotak dan mengapa ia ada: satu mesin Windows 11 yang menjalankan aplikasi FastAPI di bawah Uvicorn, dijaga tetap hidup sebagai layanan Windows oleh NSSM, menyimpan datanya di PostgreSQL, mengirim pembaruan langsung dengan Server-Sent Events, berpikir dengan model AI lokal dalam dua jalur, dan menjangkau internet hanya melalui Cloudflare Tunnel yang mengarah keluar, dengan situs pemasaran WordPress di sampingnya. Anda akan dapat menyatakan dalam angka apa arti "andal", dan mengapa sistem kecil di satu mesin dapat andal jika dikelola dengan cermat. Anda akan memahami bagaimana suite lebih dari seribu lima ratus uji otomatis berperan sebagai *release gate* (gerbang rilis), dan mengapa rilis Profitina baru menyatakan berhasil setelah layanan yang berjalan menjawab pemeriksaan kesehatannya dengan nomor versi yang baru. Anda akan mengikuti penyelidikan kinerja sungguhan, ketika perpindahan kanal di feed komunitas turun dari 868 milidetik menjadi 11, dan mempelajari kebiasaan rekayasa yang memungkinkan perbaikan itu dan menjaganya tetap beres. Terakhir, dalam praktikum G4, Anda akan membangun tiga program kecil dalam Python dan Java: penjadwal yang memindai ulang halaman terpantau dengan irama per tingkat layanan memakai *priority queue* (antrean prioritas), *endpoint* pemeriksaan kesehatan, dan skrip gerbang rilis yang menolak "men-*deploy*" bila uji gagal dan mengembalikan keadaan bila versi baru tidak menyala.

9.1 Titik Berangkat Kita

Bab 8 menjelaskan siapa yang membayar Profitina dan mengapa. Bab ini menjelaskan apa yang sebenarnya mereka bayar, di bawah layar-layarnya: mesinnya, perangkat lunak yang berjalan di atasnya, dan kebiasaan yang menjaganya tetap berjalan.

Mudah untuk mengira bahwa arsitektur adalah soal diagram besar dan teknologi terkenal. Bagi produk muda, arsitektur terutama menyangkut tiga pertanyaan yang lebih bersahaja. *Di mana setiap potong pekerjaan terjadi? Apa yang terjadi bila sesuatu rusak? Bagaimana kita mengubah sistem tanpa merusaknya?* Pertanyaan pertama menghasilkan arsitektur di Subbab 9.3. Pertanyaan kedua menghasilkan keandalan di Subbab 9.2 dan 9.4. Pertanyaan ketiga, yang paling sering dijawab tim paling akhir lalu disesali, menghasilkan gerbang rilis di Subbab 9.5 dan 9.6.

Kita akan menggambarkan Profitina pada tingkat kotak: apa setiap komponen dan mengapa dipilih. Kita tidak akan menjelaskan cara kerja penilaian di dalam kotak-kotak itu, dan kita tidak akan mencetak apa pun yang dapat membantu seseorang menyerang sistem, seperti nama mesin, jalur folder, atau rincian cara mesin dijangkau. Hal-hal itu tidak diperlukan untuk memetik pelajarannya, dan insinyur yang baik sejak awal belajar untuk tidak mencantumkannya dalam dokumen yang akan dibaca banyak orang.

Bab ini ditutup dengan praktikum G4, yang membangun ulang tiga gagasan terpenting bagi operasi sehari-hari, yaitu penjadwal, pemeriksaan kesehatan, dan gerbang rilis, sebagai program kecil yang dapat Anda jalankan di komputer sendiri.

9.2 Keandalan dengan Kata-Kata Sederhana

Sebelum melihat arsitekturnya, kita memerlukan cara untuk membicarakan seberapa baik ia bekerja. Semua orang setuju bahwa layanan harus "andal"; lebih sedikit orang yang dapat menyebut angka yang mereka maksud.

9.2.1 Ketersediaan

> **DEFINISI.** *Availability* (ketersediaan) *A* sebuah layanan selama suatu periode adalah bagian periode itu ketika layanan menjalankan tugasnya. Nilainya sering ditulis sebagai persentase, dan secara tidak resmi dihitung dalam "sembilan": 99,9% disebut "tiga sembilan".

Cara yang berguna untuk memikirkan ketersediaan adalah melalui dua rata-rata. *Mean time between failures* (MTBF, rerata waktu antarkegagalan) adalah berapa lama, rata-rata, layanan berjalan sebelum sesuatu menghentikannya. *Mean time to repair* (MTTR, rerata waktu pemulihan) adalah berapa lama, rata-rata, waktu yang dibutuhkan untuk menghidupkannya kembali. Dalam jangka panjang, layanan berganti-ganti antara masa hidup sekitar MTBF dan masa mati sekitar MTTR, sehingga

$$A = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \tag{9.1}$$

Persamaan (9.1) memuat pelajaran yang mengejutkan banyak pemula. Ada dua cara menaikkan ketersediaan, dan cara kedua sering jauh lebih murah daripada yang pertama. Kita dapat membuat kegagalan lebih jarang (menaikkan MTBF), yang biasanya berarti redundansi yang mahal. Atau kita dapat mempercepat pemulihan (menurunkan MTTR), yang sering berarti hal-hal sederhana: pengelola layanan yang menghidupkan ulang proses yang mogok dalam hitungan detik, cadangan yang dapat dipulihkan dalam hitungan menit, proses rilis yang dapat mundur tanpa perlu berpikir panjang. Layanan yang gagal sekali seminggu tetapi pulih dalam sepuluh detik memiliki ketersediaan di atas 99,998%.

9.2.2 Arti "sembilan" dalam menit

Waktu mati D yang diperbolehkan dalam periode sepanjang T pada ketersediaan A adalah

$$D = (1 - A) \cdot T \tag{9.2}$$

Tabel 9.1 menerapkan Persamaan (9.2) pada satu bulan 30 hari dan satu tahun 365 hari.

Tabel 9.1 Waktu mati yang diperbolehkan pada target ketersediaan yang lazim, menurut Persamaan (9.2).

Ketersediaan	Waktu mati per 30 hari	Waktu mati per tahun
99%	7,2 jam	87,6 jam
99,9%	43,2 menit	8,76 jam
99,99%	4,32 menit	52,6 menit
99,999%	25,9 detik	5,26 menit

> **CONTOH.** Sebuah binatu kecil memasang layanan pemesanan daring dengan target 99,9%. Menurut Persamaan (9.2), dalam 30 hari ia boleh mati $(1 - 0.999) \times 30 \times 24 \times 60 = 43.2$

> menit. Jika satu rilis yang ceroboh membuat layanan mati satu jam, jatah sebulan penuh habis dalam satu sore. Sebaliknya, jika setiap rilis dilakukan dengan gerbang yang cermat dan hanya mematikan layanan selama lima detik saat penyalaan ulang, delapan rilis sebulan hanya memakan empat puluh detik dari jatah itu. Angka ini memperlihatkan mengapa bab ini memberi perhatian begitu besar pada cara merilis: bagi sistem kecil, rilis adalah peristiwa yang paling sering mengancam ketersediaan.

Setiap tambahan satu "sembilan" kira-kira menuntut sepuluh kali upaya dibanding sembilan sebelumnya. Para *site reliability engineer* Google, yang menulis buku paling terkenal tentang pokok ini, berpendapat bahwa target yang tepat bukan "setinggi mungkin" melainkan "setinggi yang dibutuhkan pengguna", dan bahwa layanan yang lebih andal daripada kebutuhan penggunanya menyia-nyiakan upaya yang semestinya dapat dipakai untuk fitur [1]. Bagi platform GEO muda yang penggunanya memindai situs dan membaca daftar perbaikan, beberapa menit waktu mati selama rilis yang cermat pada jam sepi masih dapat diterima; kehilangan riwayat pindaian pengguna tidak dapat diterima. Keandalan adalah soal melindungi hal yang paling penting, bukan soal satu angka.

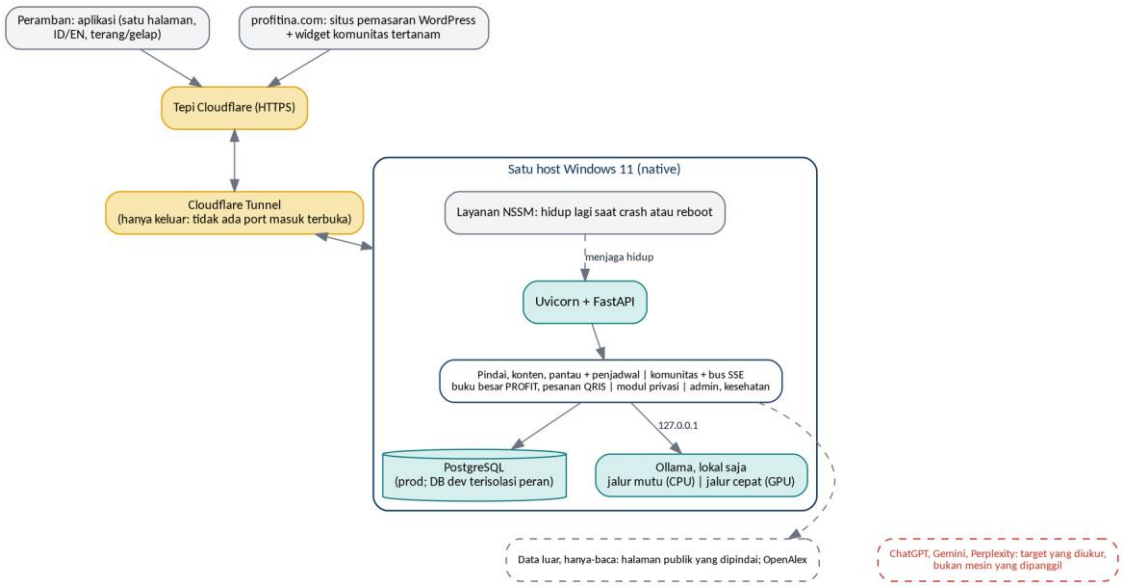
9.2.3 Keandalan mencakup kebenaran

Hal kedua, yang kurang kentara: layanan yang cepat menjawab tetapi jawabannya salah tidaklah andal. Aturan rekayasa Profitina memuat satu aturan yang terdengar hampir seperti pesan moral: **galat harus lantang**. Dalam tinjauan pertengahan September 2026, tim menemukan sebuah tempat di mana hitungan basis data yang gagal, dibungkus `try`/`except` yang mengembalikan nol, akan menampilkan kepada pengguna angka "0" yang meyakinkan alih-alih galat. Tidak akan ada yang pernah melaporkannya, karena nol tampak seperti jawaban biasa. Perbaikannya adalah membiarkan galat itu muncul. Jawaban salah yang diam lebih buruk daripada kegagalan yang terlihat, karena ia tidak dapat disadari, dan apa yang tidak dapat disadari tidak dapat diperbaiki.

Aturan yang sama muncul dalam produk itu sendiri: bila tidak ada model AI lokal yang tersedia, aplikasi mengatakannya alih-alih diam-diam menyajikan templat (Bab 6). Kejujuran adalah sifat keandalan.

9.3 Arsitektur pada Tingkat Kotak

Gambar 9.1 adalah seluruh sistem produksi Profitina dalam satu halaman, sebagaimana adanya pada rilis September 2026.



Gambar 9.1 Arsitektur Profitina pada tingkat kotak. Peramban dan situs pemasaran menjangkau satu host Windows 11 melalui tepi Cloudflare dan terowongan yang hanya mengarah keluar; di dalamnya, NSSM menjaga Uvicorn dan FastAPI tetap hidup, dengan PostgreSQL dan AI lokal di sampingnya. Mesin jawaban adalah target yang diukur, tidak pernah dipanggil.

Kita akan menelusurinya dari luar ke dalam, mengikuti sebuah permintaan dari ponsel seseorang sampai ke basis data dan kembali.

9.3.1 Dua pintu depan

Ada dua tempat orang bertemu Profitina.

Aplikasi adalah *single-page application* (aplikasi satu halaman) dalam JavaScript biasa, dikirim ke peramban sekali lalu memperbarui dirinya tanpa memuat ulang halaman. Aplikasi itu memakai *hash routing* (alamat seperti `#/komunitas/post/123`), kamus dwibahasa Indonesia dan Inggris, serta tema terang dan gelap. Aplikasi satu halaman cocok untuk dasbor yang dibiarkan terbuka, karena berpindah antarbagian hanya menuntut datanya, bukan halaman baru seutuhnya.

Situs pemasaran, profitina.com, adalah situs WordPress terpisah dengan tema dwibahasa khusus. Situs itu menjelaskan produk dalam halaman *answer-first*, menerbitkan data terstruktur dan berkas `llms.txt` untuk mesin jawaban AI, serta menyematkan widget komunitas dalam mode baca saja. Bab 10 membahasnya sebagai studi kasus GEO tersendiri. Dari sisi arsitektur, yang penting adalah pemisahannya: situs pemasaran dapat disunting, di-*cache*, bahkan rusak tanpa menyentuh aplikasi.

9.3.2 Tepi dan terowongan

Kedua pintu depan menjangkau internet melalui Cloudflare. *Edge*-nya (tepi), jaringan pusat data yang dekat dengan pengguna, mengakhiri HTTPS, menyerap lalu lintas yang merugikan, dan meneruskan permintaan yang sah.

Tautan dari tepi ke mesin aplikasi adalah **Cloudflare Tunnel**. Terowongan itu adalah program kecil yang berjalan di mesin, membuka koneksi *keluar* ke Cloudflare dan menjaganya tetap terbuka; permintaan untuk aplikasi mengalir kembali melalui koneksi itu. Dokumentasi Cloudflare menggambarkan modelnya dengan tepat: konektor “memulai koneksi keluar melalui firewall Anda”, sehingga Anda dapat “memblokir semua akses masuk ke *origin* kecuali melalui Cloudflare” [2].

> **WAWASAN KUNCI:** Terowongan yang hanya mengarah keluar mengubah masalah jaringan yang sulit menjadi mudah. Mesin tidak memerlukan alamat publik tetap dan tidak memerlukan port masuk yang terbuka. Tidak ada apa pun di internet publik yang dapat mengetuk pintunya secara langsung; setiap permintaan sudah lebih dulu melewati tepi.

Rancangan ini cocok untuk mesin yang berada di jaringan biasa, di belakang router yang pengaturannya mungkin tidak dikendalikan pemiliknya, dan menghapus satu kategori serangan sekaligus. Rancangan ini juga menggambarkan prinsip yang akan berulang dalam bab ini: utamakan susunan yang membuat hal berbahaya menjadi mustahil, bukan sekadar terlarang.

9.3.3 Satu host, dijaga hidup oleh NSSM

Aplikasi berjalan di satu mesin dengan **Windows 11 native**: tanpa mesin virtual, tanpa subsistem Linux, tanpa kontainer. Aplikasi Python dijalankan sebagai *service* (layanan) Windows, sehingga ia menyala ketika mesin dinyalakan dan berjalan tanpa ada yang masuk log. Pembungkus layanannya adalah **NSSM**, "Non-Sucking Service Manager", alat sumber terbuka kecil yang mengubah program apa pun menjadi layanan Windows dan menghidupkannya ulang secara otomatis bila berhenti [3].

Dalam kerangka Persamaan (9.1), NSSM adalah alat MTTR. Jika aplikasi mogok, ia kembali dalam hitungan detik; jika mesin di-*reboot* setelah pembaruan, layanan menyala sendiri. Keduanya tidak memerlukan manusia.

9.3.4 Aplikasinya: FastAPI di bawah Uvicorn

Aplikasinya ditulis dalam Python dengan **FastAPI**, kerangka kerja untuk membangun API web yang memakai petunjuk tipe Python untuk memvalidasi permintaan dan menghasilkan dokumentasi [4]. Aplikasi FastAPI dilayani oleh server ASGI; Profitina memakai **Uvicorn** [5]. ASGI (Asynchronous Server Gateway Interface) adalah standar Python yang memungkinkan satu proses memegang banyak koneksi terbuka sekaligus, yang penting bagi kotak berikutnya.

Di dalam aplikasi, pekerjaan dibagi ke dalam modul-modul menurut bidang: pemindaian, konten, dan Technical Assistant; pemantauan beserta penjadwalnya; komunitas dengan kanal, pesan langsung, dan moderasi; buku besar PROFIT, pesanan QRIS, dan pembelanjaan; modul privasi yang mengimplementasikan persetujuan, ekspor, dan penghapusan (Bab 11); administrasi; dan observabilitas. Pada saat tulisan ini dibuat, aplikasi membuka sekitar dua ratus rute HTTP. Menjelaskan apa yang dikerjakan setiap modul itu wajar dan berguna; cara kerja penilaian di dalamnya tetap tinggal di dalam.

9.3.5 Waktu nyata: Server-Sent Events

Ketika seseorang mengunggah di komunitas, setiap orang yang sedang melihat kanal itu harus langsung melihatnya. Profitina melakukannya dengan **Server-Sent Events** (SSE): peramban membuka satu permintaan HTTP berumur panjang ke *endpoint* peristiwa, dan server menulis pesan teks kecil melalui koneksi itu setiap kali sesuatu terjadi [6]. Objek `EventSource` bawaan peramban menyambung ulang secara otomatis bila koneksi terputus. Bab 4 membangun feed SSE dalam praktikum G2; di sini cukup dicatat bahwa SSE bersifat satu arah (dari server ke peramban), memakai HTTP biasa, dan melewati tepi serta terowongan tanpa konfigurasi khusus.

Mengapa bukan cara yang lebih tua, yaitu peramban bertanya ke server setiap beberapa detik apakah ada yang baru? Cara itu, yang disebut *polling*, sederhana tetapi boros: sebagian besar pertanyaan dijawab "belum ada apa-apa", dan setiap pertanyaan tetap memakan koneksi, kueri, dan baterai ponsel. Dengan seratus pembaca yang masing-masing bertanya setiap tiga detik,

server menjawab dua ribu pertanyaan per menit hanya untuk mengatakan bahwa tidak ada yang terjadi. Dengan SSE, server diam sampai ada sesuatu yang layak dikatakan, lalu mengatakannya sekali kepada semua yang mendengarkan. Mengapa pula bukan WebSocket, yang dua arah? Karena feed hanya membutuhkan satu arah: unggahan baru dikirim lewat permintaan HTTP biasa, sedangkan pemberitahuan mengalir dari server ke peramban. Memilih alat yang sekecil kebutuhan adalah kebiasaan yang akan kita temui berulang kali dalam bab ini.

9.3.6 Basis data: PostgreSQL, dengan pemisahan yang ditegakkan

Data produksi disimpan di **PostgreSQL**. Ada basis data pengembangan yang terpisah, dan pemisahan keduanya ditegakkan oleh izin peran (*role*) PostgreSQL sendiri, bukan oleh kesepakatan yang dipercayakan kepada pengembang [7]. Sesi pengembangan tersambung dengan peran yang sama sekali tidak memiliki hak atas tabel produksi; kesalahan ditolak oleh basis data, bukan sekadar dicegah oleh sebuah dokumen.

Suite uji juga berjalan pada SQLite, yang lebih cepat disiapkan. Dua dialek SQL mengundang perbedaan halus, dan Subbab 9.5.3 menceritakan bagaimana salah satunya tertangkap.

9.3.7 Jalur AI: lokal saja

Semua pekerjaan AI berjalan di mesin yang sama melalui **Ollama**, server model lokal, dengan model dari keluarga Qwen dalam dua jalur (Bab 6): *jalur mutu* di prosesor untuk konten panjang, dan *jalur cepat* di kartu grafis untuk klasifikasi pendek seperti keamanan tautan dan deteksi kategori. Tidak ada data pengguna yang meninggalkan server untuk pemrosesan AI, dan mesin jawaban yang diukur Profitina (ChatGPT, Gemini, Perplexity) tidak pernah dipanggil: mereka target, bukan komponen.

Satu pelajaran kecil dari kotak ini memiliki angka. Di Windows, nama `localhost` dapat lebih dulu diterjemahkan ke alamat IPv6 `::1`, sedangkan server yang hanya mendengarkan IPv4 menunggu di `127.0.0.1`. Klien mencoba IPv6, menunggu, lalu mundur. Dalam pengukuran tim, memanggil server model di `127.0.0.1` alih-alih `localhost` memangkas waktu hingga token pertama dari **2,25 detik menjadi 0,17 detik**. Pemeriksaan kesehatan dalam praktikum di Subbab 9.10 mengikuti aturan yang sama, dan menyatakannya dalam komentar.

9.3.8 Apa yang sengaja tidak ada dalam gambar

Sama pentingnya dengan kotak-kotak pada **Gambar 9.1** adalah kotak yang tidak ada. Tidak ada penyedia awan raksasa, tidak ada Kubernetes, tidak ada *microservices*, tidak ada penskalaan otomatis. **Tabel 9.2** merangkum pilihan-pilihan itu beserta alasannya.

Tabel 9.2 Komponen-komponen **Gambar 9.1**, pilihan untuk masing-masing, dan alasan utamanya.

Kotak	Pilihan	Alasan utama
Antarmuka depan	Aplikasi satu halaman, JavaScript biasa	Dasbor yang dibiarkan terbuka; tanpa rantai *build* yang perlu dirawat
Pemasaran	WordPress, tema dwibahasa	Penyunting dapat menerbitkan tanpa menyentuh aplikasi
Tepi	Cloudflare	HTTPS, *cache*, perlindungan di depan satu mesin
Jalan masuk	Cloudflare Tunnel, hanya keluar	Tidak memerlukan alamat publik atau port masuk
Host	Satu mesin Windows 11, native	Sesuai tahapan; hal paling sederhana yang berhasil
Layanan	NSSM	Menyala ulang saat mogok dan saat *boot*, tanpa manusia
Aplikasi	FastAPI di Uvicorn (ASGI)	Validasi bertipe, banyak koneksi terbuka
Waktu nyata	Server-Sent Events	Dorongan satu arah lewat HTTP biasa
Data	PostgreSQL, dev diisolasi dengan peran	Matang, ketat; kesalahan ditolak basis data
AI	Ollama, dua jalur lokal	Privasi, kemandirian, tanpa tagihan per permintaan

9.4 Mengapa Satu Mesin Cukup, untuk Saat Ini

Satu mesin adalah *single point of failure* (titik kegagalan tunggal): jika ia mati, layanan mati. Para pengkritik arsitektur kecil biasanya berhenti di situ. Ada baiknya kita melanjutkan.

Pertama, alternatifnya memiliki modus kegagalannya sendiri. Sistem terdistribusi memiliki lebih banyak bagian bergerak, lebih banyak tautan jaringan, lebih banyak konfigurasi, dan lebih banyak cara bagi dua bagian untuk berselisih. Bagi tim beranggotakan beberapa orang, penyebab gangguan yang paling mungkin bukan kerusakan perangkat keras; melainkan kesalahan saat melakukan perubahan. Sistem yang lebih sederhana membuat lebih sedikit kesalahan semacam itu mungkin terjadi, dan membuat sisanya lebih mudah terlihat.

Kedua, biaya skala datang lebih dulu daripada manfaatnya. Menjalankan layanan yang sama di banyak mesin menuntut sesi bersama, basis data yang dapat dijangkau lewat jaringan, rilis yang terkoordinasi, dan pemantauan yang memahami semuanya. Masing-masing berarti berminggu-minggu kerja, dan tidak satu pun memperbaiki satu pindaian pun bagi satu pengguna pun selama lalu lintas masih sedang-sedang saja.

Ketiga, sisa bab ini menunjukkan bahwa sebagian besar keandalan berasal dari *proses*, bukan dari banyaknya mesin: pengelola layanan yang cepat menghidupkan ulang, cadangan yang diambil sebelum setiap perubahan, suite uji yang harus lolos, dan rilis yang membuktikan versi baru sudah berjalan sebelum mengaku berhasil. Kebiasaan itu terbawa tanpa perubahan ketika sistem kelak bertumbuh.

Satu catatan tentang cadangan, karena kata itu muncul berkali-kali dalam bab ini. Cadangan yang tidak pernah dicoba dipulihkan bukanlah cadangan; ia hanyalah harapan. Kebiasaan yang sehat adalah sesekali memulihkan cadangan ke tempat terpisah dan memeriksa bahwa isinya benar-benar dapat dipakai: aplikasi menyala, data terbaca, jumlah barisnya masuk akal. Latihan semacam itu juga mengukur MTTR yang sesungguhnya, karena waktu yang dibutuhkan untuk memulihkan baru diketahui setelah pernah dilakukan. Dalam kerangka Persamaan (9.1), latihan pemulihan adalah investasi murah untuk menurunkan penyebut.

Peta jalan publik Profitina menyisihkan skala regional untuk fase berikutnya, setelah produk dalam negeri membuktikan dirinya [8]. Pada saat itu arsitekturnya akan berubah, dan memang

sepatutnya. Membangun untuk skala itu sekarang berarti membayar hari ini untuk masalah yang belum dimiliki produk.

> **PERINGATAN:** "Satu mesin cukup" adalah klaim tentang tahapan *ini*, didukung oleh pengukuran. Itu bukan hukum umum. Versi jujurnya: ketahuilah apa yang akan membuatnya tidak benar lagi (beban berkelanjutan yang tidak sanggup dilayani mesin, atau waktu pemulihan yang tidak dapat diterima pengguna) dan awasi tanda-tandanya.

9.5 Suite Uji sebagai Gerbang Rilis

Setiap perubahan pada sistem yang berjalan adalah risiko kecil. Tujuan uji otomatis adalah membuat sebagian besar risiko itu terlihat *sebelum* perubahan sampai ke pengguna.

> **DEFINISI.** *Release gate* (gerbang rilis) adalah pemeriksaan yang harus dilewati sebuah rilis sebelum boleh berlanjut. Gerbang hanya layak disebut gerbang bila kegagalannya benar-benar menghentikan rilis: pemeriksaan yang kegagalannya dilaporkan tetapi diabaikan hanyalah peringatan, bukan gerbang.

9.5.1 Ukuran suite

Pada saat tulisan ini dibuat (24 September 2026), suite Profitina tampak seperti **Tabel 9.3**.

Tabel 9.3 Suite otomatis yang menjadi gerbang setiap rilis, pada saat tulisan ini dibuat (24 September 2026).

Bagian suite	Ukuran pada rilis September 2026
Suite Python penuh pada SQLite	1.577 lolos, 1 dilewati
Subset prioritas pada PostgreSQL 16	440 lolos (49 berkas uji)
Berkas uji Python	153
Berkas penjaga UI JavaScript	6, dengan *harness*-nya sendiri
Pemeriksaan ujung ke ujung	Playwright yang mengendalikan peramban Chromium sungguhan

Suite penuh berjalan dengan `pytest`, dikonfigurasi sehingga setiap *peringatan* menjadi galat (`filterwarnings = error`). Pengaturan ini kecil tetapi tegas: peringatan *deprecation* hari ini adalah kegagalan esok hari setelah pustaka diperbarui, dan memperlakukannya sebagai galat berarti ia diperbaiki selagi masih mudah. Penjaga JavaScript memeriksa aturan antarmuka yang mudah rusak tanpa disadari, misalnya setiap teks yang tampil harus memiliki terjemahan Indonesia dan Inggris.

Angka seperti 1.577 menua dalam hitungan hari, dan bukan itu intinya. Intinya adalah aturannya: **seluruh suite harus hijau sebelum rilis berlanjut**.

9.5.2 Mengapa gerbang, dan mengapa sekarang

Ada baiknya berhenti sejenak untuk melihat mengapa hal ini lebih penting pada 2026 daripada beberapa tahun lalu. Laporan DORA 2025, berdasarkan survei terhadap hampir 5.000 profesional teknologi, menemukan bahwa 90% responden memakai AI dalam pekerjaan dan bahwa adopsi AI kini berhubungan positif dengan *throughput* (laju) pengiriman perangkat lunak, tetapi tetap berhubungan *negatif* dengan *stabilitas* pengiriman [9]. Ringkasan laporan itu mudah diingat: "AI tidak memperbaiki sebuah tim; ia memperbesar apa yang sudah ada." Kode ditulis lebih cepat daripada sebelumnya. Tim yang ujinya tipis hanya akan mengirim kesalahannya lebih cepat. Tim dengan gerbang yang kuat mengubah kecepatan menjadi nilai.

9.5.3 Dua pelajaran dari suite

Uji hanya sebaik apa yang dijalankannya. Dua kisah dari riwayat produk menunjukkan bagaimana sebuah suite menjadi lebih baik.

Jebakan dialek. SQLite dan PostgreSQL sama-sama berbahasa SQL, tetapi tidak identik. SQLite menerima `MAX(a, b)` sebagai fungsi dua nilai yang mengembalikan nilai lebih besar; PostgreSQL tidak, dan mengharapkan `GREATEST(a, b)`. Secara terpisah, sebuah uji yang ditulis tangan pernah menyisipkan baris langsung ke tabel tanpa satu kolom yang dideklarasikan PostgreSQL sebagai `NOT NULL`; SQLite menerimanya, PostgreSQL menolak. Dua aturan lahir dari insiden-insiden ini. Pertama, uji *penjaga dialek* memindai kode untuk konstruksi yang berperilaku berbeda di kedua basis data, dan subset prioritas uji dijalankan pada PostgreSQL sungguhan. Kedua: **uji menyiapkan keadaannya melalui fungsi produk sendiri bila fungsi itu ada**, bukan dengan SQL tulisan tangan. Uji yang memakai fungsi "buat unggahan" yang sungguhan akan gagal bila fungsi itu rusak, dan tetap berjalan bila tabel mendapat kolom baru; uji dengan `INSERT` sendiri hanya memeriksa dirinya sendiri.

Nol yang diam. Aturan galat lantang di Subbab 9.2.3 juga ditegakkan dengan uji: setelah tim tahu bahwa galat yang ditelan dapat tampak seperti nol, ditambahkan uji yang sengaja memicu kegagalan itu dan memeriksa bahwa ia terlihat.

> **WAWASAN KUNCI:** Suite uji yang baik tumbuh dari insiden. Setiap bug yang sampai ke manusia adalah uji yang belum ada; menulis uji itu adalah cara mencegah bug yang sama kembali lagi.

9.6 Rilis Bertahap yang Diverifikasi Kesehatannya

Uji menunjukkan bahwa kode baru berjalan *di mesin uji*. Uji tidak menunjukkan bahwa kode itu *sedang berjalan* di produksi. Rilis harus menyeberangi celah terakhir itu, dan di celah terakhir itulah terjadi salah satu insiden paling mendidik dalam riwayat Profitina.

9.6.1 Gerbang yang berbohong

Rilis Profitina di Windows dahulu digerakkan oleh sebuah *batch script*: berkas teks berisi perintah yang dijalankan baris demi baris oleh penerjemah perintah Windows, `cmd.exe`. Ada sifat `cmd.exe` yang jarang diketahui: ia tidak membaca seluruh skrip sebelum menjalankannya. Ia membaca berkas *menurut posisi byte* sambil berjalan, mengingat di mana ia berada. Jika berkas skrip berubah saat sedang berjalan, penerjemah melanjutkan dari posisi byte yang sama di berkas yang *baru*, yang mungkin kini berada di tengah baris lain.

Pada suatu rilis, sebuah *commit* yang menyentuh skrip rilis masuk ketika skrip sedang berjalan. Penerjemah melanjutkan dari posisi yang bergeser, melewatkan sebagian pekerjaannya, dan sampai ke baris-baris yang mencetak pesan berhasil. Gerbang melaporkan rilis yang berhasil untuk versi yang sebenarnya bukan versi yang sedang berjalan.

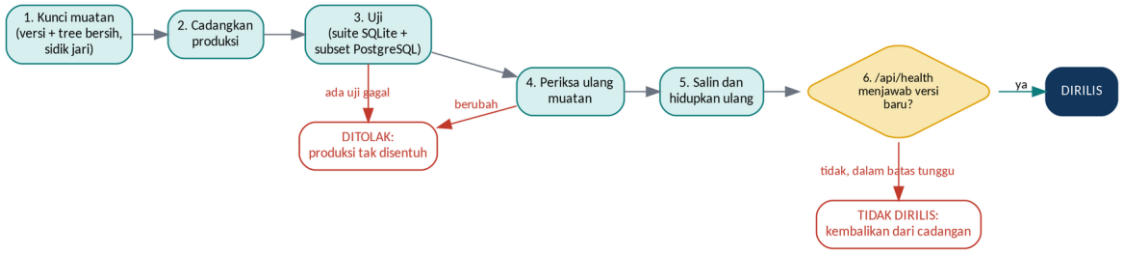
Perhatikan apa yang gagal. Bukan ujinya; uji sudah lolos. Bukan penyalinannya; mungkin penyalinan pun terjadi. Yang gagal adalah *klaimnya*: gerbang menyatakan "dirilis" berdasarkan telah sampainya ia di akhir skrip, bukan berdasarkan bukti bahwa versi baru sedang melayani pengguna.

9.6.2 Perbaikannya: berhasil berarti layanan sendiri yang mengatakannya

Dua perubahan menutup celah itu. Yang pertama bersifat mekanis: rilis menjalankan *salinan* skrip, sehingga suntingan pada skrip asli tidak dapat memengaruhi rilis yang sedang berlangsung. Yang kedua adalah yang terpenting: **rilis baru menyatakan berhasil setelah *endpoint* kesehatan layanan menjawab dengan nomor versi yang baru**. Gerbang tidak lagi memercayai dirinya sendiri; ia bertanya kepada sistem yang sedang berjalan.

9.6.3 Enam langkah

Rilis, sebagaimana adanya pada rilis September 2026, adalah urutan enam langkah yang digambarkan pada **Gambar 9.2**.



Gambar 9.2 Rilis enam langkah yang diverifikasi kesehatannya. Uji yang gagal atau muatan yang berubah selama gerbang berjalan membuat rilis ditolak tanpa menyentuh produksi; versi baru yang tidak menjawab pemeriksaan kesehatan tepat waktu dikembalikan (*rollback*).

1. **Kunci muatan.** Catat dengan tepat versi mana yang dirilis (*commit*-nya, dan *working tree* yang bersih tanpa suntingan yang belum di-*commit*), sehingga semua langkah berikutnya merujuk pada satu hal yang pasti.
2. **Cadangkan.** Ambil cadangan produksi sebelum apa pun berubah. Jika rilis gagal, ada keadaan baik yang diketahui untuk kembali.
3. **Uji.** Jalankan suite SQLite penuh dan subset prioritas PostgreSQL. Kegagalan apa pun berhenti di sini, dan produksi belum disentuh.
4. **Periksa ulang muatan.** Pastikan versinya masih versi yang dikunci pada langkah 1. Jika seseorang melakukan *commit* selama uji berjalan, kode yang diuji bukan lagi kode yang akan disalin, dan rilis berhenti.
5. **Salin dan hidupkan ulang.** Salin rilis ke tempatnya lalu hidupkan ulang layanan.
6. **Verifikasi.** Tanyakan kepada `/api/health` versi mana yang berjalan. Hanya ketika ia menjawab dengan versi baru, rilis dinyatakan berhasil. Jika tidak dalam waktu tunggu yang wajar, rilis gagal, dan cadangan dari langkah 2 membuat pengembalian menjadi hal rutin.

Ada satu lagi sifat urutan ini yang layak diperhatikan: langkah-langkah yang mahal atau berisiko diletakkan sesudah langkah-langkah yang murah dan aman. Mengunci muatan dan mencadangkan hampir tidak berbiaya; menguji memakan beberapa menit tetapi tidak menyentuh apa pun; hanya langkah 5 yang benar-benar mengubah produksi. Dengan susunan ini, sebagian besar kegagalan terjadi *sebelum* ada yang berubah, dan ketika kegagalan terjadi sesudahnya, cadangan dari langkah 2 sudah menunggu. Prinsip ini berlaku jauh di luar perangkat lunak: tukang listrik mematikan sekering sebelum membuka stopkontak, bukan sesudahnya.

Kata *bertahap* pada judul subbab ini merujuk pada langkah 1 sampai 4: rilis dirakit dan dibuktikan di area *staging* sebelum apa pun menyentuh salinan produksi.

> **CONTOH.** Mengapa langkah 4 diperlukan kalau langkah 1 sudah mengunci versi? Misalkan uji berlangsung delapan menit. Selama delapan menit itu seorang pengembang melakukan

> *commit* perubahan kecil. Tanpa langkah 4, penyalinan pada langkah 5 akan mengambil berkas terbaru, termasuk perubahan yang belum diuji. Gerbang akan merilis kode yang tidak pernah diujinya dan melaporkan seolah-olah sudah. Langkah 4 mengubah ketidakcocokan diam-diam itu menjadi penolakan.

9.6.4 Apa yang dilaporkan endpoint kesehatan

Endpoint kesehatan adalah URL yang dapat dipanggil alat pemantau, dan skrip rilis, untuk bertanya "apakah kamu bekerja?". Panduan arsitektur Microsoft menggambarkan pola ini sebagai pemeriksaan fungsional yang dibuka melalui sebuah *endpoint* yang "dapat diakses alat eksternal secara berkala", di mana "kode respons menunjukkan status aplikasi, dengan informasi opsional tentang kesehatan komponen dan layanan" [10].

`/api/health` milik Profitina melaporkan versi yang berjalan dan keadaan dependensinya: basis data, AI lokal, disk, lama menyala, dan jumlah galat server terbaru. Dua rincian penting.

Ia melaporkan versi yang dijalankan proses, bukan versi di disk. Inilah yang membuat langkah 6 bermakna: proses yang tidak pernah dihidupkan ulang tetap melaporkan versi lamanya, berapa pun banyaknya berkas baru yang disalin di sampingnya.

Ia membedakan "down" (mati) dari "degraded" (menurun). Jika basis data tidak dapat dijangkau, layanan tidak dapat melayani pengguna dan harus menyatakannya dengan status galat. Jika hanya jalur AI yang tidak tersedia, layanan tetap bekerja, dan jawaban jujurnya adalah "bekerja, tanpa AI", bukan "gagal". *Endpoint* praktikum di Subbab 9.10 membuat pembedaan yang sama.

Observabilitas melangkah sedikit lebih jauh daripada kesehatan. Setiap galat server (HTTP 5xx) dicatat tanpa isi permintaan maupun *header*, dengan alamat klien di-*hash*, dan peringatan lonjakan mengawasi kenaikan mendadak. Log galat dihapus setelah 90 hari oleh tugas pembersihan yang benar-benar berjalan; janji retensi yang tidak pernah dijalankan bukanlah janji.

9.7 Kisah Kinerja: Berpindah Kanal

Subbab ini menceritakan sebuah penyelidikan kinerja dari awal sampai akhir, karena kebiasaan yang diperlihatkannya lebih berharga daripada hasilnya.

9.7.1 Gejalanya

Pada September 2026 pengguna melaporkan bahwa berpindah kanal di feed komunitas "sangat lambat". Gejalanya samar, seperti gejala pada umumnya. Tugas pertama seorang insinyur adalah mengubah perasaan menjadi angka. Tim mengukur `/api/feed` untuk satu halaman berisi 30 unggahan dan menghitung apa yang dikerjakannya. Jawabannya mengejutkan: **868 milidetik**, **335 kueri basis data**, **122 koneksi basis data baru**, dan **6 panggilan ke model AI**, untuk satu halaman unggahan.

Cara mengukur juga penting. Satu kali pengukuran dapat menyesatkan: permintaan pertama sering lebih lambat karena *cache* masih dingin, dan permintaan yang kebetulan bersamaan dengan pekerjaan lain dapat tampak lambat tanpa alasan. Karena itu, pengukuran yang layak dipercaya mengulang permintaan yang sama beberapa kali, membuang pemanasan, dan melaporkan angka tipikal seperti median. Tim juga menghitung *pekerjaan*, bukan hanya *waktu*: jumlah kueri, koneksi, dan panggilan model tidak bergantung pada seberapa sibuk mesin saat itu,

sehingga angka-angka itu tetap bermakna di komputer mana pun. Waktu memberi tahu bahwa ada masalah; hitungan pekerjaan memberi tahu di mana letaknya.

9.7.2 Tiga akar penyebab

Penyelidikan menemukan tiga penyebab terpisah, masing-masing biasa saja bila berdiri sendiri.

Panggilan AI di jalur baca. Ketika menampilkan unggahan yang berisi tautan, kode meminta pendapat model lokal tentang setiap tautan yang tampak mencurigakan. Itu wajar dilakukan *sekali*, saat unggahan dibuat. Dilakukan pada setiap penampilan, untuk setiap pembaca, hal itu menaruh panggilan model di dalam gelung yang menggambar feed.

Satu koneksi baru per panggilan. Setiap fungsi pembantu membuka koneksinya sendiri ke PostgreSQL. Membuka koneksi melibatkan jabat tangan jaringan dan autentikasi; melakukannya 122 kali per halaman adalah beban murni.

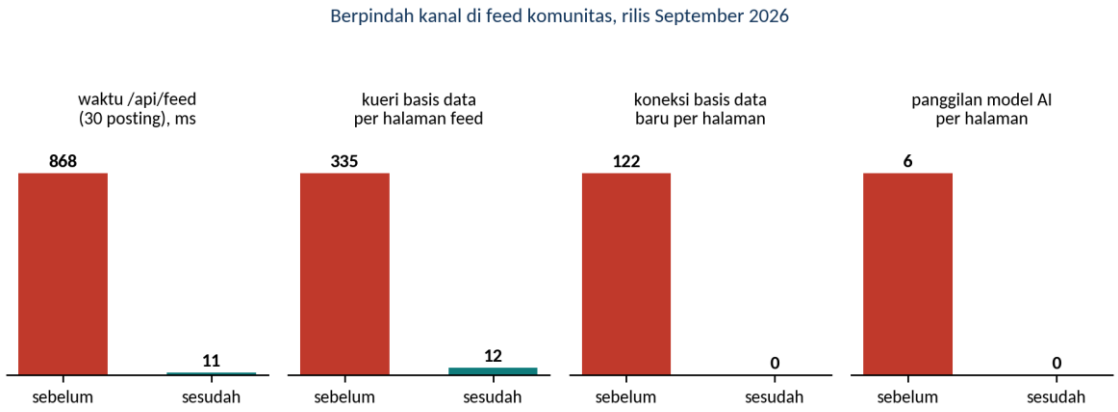
Pola kueri N+1. Feed memuat 30 unggahan dengan satu kueri, lalu, untuk setiap unggahan, menjalankan kueri tambahan untuk pembuat unggahannya, kanalnya, jumlah balasan dan suka, dan seterusnya, sekitar sebelas per unggahan. Jika satu halaman berisi N unggahan dan setiap unggahan memerlukan k kueri tambahan, totalnya

$$Q(N) = 1 + kN \tag{9.3}$$

Dengan $N = 30$ dan $k \approx 11$, Persamaan (9.3) memberikan sekitar 331 kueri, dekat dengan 335 yang terukur. Nama “N+1” berasal persis dari bentuk ini: satu kueri untuk daftar, lalu N lagi (di sini kN) untuk anggotanya. Perbaikannya, *prefetched hydration* (pengisian yang diambil di muka), mengumpulkan apa yang dibutuhkan ke-30 unggahan dan mengambilnya dalam segelintir kueri, satu per jenis data, sehingga jumlah kueri tidak lagi bertambah seiring N.

9.7.3 Perbaikan dan hasilnya

Perbaikannya terdiri atas tiga bagian, satu untuk setiap penyebab: jalur baca tidak pernah memanggil model (pendapat tentang tautan dihitung saat unggahan ditulis lalu disimpan); koneksi diambil dari *pool* (kolam) di dalam proses, dibuka sekali dan dipakai ulang; dan pengisian data diambil di muka. **Gambar 9.3** memperlihatkan hasilnya.



Gambar 9.3 Berpindah kanal di feed komunitas, sebelum dan sesudah perbaikan September 2026: waktu untuk halaman 30 unggahan, kueri basis data, koneksi baru, dan panggilan AI per halaman.

Feed kini memerlukan **11 milidetik**, dengan **12 kueri**, **tanpa koneksi baru**, dan **tanpa panggilan AI**. Faktor percepatannya adalah

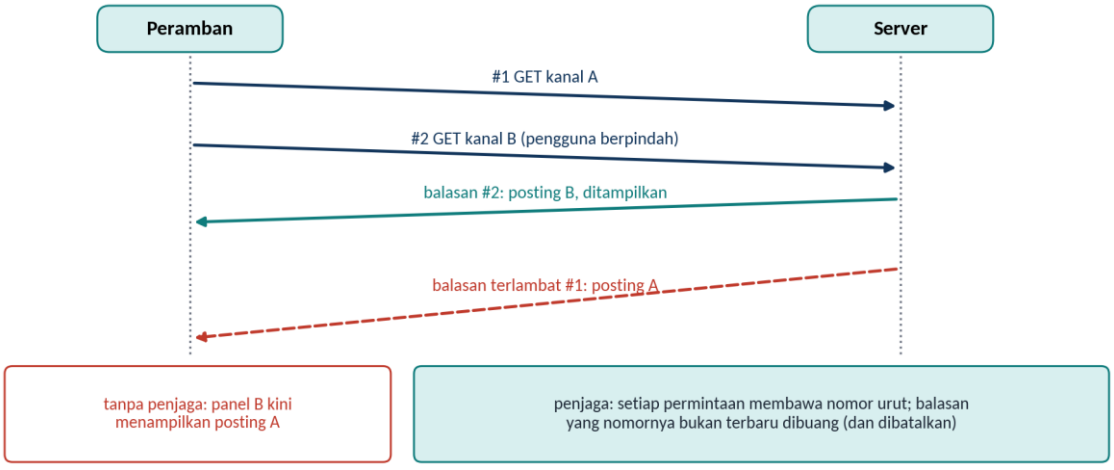
$$S = \frac{868\ \text{ms}}{11\ \text{ms}} \approx 79 \tag{9.4}$$

Perlakuan yang sama menurunkan *endpoint* dasbor dari 145 menjadi 15 milidetik dan widget komunitas tersemat dari 264 kueri menjadi 8. Dalam peramban Chromium sungguhan, ke-30 kartu feed kini tampil sekitar 143 milidetik setelah perpindahan kanal, yang dirasakan orang sebagai seketika.

9.7.4 Aturannya, dan balapannya

Perbaikan yang tidak dituliskan sebagai aturan akan dibatalkan oleh perubahan berikutnya yang bermaksud baik. Tim menarik satu aturan: **fungsi yang dipanggil per butir saat menampilkan tidak boleh menyentuh model atau membuka koneksinya sendiri.** Aturan itu cukup pendek untuk diingat dan cukup spesifik untuk diperiksa saat tinjauan kode, dan kini ada uji yang menghitung kueri pada jalur feed, sehingga kembalinya pola N+1 menggagalkan gerbang rilis di Subbab 9.5.

Penyelidikan itu juga menemukan bug kebenaran yang bersembunyi di balik kelambatan: *stale-response race* (balapan respons basi), yang digambarkan pada **Gambar 9.4**. Pengguna membuka kanal A, lalu cepat berpindah ke B. Kedua permintaan sedang dalam perjalanan. Jika balasan B tiba lebih dulu dan balasan A datang terlambat, halaman yang naif menampilkan unggahan B lalu menimpanya dengan unggahan A, di bawah judul B. Ketika setiap permintaan memakan hampir satu detik, hal ini sering terjadi.



Gambar 9.4 Balapan respons basi. Balasan terlambat untuk permintaan yang lebih awal tidak boleh menimpa balasan untuk permintaan terbaru; nomor urut pada setiap permintaan, yang diperiksa saat balasan tiba, membuang balasan yang basi.

Perbaikannya memberi setiap permintaan nomor urut dan mengabaikan balasan yang nomornya bukan yang terbaru; `AbortController` milik peramban juga membatalkan permintaan yang sudah usang, sehingga tidak lagi memakai jaringan [11]. Penjaga itu kemudian diterapkan pada *setiap* panel yang memuat data per kanal, bukan hanya feed, karena bug yang ditemukan di satu tempat biasanya juga tinggal di tetangganya. Soal 9.7 meminta Anda menulis penjaga itu.

> **WAWASAN KUNCI:** Ukur dulu, namai penyebabnya, perbaiki satu per satu, tulis aturannya, dan tambahkan uji yang menegakkan aturan itu. Kecepatan yang tidak dilindungi uji hanyalah pinjaman.

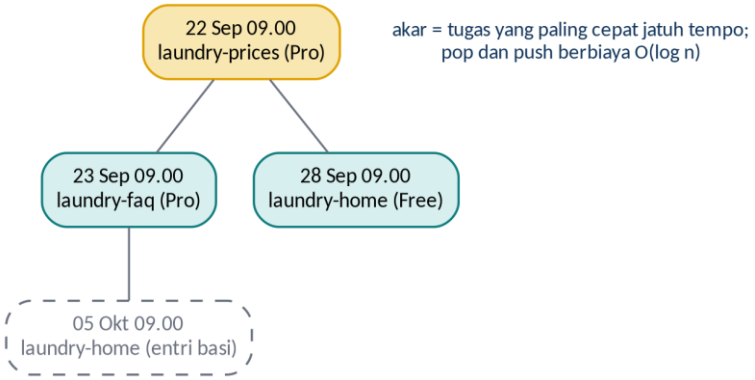
9.8 Menjadwalkan Pindaian Ulang: Priority Queue

Bab 8 mencantumkan pemantauan di antara perbedaan tingkat layanan: Free mendapat satu monitor yang dipindai ulang setiap minggu, Pro hingga lima puluh yang dipindai ulang setiap hari. Harus ada yang memutuskan, setiap saat, monitor mana yang jatuh tempo berikutnya. Yang memutuskan itu adalah *scheduler* (penjadwal), dan struktur data di jantungnya adalah *priority queue*.

9.8.1 Masalahnya

Misalkan ada n monitor, masing-masing dengan *waktu jatuh tempo*, yaitu saat pindaian berikutnya harus berjalan. Penjadwal berulang kali memerlukan tiga operasi: menemukan monitor dengan waktu jatuh tempo paling awal; mengeluarkannya; dan memasukkannya kembali dengan waktu jatuh tempo baru setelah dijalankan. Daftar biasa akan membuat salah satunya lambat: menjaga daftar tetap terurut membuat penyisipan berbiaya hingga n langkah, dan membiarkannya tak terurut membuat pencarian yang paling awal berbiaya n langkah.

Binary min-heap (tumpukan-min biner) mengerjakan ketiganya dengan baik. Ia adalah pohon biner yang disimpan dalam larik, dengan kunci setiap simpul tidak lebih besar daripada kunci anak-anaknya, sehingga kunci terkecil selalu berada di akar [12]. Melihat akar berbiaya satu langkah; mengeluarkan akar atau menyisipkan entri baru berbiaya sebanyak langkah yang sebanding dengan tinggi pohon, yaitu sekitar $\log_2 n$. **Gambar 9.5** memperlihatkan heap kecil berisi pindaian terjadwal.



Gambar 9.5 Min-heap pindaian terjadwal yang diurutkan menurut waktu jatuh tempo. Akar adalah tugas berikutnya; entri bergaris putus-putus adalah entri basi yang tertinggal akibat penjadwalan ulang, dan dilewati ketika mencapai akar.

Jika penjadwal melakukan m operasi *pop* dan *push* dalam sehari atas heap berisi n entri, total pekerjaannya adalah

$$W = O(m \log n) \tag{9.5}$$

Dengan $n = 1{,}000$ monitor, $\log_2 n \approx 10$, sehingga setiap operasi kira-kira sepuluh langkah. Modul `heapq` Python dan kelas `PriorityQueue` Java sama-sama mengimplementasikan heap biner [13], [14].

9.8.2 Irama dan banyaknya pemindaian yang diharapkan

Setiap tingkat memiliki *irama* c: tujuh hari untuk Free, satu hari untuk Pro. Selama periode sepanjang P yang dimulai dengan pindaian pada waktu nol, sebuah monitor berjalan pada waktu $0, c, 2c, \ldots$ yang berada di dalam periode, yaitu sebanyak

$$r = \lceil \frac{P}{c} \rceil \tag{9.6}$$

kali, dengan $\lceil x \rceil$ bilangan bulat terkecil yang tidak kurang dari x. Untuk periode 14 hari, Persamaan (9.6) memberikan $\lceil 14/7 \rceil = 2$ pindaian Free dan $\lceil 14/1 \rceil = 14$ pindaian Pro; uji dalam praktikum memeriksa persis angka-angka ini.

9.8.3 Rincian yang membedakan mainan dari alat

Penghapusan malas. Ketika sebuah monitor dihapus atau tingkatnya berubah, entri lamanya masih berada di suatu tempat di dalam heap, dan mengeluarkan entri dari tengah heap itu merepotkan. Sebagai gantinya, setiap monitor membawa nomor *versi* yang naik setiap kali ia dijadwalkan ulang, dan setiap entri heap mencatat versi saat entri itu dibuat. Ketika sebuah entri mencapai akar, penjadwal membandingkan versinya; ketidakcocokan berarti entri itu basi, dan entri itu cukup dibuang. Kotak bergaris putus-putus pada **Gambar 9.5** adalah entri semacam itu.

Pekerja yang terlambat melompat, bukan menyembur. Jika mesin mati selama tiga hari, sebuah monitor Pro telah melewatkan tiga slot harian. Menjalankan ketiganya begitu mesin kembali akan menghasilkan semburan pindaian yang identik, membebani sistem pada saat terburuk dan tidak memberi tahu pengguna hal baru. Penjadwal justru menjalankan monitor itu sekali lalu memajukan waktu jatuh tempo berikutnya ke slot pertama di masa depan, sambil menghitung slot yang dilompati. Jika t adalah waktu sekarang dan d waktu jatuh tempo lama, banyaknya slot yang dilompati adalah

$$s = \lfloor \frac{t - d}{c} \rfloor \tag{9.7}$$

Untuk monitor Pro yang jatuh tempo pada hari ke-1 dan pekerjanya kembali pada hari ke-5 lewat satu jam, Persamaan (9.7) memberikan $\lfloor 4.04 \rfloor = 4$ slot yang dilompati, dan waktu jatuh tempo berikutnya adalah hari ke-6.

Idempoten. Menjalankan penjadwal dua kali pada saat yang sama tidak boleh menjalankan apa pun dua kali. Karena tugas yang sudah dijalankan dimasukkan kembali dengan waktu jatuh tempo yang *lebih lambat*, panggilan kedua pada saat yang sama tidak menemukan apa pun yang jatuh tempo. Operasi yang efeknya sama baik dilakukan sekali maupun berkali-kali disebut *idempoten*, dan itulah sifat yang membuat penjadwal aman dipanggil oleh pengatur waktu yang mungkin berbunyi dua kali.

Jam simulasi. Penjadwal tidak pernah membaca jam sungguhan; ia diberi tahu pukul berapa sekarang. Dengan begitu, tiga minggu penjadwalan dapat diuji dalam satu milidetik, termasuk gangguan tiga hari, sesuatu yang tidak mungkin dilakukan uji dengan jam sungguhan.

9.9 Praktikum G4, Bagian 1: Penjadwal

Praktikum G4 terdiri atas tiga bagian. Bagian ini membangun penjadwal, dan uji-ujinya menjadi suite uji yang akan dijaga gerbang rilis pada Bagian 3.

9.9.1 Program Python

Kode 9.1 adalah penjadwalnya, dengan peragaan yang menjalankan 21 hari simulasi untuk tiga halaman Profitina Laundry.

Kode 9.1 `scheduler.py`: penjadwal min-heap dengan irama per tingkat, penghapusan malas, pengejaran yang melompat alih-alih menyembur, dan jam simulasi.

```
"""A tiny re-scan scheduler: a min-heap of (due time, monitor) with a
per-tier cadence, driven by a simulated clock."""
import heapq
import zlib
from dataclasses import dataclass
from datetime import datetime, timedelta
from zoneinfo import ZoneInfo

WIB = ZoneInfo("Asia/Jakarta")
CADENCE = {"free": timedelta(days=7), "pro": timedelta(days=1)}


@dataclass
class Monitor:
    mid: str
    tier: str
    version: int = 0              # bumped whenever the monitor is rescheduled
    next_due: datetime | None = None
    last_run: datetime | None = None


def stub_scan(mid: str, when: datetime) -> int:
    """Stands in for a real scan: a repeatable score from 50 to 89."""
    key = f"{mid}|{when.astimezone(WIB):%Y-%m-%d}".encode()
    return 50 + zlib.crc32(key) % 40


class Scheduler:
    def __init__(self, scan=stub_scan):
        self.scan = scan
        self.heap: list[tuple[datetime, int, str, int]] = []
        self.seq = 0                    # breaks ties in insertion order
        self.monitors: dict[str, Monitor] = {}
        self.results: list[tuple[str, datetime, int]] = []
        self.skipped = 0

    def _push(self, m: Monitor, due: datetime) -> None:
        m.version += 1                  # older heap entries become stale
        m.next_due = due
        self.seq += 1
        heapq.heappush(self.heap, (due, self.seq, m.mid, m.version))

    def add(self, m: Monitor, now: datetime) -> None:
        self.monitors[m.mid] = m
        self._push(m, now)              # first scan as soon as possible
```

Kode 9.1 (Lanjutan) *scheduler.py: penjadwal min-heap dengan irama per tingkat, penghapusan malas, pengejaran yang melompat alih-alih menyembur, dan jam simulasi.*

```
    def remove(self, mid: str) -> None:
        self.monitors.pop(mid, None)   # its heap entry is skipped later

    def change_tier(self, mid: str, tier: str, now: datetime) -> None:
        m = self.monitors[mid]
        m.tier = tier
        base = m.last_run or now
        self._push(m, max(now, base + CADENCE[tier]))

    def run_due(self, now: datetime) -> int:
        """Run every job whose time has come; return how many ran."""
        ran = 0
        while self.heap and self.heap[0][0] <= now:
            due, _, mid, version = heapq.heappop(self.heap)
            m = self.monitors.get(mid)
            if m is None or m.version != version:
                continue               # removed or rescheduled: stale entry
            self.results.append((mid, now, self.scan(mid, now)))
            m.last_run = now
            ran += 1
            step = CADENCE[m.tier]
            nxt = due + step
            while nxt <= now:          # we were late: skip, never burst
                nxt += step
                self.skipped += 1
            self._push(m, nxt)
        return ran


def at(text: str) -> datetime:
    return datetime.strptime(text, "%Y-%m-%d %H:%M").replace(tzinfo=WIB)


def main() -> None:
    s = Scheduler()
    start = at("2026-09-21 09:00")
    s.add(Monitor("laundry-home", "free"), start)
    s.add(Monitor("laundry-prices", "pro"), start)
    down_from, down_to = at("2026-10-01 00:00"), at("2026-10-03 23:59")
    days = 21
    for step in range(days * 8):               # every 3 hours
        now = start + timedelta(hours=3 * step)
        if now == at("2026-09-23 09:00"):
            s.add(Monitor("laundry-faq", "pro"), now)
        if now == at("2026-09-30 12:00"):
            s.change_tier("laundry-home", "pro", now)
```

Kode 9.1 (Lanjutan) *scheduler.py: penjadwal min-heap dengan irama per tingkat, penghapusan malas, pengejaran yang melompat alih-alih menyembur, dan jam simulasi.*

```
        if now == at("2026-10-06 09:00"):
            s.remove("laundry-faq")
        if down_from <= now <= down_to:
            continue                            # the machine is off
        s.run_due(now)

    first = start.date()
    print("day of month    " + "".join(str((first + timedelta(d)).day % 10)
                                       for d in range(days)))
    for mid in ("laundry-home", "laundry-prices", "laundry-faq"):
        row = ["."] * days
        for d in range(days):
            day = first + timedelta(d)
            if down_from.date() <= day <= down_to.date():
                row[d] = "-"
        for rmid, when, _ in s.results:
            if rmid == mid:
                row[(when.date() - first).days] = "x"
        print(f"{mid:<15}{''.join(row)}")
    print("(last digit of the date, 21 Sep to 11 Oct 2026)")
    print("x = scan ran, . = nothing due, - = machine off")
    print("runs:", len(s.results), "| skipped slots:", s.skipped,
          "| heap entries left:", len(s.heap))
    for mid, when, score in s.results[:4]:
        print(f"  {when:%d %b %H:%M} WIB  {mid:<15} score {score}")


if __name__ == "__main__":
    main()
```

Heap menyimpan *tuple* `(due, seq, mid, version)`. Python membandingkan *tuple* unsur demi unsur, sehingga heap mengurutkan entri menurut waktu jatuh tempo lebih dulu; `seq`, penghitung yang hanya bertambah, memutus seri menurut urutan penyisipan sehingga dua monitor yang jatuh tempo pada saat yang sama tidak pernah memaksa Python membandingkan sisa *tuple*. `_push` adalah satu-satunya tempat entri dibuat, dan ia selalu menaikkan versi monitor lebih dulu.

`run_due` adalah jantung program. Selama akar heap sudah jatuh tempo, ia mengeluarkan entri, membuangnya bila basi, menjalankan pindaian, mencatat hasilnya, dan menghitung waktu jatuh tempo berikutnya dari *waktu jatuh tempo lama*, bukan dari sekarang, sehingga monitor harian tetap berada di kisi hariannya. Gelung `while` di dalamnya adalah Persamaan (9.7) yang dikerjakan dengan cara menghitung.

`stub_scan` menggantikan pindaian sungguhan. Fungsi ini mengembalikan angka 50 sampai 89 yang dapat diulang, dihitung dengan CRC-32 dari nama monitor dan tanggal WIB, sehingga versi Python dan Java mencetak skor yang sama. Angka itu bukan skor visibilitas dan tidak berpura-pura menjadi skor visibilitas; praktikum G1 di Bab 3 adalah tempat untuk memasang pemeriksa sungguhan.

Peragaannya menambahkan monitor Free untuk beranda binatu dan monitor Pro untuk halaman harganya pada Senin 21 September 2026, menambahkan monitor Pro untuk halaman FAQ-nya dua hari kemudian, meningkatkan beranda ke Pro pada 30 September, mematikan mesin dari 1 sampai 3 Oktober, dan menghapus monitor FAQ pada 6 Oktober. Program mencetak kalender kecil, satu kolom per hari:

```
day of month   12345678901234567890
laundry-home   x.....x.x---xxxxxxxx
laundry-prices xxxxxxxxxx---xxxxxxxx
laundry-faq    ..xxxxxxxx---xx......
(last digit of the date, 21 Sep to 11 Oct 2026)
x = scan ran, . = nothing due, - = machine off
runs: 42 | skipped slots: 6 | heap entries left: 2
  21 Sep 09:00 WIB  laundry-home    score 71
  21 Sep 09:00 WIB  laundry-prices  score 81
  22 Sep 09:00 WIB  laundry-prices  score 79
  23 Sep 09:00 WIB  laundry-prices  score 77
```

Bacalah baris-barisnya. `laundry-home` berjalan pada 21 September dan seminggu kemudian pada tanggal 28, sebagaimana mestinya monitor Free. Pada tanggal 30, peningkatan langganan langsung memajukan pindaian berikutnya, karena sudah lewat sehari sejak pindaian terakhirnya, dan sejak itu ia berjalan setiap hari. Ketiga monitor menunjukkan `-` untuk tiga hari mesin mati, lalu berlanjut pada 4 Oktober dengan satu kali jalan masing-masing: enam slot yang terlewat dilompati (dua untuk setiap monitor Pro), tidak diputar ulang. `laundry-faq` berhenti setelah 5 Oktober. Pada akhirnya tersisa dua entri heap, satu untuk setiap monitor yang masih hidup; entri terakhir monitor FAQ dibuang sebagai entri basi ketika mencapai akar.

9.9.2 Program Java

Kode 9.2 adalah penjadwal yang sama dalam Java. `PriorityQueue` menerima sebuah pembanding; `Comparator.comparing(Entry::due).thenComparingLong(Entry::seq)` menyatakan “menurut waktu jatuh tempo, lalu menurut urutan” secara langsung. Keluarannya identik dengan keluaran Python di atas.

Kode 9.2 `Scheduler.java`: penjadwal yang sama dengan `java.util.PriorityQueue` dan *record*.

```
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.PriorityQueue;
import java.util.zip.CRC32;

/** A tiny re-scan scheduler: a min-heap of (due time, monitor) with a
 *  per-tier cadence, driven by a simulated clock. */
public class Scheduler {
    static final ZoneId WIB = ZoneId.of("Asia/Jakarta");
    static final Map<String, Duration> CADENCE = Map.of(
            "free", Duration.ofDays(7), "pro", Duration.ofDays(1));

    static class Monitor {
        final String mid;
        String tier;
        int version = 0;        // bumped whenever the monitor is rescheduled
        ZonedDateTime nextDue;
        ZonedDateTime lastRun;
        Monitor(String mid, String tier) { this.mid = mid; this.tier = tier; }
    }

    record Entry(ZonedDateTime due, long seq, String mid, int version) { }
    record Result(String mid, ZonedDateTime when, int score) { }

    /** Stands in for a real scan: a repeatable score from 50 to 89. */
    static int stubScan(String mid, ZonedDateTime when) {
        String day = when.withZoneSameInstant(WIB).toLocalDate().toString();
        CRC32 crc = new CRC32();
        crc.update((mid + "|" + day).getBytes(StandardCharsets.UTF_8));
        return 50 + (int) (crc.getValue() % 40);
    }
```

Kode 9.2 (Lanjutan) *Scheduler.java: penjadwal yang sama dengan java.util.PriorityQueue dan record.*

```
    final PriorityQueue<Entry> heap = new PriorityQueue<>(
            Comparator.comparing(Entry::due).thenComparingLong(Entry::seq));
    long seq = 0;                       // breaks ties in insertion order
    final Map<String, Monitor> monitors = new HashMap<>();
    final List<Result> results = new ArrayList<>();
    int skipped = 0;

    private void push(Monitor m, ZonedDateTime due) {
        m.version++;                    // older heap entries become stale
        m.nextDue = due;
        heap.add(new Entry(due, ++seq, m.mid, m.version));
    }

    void add(Monitor m, ZonedDateTime now) {
        monitors.put(m.mid, m);
        push(m, now);                   // first scan as soon as possible
    }

    void remove(String mid) {
        monitors.remove(mid);           // its heap entry is skipped later
    }

    void changeTier(String mid, String tier, ZonedDateTime now) {
        Monitor m = monitors.get(mid);
        m.tier = tier;
        ZonedDateTime base = m.lastRun != null ? m.lastRun : now;
        ZonedDateTime due = base.plus(CADENCE.get(tier));
        push(m, due.isAfter(now) ? due : now);
    }

    /** Run every job whose time has come; return how many ran. */
    int runDue(ZonedDateTime now) {
        int ran = 0;
        while (!heap.isEmpty() && !heap.peek().due().isAfter(now)) {
            Entry e = heap.poll();
            Monitor m = monitors.get(e.mid());
            if (m == null || m.version != e.version())
                continue;               // removed or rescheduled: stale entry
            results.add(new Result(m.mid, now, stubScan(m.mid, now)));
            m.lastRun = now;
            ran++;
            Duration step = CADENCE.get(m.tier);
            ZonedDateTime next = e.due().plus(step);
            while (!next.isAfter(now)) {  // we were late: skip, never burst
                next = next.plus(step);
                skipped++;
```

Kode 9.2 (Lanjutan) *Scheduler.java: penjadwal yang sama dengan java.util.PriorityQueue dan record.*

```
            }
            push(m, next);
        }
        return ran;
    }

    static ZonedDateTime at(String text) {
        DateTimeFormatter f = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        return LocalDateTime.parse(text, f).atZone(WIB);
    }

    public static void main(String[] args) {
        Scheduler s = new Scheduler();
        ZonedDateTime start = at("2026-09-21 09:00");
        s.add(new Monitor("laundry-home", "free"), start);
        s.add(new Monitor("laundry-prices", "pro"), start);
        ZonedDateTime downFrom = at("2026-10-01 00:00");
        ZonedDateTime downTo = at("2026-10-03 23:59");
        int days = 21;
        for (int step = 0; step < days * 8; step++) {      // every 3 hours
            ZonedDateTime now = start.plusHours(3L * step);
            if (now.equals(at("2026-09-23 09:00")))
                s.add(new Monitor("laundry-faq", "pro"), now);
            if (now.equals(at("2026-09-30 12:00")))
                s.changeTier("laundry-home", "pro", now);
            if (now.equals(at("2026-10-06 09:00")))
                s.remove("laundry-faq");
            if (!now.isBefore(downFrom) && !now.isAfter(downTo))
                continue;                                  // the machine is off
            s.runDue(now);
        }

        LocalDate first = start.toLocalDate();
        StringBuilder head = new StringBuilder("day of month   ");
        for (int d = 0; d < days; d++)
            head.append(first.plusDays(d).getDayOfMonth() % 10);
        System.out.println(head);
        for (String mid : new String[] {"laundry-home", "laundry-prices",
                                        "laundry-faq"}) {
            char[] row = ".".repeat(days).toCharArray();
            for (int d = 0; d < days; d++) {
                LocalDate day = first.plusDays(d);
                if (!day.isBefore(downFrom.toLocalDate())
                        && !day.isAfter(downTo.toLocalDate())) row[d] = '-';
            }
            for (Result r : s.results)
```

Kode 9.2 (Lanjutan) *Scheduler.java: penjadwal yang sama dengan java.util.PriorityQueue dan record.*

```
                if (r.mid().equals(mid))
                    row[(int) ChronoUnit.DAYS.between(first,
                            r.when().toLocalDate())] = 'x';
            System.out.printf("%-15s%s%n", mid, new String(row));
        }
        System.out.println("(last digit of the date, 21 Sep to 11 Oct 2026)");
        System.out.println("x = scan ran, . = nothing due, - = machine off");
        System.out.println("runs: " + s.results.size() + " | skipped slots: "
                + s.skipped + " | heap entries left: " + s.heap.size());
        DateTimeFormatter f = DateTimeFormatter.ofPattern("dd MMM HH:mm",
                Locale.US);
        for (Result r : s.results.subList(0, 4))
            System.out.printf("  %s WIB  %-15s score %d%n",
                    r.when().format(f), r.mid(), r.score());
    }
}
```

9.9.3 Uji-ujinya

Kode 9.3 dan **Kode 9.4** berisi tujuh uji, masing-masing mengunci satu aturan dari Subbab 9.8: irama paket, jumlah yang diharapkan menurut Persamaan (9.6), urutan heap, sifat idempoten, melompat alih-alih menyembur (Persamaan (9.7)), peningkatan langganan yang memajukan pindaian berikutnya, dan monitor yang dihapus tidak pernah berjalan lagi. Uji Java memakai pelari `MiniTest` dari Subbab 8.12.4, yang disalin ke folder bab ini.

Kode 9.3 `test_scheduler.py`: tujuh uji untuk penjadwal. Gerbang rilis menjalankan persis berkas ini.

```
"""Tests for the re-scan scheduler; the release gate runs these."""
import unittest
from datetime import timedelta

from scheduler import CADENCE, Monitor, Scheduler, at

T0 = at("2026-09-21 09:00")


def run_hourly(s: Scheduler, hours: int) -> None:
    for h in range(hours + 1):
        s.run_due(T0 + timedelta(hours=h))


class SchedulerTests(unittest.TestCase):
    def test_cadence_matches_the_plans(self):
        self.assertEqual(CADENCE["free"], timedelta(days=7))
        self.assertEqual(CADENCE["pro"], timedelta(days=1))

    def test_two_weeks_give_2_free_and_14_pro_scans(self):
        s = Scheduler()
        s.add(Monitor("f", "free"), T0)
        s.add(Monitor("p", "pro"), T0)
        run_hourly(s, 14 * 24 - 1)
        runs = [mid for mid, _, _ in s.results]
        self.assertEqual((runs.count("f"), runs.count("p")), (2, 14))

    def test_jobs_come_out_in_due_order(self):
        s = Scheduler()
        s.add(Monitor("late", "pro"), T0 + timedelta(hours=5))
        s.add(Monitor("early", "pro"), T0 + timedelta(hours=1))
        s.run_due(T0 + timedelta(hours=6))
        self.assertEqual([r[0] for r in s.results], ["early", "late"])

    def test_running_twice_at_the_same_time_is_idempotent(self):
        s = Scheduler()
        s.add(Monitor("p", "pro"), T0)
        self.assertEqual(s.run_due(T0), 1)
        self.assertEqual(s.run_due(T0), 0)
```

Kode 9.3 (Lanjutan) *test_scheduler.py: tujuh uji untuk penjadwal. Gerbang rilis menjalankan persis berkas ini.*

```
    def test_late_worker_skips_instead_of_bursting(self):
        s = Scheduler()
        s.add(Monitor("p", "pro"), T0)
        s.run_due(T0)
        self.assertEqual(s.run_due(T0 + timedelta(days=5, hours=1)), 1)
        self.assertEqual(s.skipped, 4)
        self.assertEqual(s.monitors["p"].next_due, T0 + timedelta(days=6))

    def test_upgrade_brings_the_next_scan_forward(self):
        s = Scheduler()
        s.add(Monitor("f", "free"), T0)
        s.run_due(T0)
        s.change_tier("f", "pro", T0 + timedelta(hours=2))
        self.assertEqual(s.monitors["f"].next_due, T0 + timedelta(days=1))

    def test_removed_monitor_never_runs_again(self):
        s = Scheduler()
        s.add(Monitor("p", "pro"), T0)
        s.run_due(T0)
        s.remove("p")
        run_hourly(s, 72)
        self.assertEqual(len(s.results), 1)
        self.assertEqual(s.heap, [])


if __name__ == "__main__":
    unittest.main()
```

Kode 9.4 `TestScheduler.java`: tujuh uji yang sama dalam Java.

```
import java.time.Duration;
import java.time.ZonedDateTime;

/** Tests for the re-scan scheduler; the release gate runs these. */
public class TestScheduler {
    static final ZonedDateTime T0 = Scheduler.at("2026-09-21 09:00");

    static void runHourly(Scheduler s, int hours) {
        for (int h = 0; h <= hours; h++) s.runDue(T0.plusHours(h));
    }

    public static void main(String[] args) {
        MiniTest t = new MiniTest();
        t.test("cadence matches the plans", () -> {
            MiniTest.equal(Duration.ofDays(7), Scheduler.CADENCE.get("free"));
            MiniTest.equal(Duration.ofDays(1), Scheduler.CADENCE.get("pro"));
        });
        t.test("two weeks give 2 Free and 14 Pro scans", () -> {
            Scheduler s = new Scheduler();
            s.add(new Scheduler.Monitor("f", "free"), T0);
            s.add(new Scheduler.Monitor("p", "pro"), T0);
            runHourly(s, 14 * 24 - 1);
            long f = s.results.stream().filter(r -> r.mid().equals("f")).count();
            long p = s.results.stream().filter(r -> r.mid().equals("p")).count();
            MiniTest.equal(2L, f);
            MiniTest.equal(14L, p);
        });
        t.test("jobs come out in due order", () -> {
            Scheduler s = new Scheduler();
            s.add(new Scheduler.Monitor("late", "pro"), T0.plusHours(5));
            s.add(new Scheduler.Monitor("early", "pro"), T0.plusHours(1));
            s.runDue(T0.plusHours(6));
            MiniTest.equal("early", s.results.get(0).mid());
            MiniTest.equal("late", s.results.get(1).mid());
        });
        t.test("running twice at the same time is idempotent", () -> {
            Scheduler s = new Scheduler();
            s.add(new Scheduler.Monitor("p", "pro"), T0);
            MiniTest.equal(1, s.runDue(T0));
            MiniTest.equal(0, s.runDue(T0));
        });
        t.test("late worker skips instead of bursting", () -> {
            Scheduler s = new Scheduler();
            s.add(new Scheduler.Monitor("p", "pro"), T0);
            s.runDue(T0);
            MiniTest.equal(1, s.runDue(T0.plusDays(5).plusHours(1)));
```

Kode 9.4 (Lanjutan) *TestScheduler.java: tujuh uji yang sama dalam Java.*

```
            MiniTest.equal(4, s.skipped);
            MiniTest.equal(T0.plusDays(6), s.monitors.get("p").nextDue);
        });
        t.test("upgrade brings the next scan forward", () -> {
            Scheduler s = new Scheduler();
            s.add(new Scheduler.Monitor("f", "free"), T0);
            s.runDue(T0);
            s.changeTier("f", "pro", T0.plusHours(2));
            MiniTest.equal(T0.plusDays(1), s.monitors.get("f").nextDue);
        });
        t.test("removed monitor never runs again", () -> {
            Scheduler s = new Scheduler();
            s.add(new Scheduler.Monitor("p", "pro"), T0);
            s.runDue(T0);
            s.remove("p");
            runHourly(s, 72);
            MiniTest.equal(1, s.results.size());
            MiniTest.equal(0, s.heap.size());
        });
        t.finish();
    }
}
```

```
.......
----------------------------------------------------------------------
Ran 7 tests in 0.001s

OK
ok    cadence matches the plans
ok    two weeks give 2 Free and 14 Pro scans
ok    jobs come out in due order
ok    running twice at the same time is idempotent
ok    late worker skips instead of bursting
ok    upgrade brings the next scan forward
ok    removed monitor never runs again
Ran 7 tests: 7 passed, 0 failed
```

9.10 Praktikum G4, Bagian 2: Endpoint Kesehatan

Bagian kedua adalah *endpoint* kesehatan yang menjawab `GET /api/health` dengan versi yang berjalan dan keadaan tiga dependensi. Bagian ini mengikuti aturan di Subbab 9.6.4.

9.10.1 Program Python

Kode 9.5 hanya memakai `http.server` dari pustaka standar.

Kode 9.5 `health.py`: *endpoint* kesehatan yang melaporkan versi yang berjalan dan tiga pemeriksaan, serta membedakan mati dari menurun.

```
"""A health-check endpoint: GET /api/health answers which version is
running and whether its dependencies work.
usage: python health.py [PORT] [APP_DIR]"""
import json
import shutil
import socket
import sys
import time
from http.server import BaseHTTPRequestHandler, ThreadingHTTPServer
from pathlib import Path

STARTED = time.monotonic()
APP_DIR = Path(sys.argv[2]) if len(sys.argv) > 2 else Path(__file__).parent
MIN_FREE_BYTES = 1 * 1024**3          # our lab's threshold: 1 GiB
# read once at start: a running process reports the code it was started with
VERSION = (APP_DIR / "VERSION").read_text(encoding="utf-8").strip()


def check_store() -> str:
    """Can we write a file and read the same bytes back?"""
    try:
        data = APP_DIR / "_data"
        data.mkdir(exist_ok=True)
        probe = data / "health.tmp"
        probe.write_text("ping", encoding="utf-8")
        return "ok" if probe.read_text(encoding="utf-8") == "ping" else "fail"
    except OSError:
        return "fail"


def check_disk() -> str:
    return "ok" if shutil.disk_usage(APP_DIR).free >= MIN_FREE_BYTES else "fail"


def check_ai() -> str:
    """Is a local model server listening? 127.0.0.1, never 'localhost'."""
    try:
        with socket.create_connection(("127.0.0.1", 11434), timeout=0.3):
            return "ok"
    except OSError:
        return "unavailable"
```

Kode 9.5 (Lanjutan) *health.py: endpoint kesehatan yang melaporkan versi yang berjalan dan tiga pemeriksaan, serta membedakan mati dari menurun.*

```
def health() -> tuple[int, dict]:
    checks = {"store": check_store(), "disk": check_disk(), "ai": check_ai()}
    if "fail" in (checks["store"], checks["disk"]):
        status, code = "down", 503          # cannot serve users
    elif checks["ai"] != "ok":
        status, code = "degraded", 200      # serves, and says what is missing
    else:
        status, code = "ok", 200
    return code, {"status": status, "version": VERSION, "checks": checks,
                  "uptime_s": int(time.monotonic() - STARTED)}


class Handler(BaseHTTPRequestHandler):
    def do_GET(self) -> None:
        if self.path != "/api/health":
            self.send_error(404)
            return
        code, body = health()
        raw = (json.dumps(body, indent=2) + "\n").encode()
        self.send_response(code)
        self.send_header("Content-Type", "application/json")
        self.send_header("Cache-Control", "no-store")   # never cache health
        self.send_header("Content-Length", str(len(raw)))
        self.end_headers()
        self.wfile.write(raw)

    def log_message(self, *args) -> None:
        pass                                    # keep the console quiet


def main() -> None:
    port = int(sys.argv[1]) if len(sys.argv) > 1 else 7090
    server = ThreadingHTTPServer(("127.0.0.1", port), Handler)
    print(f"health on http://127.0.0.1:{port}/api/health", flush=True)
    server.serve_forever()


if __name__ == "__main__":
    main()
```

Tiga pemeriksaan menggantikan dependensi layanan sungguhan. `check_store` menulis berkas kecil lalu membacanya kembali, uji jujur paling sederhana bahwa penyimpanan bekerja; dalam layanan sungguhan ini berupa `SELECT 1` ke basis data. `check_disk` membandingkan ruang kosong dengan ambang yang dipilih untuk praktikum. `check_ai` mencoba membuka koneksi ke server model lokal di `127.0.0.1` port 11434, alamat bawaan API Ollama [15], dengan batas waktu 0,3 detik agar pemeriksaan kesehatan itu sendiri tetap cepat.

`health` mengubah hasil pemeriksaan menjadi status. Penyimpanan atau disk yang gagal membuat layanan `down` dengan status HTTP 503 (*Service Unavailable*), karena ia tidak dapat melayani pengguna. AI yang tidak tersedia membuatnya `degraded` dengan status 200, karena ia tetap melayani, dan menyebut dengan tepat apa yang tidak ada. Semuanya baik hanya bila ketiga pemeriksaan lolos.

Versi dibaca **sekali, ketika proses mulai**, ke dalam konstanta `VERSION`. Satu baris inilah yang membuat gerbang rilis pada Bagian 3 jujur: jika berkas baru disalin di samping proses yang sedang berjalan tetapi proses itu tidak dihidupkan ulang, ia terus melaporkan versi lamanya. Responsnya membawa `Cache-Control: no-store`, karena jawaban kesehatan yang di-*cache* lebih buruk daripada tanpa jawaban.

Jalankan server di satu terminal, lalu tanyai dari terminal lain (Windows 10 dan 11 sudah menyertakan `curl`; di PowerShell, `Invoke-RestMethod` juga dapat dipakai):

```
python health.py 7090
```

```
health on http://127.0.0.1:7090/api/health
HTTP/1.0 200 OK
Server: BaseHTTP/0.6 Python/3.11.15
Date: Thu, 24 Sep 2026 04:45:34 GMT
Content-Type: application/json
Cache-Control: no-store
Content-Length: 146

{
  "status": "degraded",
  "version": "0.1.0",
  "checks": {
    "store": "ok",
    "disk": "ok",
    "ai": "unavailable"
  },
  "uptime_s": 1
}
```

Di komputer yang dipakai untuk buku ini tidak ada server model yang berjalan, sehingga `ai` bernilai `unavailable` dan statusnya `degraded`, dengan HTTP 200. Jalankan Ollama lalu tanyakan lagi, dan statusnya menjadi `ok`. Hentikan server dengan Ctrl+C.

9.10.2 Program Java

Kode 9.6 memakai `com.sun.net.httpserver.HttpServer` bawaan JDK. Program ini menyusun teks JSON secara manual, dengan indentasi yang sama seperti `json.dumps(..., indent=2)` di Python, sehingga kedua server mengembalikan isi yang sama.

Kode 9.6 `HealthServer.java`: *endpoint* yang sama dengan `com.sun.net.httpserver`.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;

/** A health-check endpoint: GET /api/health answers which version is
 *  running and whether its dependencies work.
 *  usage: java HealthServer [PORT] [APP_DIR] */
public class HealthServer {
    static final long STARTED = System.nanoTime();
    static final long MIN_FREE_BYTES = 1L << 30;     // our lab's threshold: 1 GiB
    static Path appDir = Path.of(".");
    static String version;       // read once at start, like the running code

    /** Can we write a file and read the same bytes back? */
    static String checkStore() {
        try {
            Path data = Files.createDirectories(appDir.resolve("_data"));
            Path probe = data.resolve("health.tmp");
            Files.writeString(probe, "ping");
            return Files.readString(probe).equals("ping") ? "ok" : "fail";
        } catch (IOException e) {
            return "fail";
        }
    }

    static String checkDisk() {
        return appDir.toFile().getUsableSpace() >= MIN_FREE_BYTES ? "ok" : "fail";
    }

    /** Is a local model server listening? 127.0.0.1, never "localhost". */
    static String checkAi() {
        try (Socket s = new Socket()) {
            s.connect(new InetSocketAddress("127.0.0.1", 11434), 300);
            return "ok";
        } catch (IOException e) {
            return "unavailable";
        }
    }
```

Kode 9.6 (Lanjutan) *HealthServer.java: endpoint yang sama dengan com.sun.net.httpserver.*

```
    static void handle(HttpExchange ex) throws IOException {
        if (!ex.getRequestURI().getPath().equals("/api/health")) {
            ex.sendResponseHeaders(404, -1);
            ex.close();
            return;
        }
        String store = checkStore(), disk = checkDisk(), ai = checkAi();
        String status;
        int code;
        if (store.equals("fail") || disk.equals("fail")) {
            status = "down"; code = 503;             // cannot serve users
        } else if (!ai.equals("ok")) {
            status = "degraded"; code = 200;         // serves, says what is missing
        } else {
            status = "ok"; code = 200;
        }
        long uptime = (System.nanoTime() - STARTED) / 1_000_000_000L;
        String json = "{\n"
                + "  \"status\": \"" + status + "\",\n"
                + "  \"version\": \"" + version + "\",\n"
                + "  \"checks\": {\n"
                + "    \"store\": \"" + store + "\",\n"
                + "    \"disk\": \"" + disk + "\",\n"
                + "    \"ai\": \"" + ai + "\"\n"
                + "  },\n"
                + "  \"uptime_s\": " + uptime + "\n"
                + "}\n";
        byte[] raw = json.getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", "application/json");
        ex.getResponseHeaders().set("Cache-Control", "no-store"); // never cache
        ex.sendResponseHeaders(code, raw.length);
        try (OutputStream out = ex.getResponseBody()) {
            out.write(raw);
        }
    }
```

Kode 9.6 (Lanjutan) *HealthServer.java: endpoint yang sama dengan com.sun.net.httpserver.*

```
    public static void main(String[] args) throws IOException {
        int port = args.length > 0 ? Integer.parseInt(args[0]) : 7090;
        if (args.length > 1) appDir = Path.of(args[1]);
        version = Files.readString(appDir.resolve("VERSION")).strip();
        HttpServer server = HttpServer.create(
                new InetSocketAddress("127.0.0.1", port), 0);
        server.createContext("/", HealthServer::handle);
        server.start();
        System.out.println("health on http://127.0.0.1:" + port + "/api/health");
    }
}
```

```
javac -d out HealthServer.java
java -cp out HealthServer 7090 .
```

```
HTTP/1.1 200 OK
Date: Thu, 24 Sep 2026 04:45:43 GMT
Content-type: application/json
Content-length: 146
Cache-control: no-store

{
  "status": "degraded",
  "version": "0.1.0",
  "checks": {
    "store": "ok",
    "disk": "ok",
    "ai": "unavailable"
  },
  "uptime_s": 1
}
```

Isinya identik; hanya *header*-nya yang berbeda, karena kedua server HTTP menuliskannya dengan cara berbeda (server Java hanya mengapitalkan huruf pertama setiap nama *header*, yang diperbolehkan HTTP karena nama *header* tidak peka huruf besar-kecil).

9.11 Praktikum G4, Bagian 3: Gerbang Rilis

Bagian ketiga menyatukan dua bagian pertama. Gerbang "men-*deploy*" penjadwal dan *endpoint* kesehatan ke sebuah folder yang berperan sebagai produksi, menjalankan server kesehatan dari sana, dan mengikuti enam langkah pada **Gambar 9.2**.

9.11.1 Bagaimana praktikum memetakan rilis sungguhan

Tabel 9.4 menunjukkan bagaimana setiap langkah ditirukan. Folder-foldernya berada di dalam direktori `_gate` di samping kode: `staging`, `prod`, dan `backup`.

Tabel 9.4 Bagaimana gerbang praktikum menirukan setiap langkah rilis sungguhan.

Langkah	Rilis sungguhan (Subbab 9.6.3)	Gerbang praktikum
1	Kunci *commit* dan *tree* yang bersih	Salin muatan ke `staging`, tulis `VERSION`, catat sidik jari SHA-256 berkas sumber
2	Cadangkan produksi	Salin `prod` ke `backup`
3	Suite SQLite + subset PostgreSQL	Jalankan uji penjadwal di dalam `staging`
4	Periksa ulang muatan	Hitung ulang sidik jari; tolak bila berubah
5	Salin dan hidupkan ulang layanan	Salin `staging` ke `prod`; hentikan server kesehatan lama, jalankan yang baru
6	`/api/health` menjawab versi baru	Tanyai `/api/health` hingga 5 detik; jika tidak, kembalikan dari `backup`

Dua *flag* memungkinkan kita sengaja membuat keadaan salah. `--inject-bug` menyelipkan regresi ke salinan penjadwal di *staging* setelah muatan dikunci: irama Pro diubah dari satu hari menjadi tujuh. `--skip-restart` menyalin berkas baru tetapi "lupa" menghidupkan ulang server, persis situasi gerbang yang berbohong di Subbab 9.6.1.

Jika pemeriksaan kesehatan diberi waktu hingga w detik dan bertanya setiap Δ detik, ia melakukan paling banyak

$$n_{\text{polls}} = \frac{w}{\Delta} = \frac{5\ \text{s}}{0.25\ \text{s}} = 20 \tag{9.8}$$

percobaan. Persamaan (9.8) adalah anggaran kesabaran: cukup lama untuk penyalaan ulang yang normal, cukup singkat agar kegagalan segera dinyatakan.

9.11.2 Gerbang Python

Kode 9.7 adalah gerbangnya. Program ini memakai `subprocess` untuk menjalankan uji dan menyalakan server, `hashlib` untuk sidik jari, dan `urllib` untuk bertanya kepada *endpoint* kesehatan.

Kode 9.7 `release_gate.py`: gerbang rilis enam langkah yang menolak bila uji gagal atau muatan berubah, dan menyatakan berhasil hanya ketika `/api/health` menjawab versi baru.

```
"""A release gate in six steps. It refuses to "deploy" when the tests
fail, and it reports success only when /api/health answers the new
version.
usage: python release_gate.py VERSION [--inject-bug] [--skip-restart]
       python release_gate.py --stop"""
import hashlib
import json
import os
import re
import shutil
import signal
import socket
import subprocess
import sys
import time
import urllib.request
from pathlib import Path

HERE = Path(__file__).resolve().parent
GATE = HERE / "_gate"
STAGING, PROD, BACKUP = GATE / "staging", GATE / "prod", GATE / "backup"
PIDFILE = GATE / "server.pid"
PAYLOAD = ["scheduler.py", "test_scheduler.py", "health.py"]
PORT = 7091
HEALTH = f"http://127.0.0.1:{PORT}/api/health"


def fingerprint(folder: Path) -> str:
    h = hashlib.sha256()
    for name in PAYLOAD:
        h.update(name.encode() + b"\0" + (folder / name).read_bytes())
    return h.hexdigest()[:12]


def copy_tree(src: Path, dst: Path) -> None:
    shutil.rmtree(dst, ignore_errors=True)
    shutil.copytree(src, dst, ignore=shutil.ignore_patterns("_data",
                                                            "__pycache__"))
```

Kode 9.7 (Lanjutan) *release_gate.py: gerbang rilis enam langkah yang menolak bila uji gagal atau muatan berubah, dan menyatakan berhasil hanya ketika /api/health menjawab versi baru.*

```
def port_open() -> bool:
    try:
        with socket.create_connection(("127.0.0.1", PORT), timeout=0.2):
            return True
    except OSError:
        return False


def stop_server() -> None:
    if PIDFILE.exists():
        try:
            os.kill(int(PIDFILE.read_text()), signal.SIGTERM)
        except OSError:
            pass                                    # already gone
        PIDFILE.unlink()
    for _ in range(50):                             # wait for the port
        if not port_open():
            return
        time.sleep(0.1)


def start_server() -> None:
    proc = subprocess.Popen([sys.executable, str(PROD / "health.py"),
                             str(PORT), str(PROD)],
                            stdout=subprocess.DEVNULL,
                            stderr=subprocess.DEVNULL,
                            start_new_session=True)  # outlives the gate
    PIDFILE.write_text(str(proc.pid))


def running_version(want: str | None, wait_s: float) -> str | None:
    """Poll /api/health until it answers `want` or the time is up."""
    deadline = time.monotonic() + wait_s
    seen = None
    while time.monotonic() < deadline:
        try:
            with urllib.request.urlopen(HEALTH, timeout=1) as r:
                seen = json.load(r)["version"]
        except (OSError, ValueError, KeyError):
            pass                                    # not up yet
        if seen == want:
            return seen
        time.sleep(0.25)
    return seen
```

Kode 9.7 (Lanjutan) *release_gate.py: gerbang rilis enam langkah yang menolak bila uji gagal atau muatan berubah, dan menyatakan berhasil hanya ketika /api/health menjawab versi baru.*

```
def run_tests(folder: Path) -> tuple[bool, str]:
    r = subprocess.run([sys.executable, "-m", "unittest", "test_scheduler"],
                       cwd=folder, capture_output=True, text=True)
    ran = re.search(r"Ran (\d+) test", r.stderr)
    failed = re.findall(r"^FAIL: (\w+)", r.stderr, re.M)
    summary = f"{ran.group(1) if ran else '?'} run, {len(failed)} failed"
    for name in failed:
        summary += f"\n        failed: {name}"
    return r.returncode == 0, summary


def gate(target: str, inject_bug: bool, skip_restart: bool) -> int:
    GATE.mkdir(exist_ok=True)
    print(f"[1/6] lock payload for {target}", end=": ")
    shutil.rmtree(STAGING, ignore_errors=True)
    STAGING.mkdir(parents=True)
    for name in PAYLOAD:
        shutil.copy2(HERE / name, STAGING / name)
    (STAGING / "VERSION").write_text(target + "\n")
    lock = fingerprint(HERE)
    print(lock)
    if inject_bug:                                  # a regression slips in
        f = STAGING / "scheduler.py"
        f.write_text(f.read_text().replace('"pro": timedelta(days=1)',
                                           '"pro": timedelta(days=7)'))

    print("[2/6] backup", end=": ")
    if (PROD / "VERSION").exists():
        old = (PROD / "VERSION").read_text().strip()
        copy_tree(PROD, BACKUP)
        print(f"saved production {old}")
    else:
        old = None
        print("nothing in production yet")

    ok, summary = run_tests(STAGING)
    print(f"[3/6] tests: {summary}")
    if not ok:
        print("REFUSED: tests failed; production was not touched")
        return 1
```

Kode 9.7 (Lanjutan) *release_gate.py: gerbang rilis enam langkah yang menolak bila uji gagal atau muatan berubah, dan menyatakan berhasil hanya ketika /api/health menjawab versi baru.*

```
    print("[4/6] re-check payload", end=": ")
    if fingerprint(HERE) != lock:
        print("changed during the gate")
        print("REFUSED: payload changed; run the gate again")
        return 1
    print("unchanged")

    print("[5/6] copy and restart")
    copy_tree(STAGING, PROD)
    if not skip_restart:
        stop_server()
        start_server()

    seen = running_version(target, wait_s=5)
    print(f"[6/6] health answers: {seen}")
    if seen == target:
        print(f"RELEASED {target}")
        return 0
    print(f"NOT RELEASED: expected {target}, rolling back")
    stop_server()
    if old is None:
        print("no earlier version to restore; server stopped")
        return 1
    copy_tree(BACKUP, PROD)
    start_server()
    print(f"health after rollback: {running_version(old, wait_s=5)}")
    return 1


if __name__ == "__main__":
    if sys.argv[1:] == ["--stop"]:
        stop_server()
        print("server stopped")
        sys.exit(0)
    sys.exit(gate(sys.argv[1], "--inject-bug" in sys.argv,
                  "--skip-restart" in sys.argv))
```

Beberapa keputusan layak dikomentari. Server produksi dijalankan dengan `start_new_session=True`, sehingga ia terus berjalan setelah gerbang selesai, seperti layanan sungguhan; nomor prosesnya disimpan dalam `server.pid` agar rilis berikutnya, atau `--stop`, dapat menghentikannya. `stop_server` menunggu sampai port benar-benar bebas sebelum server baru dijalankan, karena menjalankannya terlalu dini akan gagal mengikat port. `run_tests` membaca ringkasan yang ditulis `unittest` dan mengambil nama uji yang gagal, sehingga penolakan menyebut aturan *mana* yang patah. Dan `running_version` mengembalikan apa yang terakhir dilihatnya, meskipun bukan yang diinginkannya, sehingga laporan menyebut apa yang sebenarnya dijawab layanan.

Berikut satu sesi lengkap: rilis pertama, rilis dengan bug yang disisipkan, rilis yang penyalaan ulangnya dilewati, rilis yang baik, pemeriksaan *endpoint* kesehatan, dan penghentian.

```
$ python release_gate.py 0.1.0
[1/6] lock payload for 0.1.0: ec5b3806882b
[2/6] backup: nothing in production yet
[3/6] tests: 7 run, 0 failed
[4/6] re-check payload: unchanged
[5/6] copy and restart
[6/6] health answers: 0.1.0
RELEASED 0.1.0
$ python release_gate.py 0.2.0 --inject-bug
[1/6] lock payload for 0.2.0: ec5b3806882b
[2/6] backup: saved production 0.1.0
[3/6] tests: 7 run, 5 failed
        failed: test_cadence_matches_the_plans
        failed: test_late_worker_skips_instead_of_bursting
        failed: test_removed_monitor_never_runs_again
        failed: test_two_weeks_give_2_free_and_14_pro_scans
        failed: test_upgrade_brings_the_next_scan_forward
REFUSED: tests failed; production was not touched
$ python release_gate.py 0.2.0 --skip-restart
[1/6] lock payload for 0.2.0: ec5b3806882b
[2/6] backup: saved production 0.1.0
[3/6] tests: 7 run, 0 failed
[4/6] re-check payload: unchanged
[5/6] copy and restart
[6/6] health answers: 0.1.0
NOT RELEASED: expected 0.2.0, rolling back
health after rollback: 0.1.0
$ python release_gate.py 0.2.0
[1/6] lock payload for 0.2.0: ec5b3806882b
[2/6] backup: saved production 0.1.0
[3/6] tests: 7 run, 0 failed
[4/6] re-check payload: unchanged
[5/6] copy and restart
[6/6] health answers: 0.2.0
RELEASED 0.2.0
$ curl http://127.0.0.1:7091/api/health
{
  "status": "degraded",
  "version": "0.2.0",
  "checks": {
    "store": "ok",
    "disk": "ok",
    "ai": "unavailable"
  },
  "uptime_s": 0
}
$ python release_gate.py --stop
server stopped
```

Bacalah keempat jalannya satu per satu.

Rilis pertama tidak memiliki apa pun untuk dicadangkan, lolos tujuh ujinya, dan baru dinyatakan dirilis ketika *endpoint* kesehatan menjawab `0.1.0`.

Jalan kedua mengunci versi 0.2.0, mencadangkan produksi, lalu uji menangkap regresi yang disisipkan. Lima dari tujuh uji gagal, masing-masing menyebut sebuah aturan: irama, hitungan dua minggu, melompat alih-alih menyembur, peningkatan langganan, dan monitor yang dihapus, yang entri terakhirnya kini tertinggal di heap selama seminggu. Gerbang menolak. Produksi tetap menjalankan 0.1.0, tak tersentuh.

Jalan ketiga adalah gerbang yang berbohong, yang kini tertangkap. Uji lolos, berkas disalin, tetapi tidak ada penyalaan ulang. Proses lama, yang membaca versinya ketika dimulai, masih menjawab

`0.1.0`. Gerbang tidak mengaku berhasil; ia memulihkan cadangan dan menyalakan ulang, dan layanan kembali menjawab `0.1.0`. Tanpa langkah 6, jalan ini akan mencetak "RELEASED 0.2.0".

Jalan keempat mengerjakan semuanya dengan benar, dan baru pada saat itu menyatakan `RELEASED 0.2.0`. Perintah `curl` sesudahnya mengonfirmasi hal itu secara independen.

Praktikum ini sengaja menyederhanakan dua hal yang dihadapi rilis sungguhan. Pertama, perubahan skema basis data: bila versi baru menambah kolom, cadangan kode saja tidak cukup untuk mundur, sehingga perubahan skema lazimnya dirancang agar versi lama tetap dapat bekerja dengan skema baru. Kedua, jeda saat penyalaan ulang: selama beberapa detik antara berhentinya proses lama dan siapnya proses baru, permintaan yang masuk akan gagal. Untuk sistem kecil, jeda itu diterima dan dijadwalkan pada jam sepi; sistem yang lebih besar menutupnya dengan menjalankan versi baru di samping versi lama lalu mengalihkan lalu lintas. Kedua pokok ini layak menjadi bahan eksplorasi lanjutan setelah gerbang dasar dikuasai.

9.11.3 Gerbang Java

Kode 9.8 adalah gerbang yang sama dalam Java. Program ini mengompilasi sumber di *staging* dengan `javac` di dalam folder *staging*, sehingga muatan yang tidak dapat dikompilasi gagal di langkah 3 seperti uji yang gagal. Program ini dapat dijalankan tanpa dikompilasi lebih dulu, memakai peluncur sumber berkas tunggal: `java ReleaseGate.java 0.1.0`.

Kode 9.8 `ReleaseGate.java`: gerbang yang sama dalam Java, dengan `ProcessBuilder`, `MessageDigest`, dan `java.net.http.HttpClient`.

```
import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.StandardCopyOption;
import java.security.MessageDigest;
import java.time.Duration;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HexFormat;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.stream.Stream;

/** A release gate in six steps. It refuses to "deploy" when the tests
 *  fail, and it reports success only when /api/health answers the new
 *  version.
 *  usage: java ReleaseGate.java VERSION [--inject-bug] [--skip-restart]
 *         java ReleaseGate.java --stop */
public class ReleaseGate {
    static final Path HERE = Path.of("").toAbsolutePath();
    static final Path GATE = HERE.resolve("_gate");
    static final Path STAGING = GATE.resolve("staging");
    static final Path PROD = GATE.resolve("prod");
    static final Path BACKUP = GATE.resolve("backup");
    static final Path PIDFILE = GATE.resolve("server.pid");
    static final List<String> PAYLOAD = List.of("Scheduler.java",
            "TestScheduler.java", "MiniTest.java", "HealthServer.java");
    static final int PORT = 7092;
    static final String HEALTH = "http://127.0.0.1:" + PORT + "/api/health";
    static final String JAVA = ProcessHandle.current().info().command()
            .orElse("java");
```

Kode 9.8 (Lanjutan) *ReleaseGate.java: gerbang yang sama dalam Java, dengan ProcessBuilder, MessageDigest, dan java.net.http.HttpClient.*

```
    static String fingerprint(Path folder) throws Exception {
        MessageDigest h = MessageDigest.getInstance("SHA-256");
        for (String name : PAYLOAD) {
            h.update((name + "\0").getBytes(StandardCharsets.UTF_8));
            h.update(Files.readAllBytes(folder.resolve(name)));
        }
        return HexFormat.of().formatHex(h.digest()).substring(0, 12);
    }

    static void delete(Path dir) throws IOException {
        if (!Files.exists(dir)) return;
        try (Stream<Path> s = Files.walk(dir)) {
            for (Path p : s.sorted(Comparator.reverseOrder()).toList())
                Files.delete(p);
        }
    }

    static void copyTree(Path src, Path dst) throws IOException {
        delete(dst);
        try (Stream<Path> s = Files.walk(src)) {
            for (Path p : s.toList()) {
                if (p.toString().contains("_data")) continue;
                Path q = dst.resolve(src.relativize(p).toString());
                if (Files.isDirectory(p)) Files.createDirectories(q);
                else Files.copy(p, q, StandardCopyOption.REPLACE_EXISTING);
            }
        }
    }

    static boolean portOpen() {
        try (Socket s = new Socket()) {
            s.connect(new InetSocketAddress("127.0.0.1", PORT), 200);
            return true;
        } catch (IOException e) {
            return false;
        }
    }
```

Kode 9.8 (Lanjutan) *ReleaseGate.java: gerbang yang sama dalam Java, dengan ProcessBuilder, MessageDigest, dan java.net.http.HttpClient.*

```
    static void stopServer() throws Exception {
        if (Files.exists(PIDFILE)) {
            long pid = Long.parseLong(Files.readString(PIDFILE).strip());
            ProcessHandle.of(pid).ifPresent(ProcessHandle::destroy);
            Files.delete(PIDFILE);
        }
        for (int i = 0; i < 50 && portOpen(); i++) Thread.sleep(100);
    }

    static void startServer() throws IOException {
        Process p = new ProcessBuilder(JAVA, "-cp",
                PROD.resolve("classes").toString(), "HealthServer",
                String.valueOf(PORT), PROD.toString())
                .redirectOutput(ProcessBuilder.Redirect.DISCARD)
                .redirectError(ProcessBuilder.Redirect.DISCARD)
                .start();                              // outlives the gate
        Files.writeString(PIDFILE, String.valueOf(p.pid()));
    }

    /** Poll /api/health until it answers want or the time is up. */
    static String runningVersion(String want, double waitS) throws Exception {
        HttpClient client = HttpClient.newBuilder()
                .connectTimeout(Duration.ofSeconds(1)).build();
        HttpRequest req = HttpRequest.newBuilder(URI.create(HEALTH))
                .timeout(Duration.ofSeconds(1)).build();
        Pattern v = Pattern.compile("\"version\": \"([^\"]+)\"");
        long deadline = System.nanoTime() + (long) (waitS * 1e9);
        String seen = null;
        while (System.nanoTime() < deadline) {
            try {
                String body = client.send(req,
                        HttpResponse.BodyHandlers.ofString()).body();
                Matcher m = v.matcher(body);
                if (m.find()) seen = m.group(1);
            } catch (IOException e) {
                // not up yet
            }
            if (seen != null && seen.equals(want)) return seen;
            Thread.sleep(250);
        }
        return seen;
    }
```

Kode 9.8 (Lanjutan) *ReleaseGate.java: gerbang yang sama dalam Java, dengan ProcessBuilder, MessageDigest, dan java.net.http.HttpClient.*

```
    /** Compile and run the tests in a folder; returns {passed, summary}. */
    static Object[] runTests(Path folder) throws Exception {
        List<String> cmd = new ArrayList<>(List.of("javac", "-d", "classes"));
        cmd.addAll(PAYLOAD);
        Process javac = new ProcessBuilder(cmd).directory(folder.toFile())
                .redirectErrorStream(true).start();
        javac.getInputStream().readAllBytes();
        if (javac.waitFor() != 0) return new Object[] {false, "did not compile"};
        Process run = new ProcessBuilder(JAVA, "-cp", "classes", "TestScheduler")
                .directory(folder.toFile()).start();
        String out = new String(run.getInputStream().readAllBytes(),
                StandardCharsets.UTF_8);
        run.getErrorStream().readAllBytes();
        boolean ok = run.waitFor() == 0;
        Matcher m = Pattern.compile("Ran (\\d+) tests: \\d+ passed, (\\d+) failed")
                .matcher(out);
        StringBuilder summary = new StringBuilder(
                m.find() ? m.group(1) + " run, " + m.group(2) + " failed" : "?");
        Matcher f = Pattern.compile("(?m)^FAIL  ([^:]+):").matcher(out);
        while (f.find()) summary.append("\n        failed: ").append(f.group(1));
        return new Object[] {ok, summary.toString()};
    }

    static int gate(String target, boolean injectBug, boolean skipRestart)
            throws Exception {
        Files.createDirectories(GATE);
        System.out.print("[1/6] lock payload for " + target + ": ");
        delete(STAGING);
        Files.createDirectories(STAGING);
        for (String name : PAYLOAD)
            Files.copy(HERE.resolve(name), STAGING.resolve(name));
        Files.writeString(STAGING.resolve("VERSION"), target + "\n");
        String lock = fingerprint(HERE);
        System.out.println(lock);
        if (injectBug) {                                  // a regression slips in
            Path f = STAGING.resolve("Scheduler.java");
            Files.writeString(f, Files.readString(f).replace(
                    "\"pro\", Duration.ofDays(1)", "\"pro\", Duration.ofDays(7)"));
        }
```

Kode 9.8 (Lanjutan) *ReleaseGate.java: gerbang yang sama dalam Java, dengan ProcessBuilder, MessageDigest, dan java.net.http.HttpClient.*

```
        System.out.print("[2/6] backup: ");
        String old = null;
        if (Files.exists(PROD.resolve("VERSION"))) {
            old = Files.readString(PROD.resolve("VERSION")).strip();
            copyTree(PROD, BACKUP);
            System.out.println("saved production " + old);
        } else {
            System.out.println("nothing in production yet");
        }

        Object[] tests = runTests(STAGING);
        System.out.println("[3/6] tests: " + tests[1]);
        if (!(Boolean) tests[0]) {
            System.out.println("REFUSED: tests failed; production was not touched");
            return 1;
        }

        System.out.print("[4/6] re-check payload: ");
        if (!fingerprint(HERE).equals(lock)) {
            System.out.println("changed during the gate");
            System.out.println("REFUSED: payload changed; run the gate again");
            return 1;
        }
        System.out.println("unchanged");

        System.out.println("[5/6] copy and restart");
        copyTree(STAGING, PROD);
        if (!skipRestart) {
            stopServer();
            startServer();
        }

        String seen = runningVersion(target, 5);
        System.out.println("[6/6] health answers: " + seen);
        if (target.equals(seen)) {
            System.out.println("RELEASED " + target);
            return 0;
        }
        System.out.println("NOT RELEASED: expected " + target + ", rolling back");
        stopServer();
        if (old == null) {
            System.out.println("no earlier version to restore; server stopped");
            return 1;
        }
        copyTree(BACKUP, PROD);
        startServer();
```

Kode 9.8 (Lanjutan) *ReleaseGate.java: gerbang yang sama dalam Java, dengan ProcessBuilder, MessageDigest, dan java.net.http.HttpClient.*

```
        System.out.println("health after rollback: " + runningVersion(old, 5));
        return 1;
    }

    public static void main(String[] args) throws Exception {
        if (args.length == 1 && args[0].equals("--stop")) {
            stopServer();
            System.out.println("server stopped");
            return;
        }
        List<String> a = List.of(args);
        System.exit(gate(args[0], a.contains("--inject-bug"),
                         a.contains("--skip-restart")));
    }
}
```

```
$ java ReleaseGate.java 0.1.0
[1/6] lock payload for 0.1.0: d085bacb2b69
[2/6] backup: nothing in production yet
[3/6] tests: 7 run, 0 failed
[4/6] re-check payload: unchanged
[5/6] copy and restart
[6/6] health answers: 0.1.0
RELEASED 0.1.0
$ java ReleaseGate.java 0.2.0 --inject-bug
[1/6] lock payload for 0.2.0: d085bacb2b69
[2/6] backup: saved production 0.1.0
[3/6] tests: 7 run, 5 failed
        failed: cadence matches the plans
        failed: two weeks give 2 Free and 14 Pro scans
        failed: late worker skips instead of bursting
        failed: upgrade brings the next scan forward
        failed: removed monitor never runs again
REFUSED: tests failed; production was not touched
$ java ReleaseGate.java 0.2.0 --skip-restart
[1/6] lock payload for 0.2.0: d085bacb2b69
[2/6] backup: saved production 0.1.0
[3/6] tests: 7 run, 0 failed
[4/6] re-check payload: unchanged
[5/6] copy and restart
[6/6] health answers: 0.1.0
NOT RELEASED: expected 0.2.0, rolling back
health after rollback: 0.1.0
$ java ReleaseGate.java 0.2.0
[1/6] lock payload for 0.2.0: d085bacb2b69
[2/6] backup: saved production 0.1.0
[3/6] tests: 7 run, 0 failed
[4/6] re-check payload: unchanged
[5/6] copy and restart
[6/6] health answers: 0.2.0
RELEASED 0.2.0
$ java ReleaseGate.java --stop
server stopped
```

Ceritanya sama, baris demi baris. Sidik jarinya berbeda dari Python karena dihitung atas berkas yang berbeda, dan uji yang gagal didaftar menurut urutan eksekusi pelari Java. Gerbang memakai port 7092 agar gerbang Python dan Java dapat berjalan berdampingan tanpa bentrok.

> **PERINGATAN:** Selalu akhiri dengan `--stop`. Salinan produksi server kesehatan memang dirancang untuk hidup lebih lama daripada gerbang, seperti layanan sungguhan; jika tidak dihentikan, ia terus berjalan di latar belakang sampai Anda menutup sesi atau menyalakan ulang komputer.

9.12 Profitina dalam Praktik

Pada rilis September 2026, gagasan-gagasan bab ini muncul dalam produk sebagai berikut.

- **Arsitektur.** Satu mesin Windows 11 native; FastAPI di Uvicorn sebagai layanan Windows di bawah NSSM; PostgreSQL dengan basis data pengembangan yang diisolasi melalui izin peran; Server-Sent Events untuk waktu nyata; Ollama dengan jalur mutu dan jalur cepat, dipanggil di `127.0.0.1`; situs pemasaran WordPress di samping aplikasi; semuanya dijangkau melalui tepi Cloudflare dan terowongan yang hanya mengarah keluar.
- **Sengaja tidak ada.** Tanpa penyedia awan raksasa, tanpa Kubernetes, tanpa *microservices*, tanpa penskalaan otomatis; skala regional adalah fase peta jalan berikutnya.
- **Gerbang rilis.** Suite SQLite penuh (1.577 lolos, 1 dilewati pada saat tulisan ini dibuat) dan subset prioritas PostgreSQL 16 (440 lolos) harus hijau, dengan peringatan diperlakukan sebagai galat, disertai penjaga UI JavaScript dan pemeriksaan peramban.
- **Rilis yang diverifikasi kesehatannya.** Enam langkah, berakhir dengan keberhasilan hanya ketika `/api/health` menjawab versi baru; rilis menjalankan salinan skripnya.
- **Observabilitas.** Galat server dicatat tanpa isi maupun *header* dan dengan alamat di-*hash*; peringatan lonjakan; *endpoint* kesehatan yang melaporkan basis data, AI, disk, lama menyala, dan galat terbaru; log galat dibersihkan setelah 90 hari.
- **Kinerja.** Feed untuk 30 unggahan dalam 11 milidetik setelah perbaikan September 2026, dengan 12 kueri, tanpa koneksi baru, dan tanpa panggilan model, dijaga oleh aturan bahwa fungsi tampilan per butir tidak pernah menyentuh model atau membuka koneksi.
- **Irama pemantauan.** Monitor Free dipindai ulang setiap minggu, monitor Pro setiap hari, sesuai tabel paket di Bab 8.

Praktikum G4 tidak mereproduksi kode Profitina sama sekali. Praktikum ini mengajarkan tiga gagasan operasional dengan program kecil pada halaman-halaman Profitina Laundry, tempat yang tepat untuk praktik langsung dalam seri buku ini.

9.13 Kesalahpahaman yang Sering Terjadi

KESALAHPAHAMAN 1: "Produk yang serius memerlukan banyak server." Produk memerlukan keandalan yang dibutuhkan penggunanya. Sebagian besar keandalan itu berasal dari pemulihan yang cepat (Persamaan (9.1)), cadangan, uji, dan rilis yang cermat, dan semuanya bekerja di satu mesin. Mesin tambahan menambah kapasitas sekaligus modus kegagalan baru.

KESALAHPAHAMAN 2: "Uji sudah lolos, jadi rilis berhasil." Uji menunjukkan kode bekerja di lingkungan uji. Hanya layanan yang sedang berjalan yang dapat menunjukkan bahwa versi baru sudah aktif, dan karena itulah rilis bertanya kepada `/api/health`.

KESALAHPAHAMAN 3: "Pemeriksaan kesehatan cukup menyatakan OK bila prosesnya hidup." Pemeriksaan kesehatan yang berguna melaporkan apakah layanan dapat menjalankan tugasnya, membedakan "mati" dari "menurun", dan menyebut versi yang sedang dijalankan.

KESALAHPAHAMAN 4: "Feed lambat karena mesinnya kecil." Feed lambat karena tiga pola perangkat lunak: panggilan model di jalur baca, satu koneksi per panggilan, dan kueri N+1. Mesin yang sama kini menyajikan halaman itu dalam 11 milidetik.

KESALAHPAHAMAN 5: "`localhost` dan `127.0.0.1` adalah hal yang sama." Sering memang sama, tetapi `localhost` dapat lebih dulu diterjemahkan ke IPv6. Terhadap server yang hanya

mendengarkan IPv4, hal itu dapat memakan beberapa detik per panggilan; terukur pada produk, 2,25 detik dibanding 0,17.

KESALAHPAHAMAN 6: "Penjadwal yang sempat mati harus mengejar dengan menjalankan setiap tugas yang terlewat." Memutar ulang pindaian berkala yang terlewat menghasilkan semburan pekerjaan identik pada saat terburuk. Jalankan sekali, lompati slot yang terlewat (Persamaan (9.7)), dan kembali ke kisi.

KESALAHPAHAMAN 7: "Mencetak nol ketika kueri gagal adalah bawaan yang aman." Jawaban salah yang tampak normal tidak dapat disadari. Galat harus lantang.

9.14 Ringkasan Bab

- Ketersediaan adalah $A = \text{MTBF}/(\text{MTBF} + \text{MTTR})$ (Persamaan (9.1)); pemulihan yang lebih cepat sering menaikkannya lebih murah daripada kegagalan yang lebih jarang. Waktu mati yang diperbolehkan pada setiap target mengikuti Persamaan (9.2) dan **Tabel 9.1**.
- Arsitektur Profitina (**Gambar 9.1**, **Tabel 9.2**) adalah satu host Windows 11 native dengan FastAPI di Uvicorn di bawah NSSM, PostgreSQL dengan data pengembangan yang diisolasi melalui peran, Server-Sent Events, dua jalur AI lokal melalui Ollama, dan situs pemasaran WordPress, semuanya dijangkau melalui tepi Cloudflare dan terowongan yang hanya mengarah keluar.
- Satu mesin adalah pilihan yang sesuai dengan tahapannya: kebanyakan gangguan pada tim kecil berasal dari perubahan, dan kebanyakan keandalan berasal dari proses.
- Gerbang rilis harus menghentikan rilis ketika gagal. Suite Profitina (**Tabel 9.3**) memiliki 1.577 uji yang lolos pada SQLite dan 440 pada PostgreSQL 16 pada saat tulisan ini dibuat, dengan peringatan sebagai galat; suite itu tumbuh dari insiden seperti jebakan dialek SQL dan nol yang diam.
- Rilis enam langkah (**Gambar 9.2**) mengunci, mencadangkan, menguji, memeriksa ulang, menyalin dan menyalakan ulang, lalu menyatakan berhasil hanya ketika `/api/health` menjawab versi baru; ini menutup celah yang terungkap ketika pesan berhasil sebuah *batch script* berbohong.
- Perpindahan kanal turun dari 868 menjadi 11 milidetik (**Gambar 9.3**, Persamaan (9.4)) dengan menghapus panggilan model dari jalur baca, memakai *pool* koneksi, dan mengganti kueri N+1 (Persamaan (9.3)) dengan pengisian yang diambil di muka; penjaga nomor urut memperbaiki balapan respons basi (**Gambar 9.4**).
- Penjadwal adalah min-heap waktu jatuh tempo (**Gambar 9.5**); operasinya berbiaya $O(\log n)$ (Persamaan (9.5)), banyaknya jalan mengikuti Persamaan (9.6), pekerja yang terlambat melompat (Persamaan (9.7)), dan entri basi dibuang secara malas.
- Praktikum G4 membangun penjadwal, *endpoint* kesehatan yang membedakan mati dari menurun, dan gerbang rilis yang menolak uji yang gagal dan mengembalikan versi yang tidak menjawab, dengan paling banyak Persamaan (9.8) kali bertanya.

9.15 Soal Latihan

Soal 9.1. Sebuah layanan gagal rata-rata sekali setiap 14 hari dan memerlukan 6 menit untuk dipulihkan. (a) Hitung ketersediaannya dengan Persamaan (9.1). (b) Berapa lama rerata waktu pemulihan harus dicapai agar ketersediaannya 99,99% dengan laju kegagalan yang sama? (c) Dengan memakai **Tabel 9.1**, perubahan mana yang lebih realistis bagi tim kecil: membagi dua laju kegagalan, atau memangkas waktu pemulihan menjadi satu menit?

Soal 9.2. Sebuah halaman menampilkan $N = 50$ unggahan, dan kode naif menjalankan $k = 6$ kueri tambahan per unggahan. (a) Berapa kueri yang diramalkan Persamaan (9.3)? (b) Jika setiap kueri memakan 1,5 milidetik dan koneksi baru menambah 4 milidetik per kueri, perkirakan waktunya dengan dan tanpa *pool* koneksi. (c) Dengan pengisian yang diambil di muka, halaman memerlukan satu kueri untuk daftar dan satu per jenis data (pembuat unggahan, kanal, hitungan): berapa kueri itu, dan mengapa jumlahnya tidak lagi bergantung pada N?

Soal 9.3. Mulai dari min-heap kosong, sisipkan waktu jatuh tempo 9, 4, 7, 1, 8, 2 (dalam jam) dengan urutan itu, gambarkan lariknya setelah setiap penyisipan. Lalu keluarkan nilai minimum dua kali, gambarkan lariknya setelah setiap pengeluaran. Berapa pertukaran unsur yang diperlukan setiap operasi, dan bagaimana perbandingannya dengan $\log_2 n$?

Soal 9.4. Perluas penjadwal dengan *tren mingguan*: jalankan satu monitor Pro selama 21 hari simulasi, kelompokkan skornya menurut minggu ISO, cetak rata-rata setiap minggu, dan cetak peringatan bila perubahan rata-rata dari minggu ke minggu melebihi $\delta = 5$ poin yang dipilih. Tulis dalam Python dan Java.

Soal 9.5. Bagaimana Anda menguji bahwa `health.py` mengembalikan HTTP 503 dengan status `down` ketika ruang disk kosong di bawah ambang, tanpa memenuhi disk Anda? Jelaskan perubahan pada program dan ujinya.

Soal 9.6. Langkah 4 gerbang menolak bila muatan berubah selama uji, tetapi tidak satu pun jalan peragaan memicunya. Jelaskan cara memicunya dengan sengaja, dan ramalkan keluaran gerbang.

Soal 9.7. Tulis kelas kecil `LatestOnly` (Python dan Java) yang mengimplementasikan penjaga nomor urut pada **Gambar 9.4**: `request()` mengembalikan nomor urut baru, dan `reply(seq, payload)` menampilkan isi hanya bila `seq` adalah yang terbaru. Peragakan bahwa balasan terlambat untuk kanal A dibuang setelah pengguna berpindah ke kanal B.

Soal 9.8. Jelaskan, berdasarkan apa yang *diamati* setiap langkah, mengapa rilis di Subbab 9.6.1 dapat melaporkan keberhasilan untuk versi yang tidak sedang berjalan, dan mengapa langkah 6 pada rilis yang baru tidak dapat melakukannya.

Soal 9.9. Sebutkan empat informasi yang aman dilaporkan *endpoint* kesehatan publik dan empat yang tidak boleh dilaporkannya. Beri alasan setiap butir dalam satu kalimat.

Soal 9.10. Sebagian platform membedakan pemeriksaan *liveness* (“apakah prosesnya hidup?”) dari pemeriksaan *readiness* (“apakah ia dapat melayani permintaan sekarang?”). Status mana dari ketiga status `health.py` yang sesuai dengan jawaban mana, dan apa yang semestinya dilakukan pengelola layanan seperti NSSM pada masing-masing?

9.16 Penyelesaian Soal Latihan

Penyelesaian 9.1. (a) MTBF adalah $14 \times 24 \times 60 = 20{,}160$ menit, MTTR 6 menit, sehingga $A = 20{,}160/20{,}166 \approx 0.99970$, yaitu 99,970%. (b) Diperlukan $\text{MTBF}/(\text{MTBF} + \text{MTTR}) \geq 0.9999$, sehingga $\text{MTTR} \leq \text{MTBF} \times (1/0.9999 - 1) \approx 20{,}160 \times 0.0001 \approx 2.0$ menit. (c) Memangkas waktu pemulihan menjadi satu menit memberikan $A = 20{,}160/20{,}161 \approx 99.995\%$, sudah melampaui empat sembilan, dan biasanya hanya memerlukan otomatisasi (pengelola layanan, pengembalian yang diskripkan). Membagi dua laju kegagalan akan memberikan $40{,}320/40{,}326 \approx 99.985\%$ dan biasanya menuntut kerja mendalam pada penyebabnya. Bagi tim kecil, pemulihan yang lebih cepat adalah tuas yang lebih realistis, dan itulah pelajaran Persamaan (9.1).

Penyelesaian 9.2. (a) $Q(50) = 1 + 6 \times 50 = 301$ kueri. (b) Tanpa *pool*, setiap kueri berbiaya $1.5 + 4 = 5.5$ ms, jadi sekitar $301 \times 5.5 \approx 1{,}656$ ms; dengan *pool*, sekitar $301 \times 1.5 \approx 452$ ms. *Pool* membantu, tetapi halaman masih lambat, karena masalah sebenarnya adalah jumlah kueri. (c) Satu kueri untuk daftar ditambah masing-masing satu untuk pembuat unggahan, kanal, dan hitungan adalah 4 kueri, sekitar 6 ms dengan 1,5 ms per kueri. Setiap kueri pengambilan di muka meminta semua baris yang diperlukan sekaligus (misalnya semua pembuat unggahan yang identitasnya muncul dalam daftar), sehingga jumlah kueri bergantung pada banyaknya *jenis* data, bukan banyaknya unggahan.

Penyelesaian 9.3. Dengan menuliskan heap sebagai larik (indeks 0 adalah akar; anak dari indeks i adalah $2i + 1$ dan $2i + 2$), setiap penyisipan menaruh nilai baru di ujung lalu membiarkannya naik selama lebih kecil daripada induknya; setiap pengeluaran memindahkan nilai terakhir ke akar lalu membiarkannya turun ke arah anak yang lebih kecil:

```
insert 9   [9]                0 swaps
insert 4   [4, 9]             1 swap   (4 < 9)
insert 7   [4, 9, 7]          0 swaps  (7 > 4)
insert 1   [1, 4, 7, 9]       2 swaps  (1 < 9, then 1 < 4)
insert 8   [1, 4, 7, 9, 8]    0 swaps  (8 > 4)
insert 2   [1, 4, 2, 9, 8, 7] 1 swap   (2 < 7, then 2 > 1)
pop -> 1   [2, 4, 7, 9, 8]    1 swap   (7 moved to root, 7 > 2)
pop -> 2   [4, 8, 7, 9]       1 swap   (8 moved to root, 8 > 4, 8 < 9)
```

Tidak ada operasi yang memerlukan lebih dari 2 pertukaran. Heap tidak pernah berisi lebih dari 6 entri, dan pohon biner dengan 6 simpul memiliki tinggi $\lfloor \log_2 6 \rfloor = 2$, sehingga tidak ada operasi yang dapat memerlukan lebih dari 2 pertukaran: itulah biaya $O(\log n)$ di balik Persamaan (9.5).

Penyelesaian 9.4. **Kode 9.9** dan **Kode 9.10** menjalankan penjadwal selama 21 hari dan mengelompokkan hasilnya dengan nomor minggu ISO (`isocalendar().week` di Python, `IsoFields.WEEK_OF_WEEK_BASED_YEAR` di Java).

Kode 9.9 `weekly_trend.py`: rata-rata per minggu ISO dari skor satu monitor, dengan peringatan perubahan antarminggu.

```
"""Exercise 9.4: weekly trend per monitor (ISO weeks) with an alert when
the week-over-week change is larger than a chosen delta."""
from collections import defaultdict
from datetime import timedelta

from scheduler import Monitor, Scheduler, at

DELTA = 5.0                                  # the reader's choice, in points

s = Scheduler()
start = at("2026-09-21 09:00")
s.add(Monitor("laundry-prices", "pro"), start)
for h in range(0, 21 * 24, 3):
    s.run_due(start + timedelta(hours=h))

weeks: dict[tuple[str, int], list[int]] = defaultdict(list)
for mid, when, score in s.results:
    weeks[(mid, when.isocalendar().week)].append(score)

previous = None
for (mid, week), scores in sorted(weeks.items()):
    mean = sum(scores) / len(scores)
    line = f"{mid}  ISO week {week}: {len(scores)} scans, mean {mean:.1f}"
    if previous is not None and abs(mean - previous) > DELTA:
        line += f"  ALERT: changed by {mean - previous:+.1f}"
    print(line)
    previous = mean
```

Kode 9.10 `WeeklyTrend.java`: tren yang sama dalam Java.

```
import java.time.ZonedDateTime;
import java.time.temporal.IsoFields;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

/** Exercise 9.4: weekly trend per monitor (ISO weeks) with an alert when
 *  the week-over-week change is larger than a chosen delta. */
public class WeeklyTrend {
    static final double DELTA = 5.0;         // the reader's choice, in points

    public static void main(String[] args) {
        Scheduler s = new Scheduler();
        ZonedDateTime start = Scheduler.at("2026-09-21 09:00");
        s.add(new Scheduler.Monitor("laundry-prices", "pro"), start);
        for (int h = 0; h < 21 * 24; h += 3) s.runDue(start.plusHours(h));

        Map<Integer, List<Integer>> weeks = new TreeMap<>();
        for (Scheduler.Result r : s.results)
            weeks.computeIfAbsent(r.when().get(IsoFields.WEEK_OF_WEEK_BASED_YEAR),
                    k -> new ArrayList<>()).add(r.score());

        Double previous = null;
        for (Map.Entry<Integer, List<Integer>> e : weeks.entrySet()) {
            List<Integer> scores = e.getValue();
            double mean = scores.stream().mapToInt(Integer::intValue).average()
                    .orElse(0);
            String line = String.format("laundry-prices  ISO week %d: %d scans, "
                    + "mean %.1f", e.getKey(), scores.size(), mean);
            if (previous != null && Math.abs(mean - previous) > DELTA)
                line += String.format("  ALERT: changed by %+.1f", mean - previous);
            System.out.println(line);
            previous = mean;
        }
    }
}
```

Keduanya mencetak:

```
laundry-prices  ISO week 39: 7 scans, mean 76.7
laundry-prices  ISO week 40: 7 scans, mean 59.0  ALERT: changed by -17.7
laundry-prices  ISO week 41: 7 scans, mean 70.9  ALERT: changed by +11.9
```

Peringatan menyala karena `stub_scan` menghasilkan derau, bukan karena ada yang berubah pada halaman; pemindai sungguhan akan bergeser jauh lebih sedikit dari hari ke hari. Soal ini tentang mekanismenya: mengelompokkan menurut minggu ISO (yang dimulai pada hari Senin, hari yang sama dengan awal simulasi), merata-ratakan, dan membandingkan dengan ambang yang dipilih pembaca. Memilih δ adalah keputusan produk: terlalu kecil dan pengguna belajar mengabaikan peringatan, terlalu besar dan perubahan nyata luput.

Penyelesaian 9.5. Jadikan ambangnya dapat diatur dari luar, misalnya dengan membaca variabel lingkungan `HEALTH_MIN_FREE_BYTES` dengan nilai bawaan praktikum sebagai cadangan. Sebuah uji lalu menjalankan server dengan variabel itu diisi angka yang lebih besar daripada disk mana pun (misalnya 10^{18} byte), meminta `/api/health`, dan memeriksa status 503 serta `"status": "down"`. Pendekatan yang sama menguji pemeriksaan penyimpanan, dengan mengarahkan `APP_DIR` ke folder yang tidak dapat ditulisi. Pelajaran umumnya: rancang pemeriksaan sehingga kegagalannya dapat *ditimbulkan* dalam uji, karena jika tidak, jalur kegagalan itu tidak akan pernah dijalankan sampai saatnya benar-benar penting.

Penyelesaian 9.6. Langkah 4 membandingkan sidik jari berkas sumber sebelum dan sesudah uji. Untuk memicunya, ubah sebuah berkas muatan selama uji berjalan: misalnya, tambahkan baris di `test_scheduler.py` yang tidur selama sepuluh detik, jalankan gerbang, dan selama jeda itu sunting sebuah komentar di `scheduler.py` pada folder sumber. Keluarannya akan menunjukkan langkah 1 sampai 3 seperti biasa, lalu `[4/6] re-check payload: changed during the gate` dan `REFUSED: payload changed; run the gate again`, dengan produksi tak tersentuh. Uji memang lolos, tetapi yang diuji adalah teks lama; gerbang menolak menyalin kode yang tidak diujinya.

Penyelesaian 9.7. **Kode 9.11** dan **Kode 9.12** menyimpan penghitung permintaan terbaru dan membandingkannya dengan nomor yang dibawa setiap balasan.

Kode 9.11 `latest_only.py`: penjaga nomor urut terhadap balasan basi.

```
"""Exercise 9.7: drop stale replies with a request sequence number."""


class LatestOnly:
    def __init__(self):
        self.latest = 0
        self.shown: list[str] = []

    def request(self) -> int:
        self.latest += 1                # every new request gets a number
        return self.latest

    def reply(self, seq: int, payload: str) -> bool:
        if seq != self.latest:
            return False                # stale: a newer request exists
        self.shown.append(payload)
        return True


panel = LatestOnly()
a = panel.request()                     # user opens channel A
b = panel.request()                     # ...and quickly switches to B
print("reply for B shown:", panel.reply(b, "posts of B"))
print("late reply for A shown:", panel.reply(a, "posts of A"))
print("panel shows:", panel.shown)
```

Kode 9.12 `LatestOnly.java`: penjaga yang sama dalam Java.

```java
import java.util.ArrayList;
import java.util.List;

/** Exercise 9.7: drop stale replies with a request sequence number. */
public class LatestOnly {
    int latest = 0;
    final List<String> shown = new ArrayList<>();

    int request() {
        return ++latest;                // every new request gets a number
    }

    boolean reply(int seq, String payload) {
        if (seq != latest) return false; // stale: a newer request exists
        shown.add(payload);
        return true;
    }

    public static void main(String[] args) {
        LatestOnly panel = new LatestOnly();
        int a = panel.request();        // user opens channel A
        int b = panel.request();        // ...and quickly switches to B
        System.out.println("reply for B shown: "
                + (panel.reply(b, "posts of B") ? "True" : "False"));
        System.out.println("late reply for A shown: "
                + (panel.reply(a, "posts of A") ? "True" : "False"));
        System.out.println("panel shows: ['" + String.join("', '", panel.shown)
                + "']");
    }
}
```

```
reply for B shown: True
late reply for A shown: False
panel shows: ['posts of B']
```

Di peramban, gagasan yang sama digabungkan dengan `AbortController`, yang juga membatalkan permintaan jaringan yang sudah usang [11]; penjaga tetap diperlukan, karena sebuah balasan mungkin sudah dalam perjalanan ketika pembatalan dikeluarkan.

Penyelesaian 9.8. Rilis lama mengamati *kemajuannya sendiri*: sampainya ia pada baris-baris terakhir skrip. Ketika `cmd.exe` melanjutkan dari posisi byte yang bergeser, skrip melewatkan pekerjaan tetapi tetap sampai pada baris-baris itu, sehingga pengamatannya benar (akhir skrip memang tercapai) tetapi tidak relevan (rilis belum lengkap). Langkah 6 mengamati *layanan yang sedang berjalan*: string versi yang dilaporkan proses, yang dibacanya ketika mulai. Proses yang tidak dihidupkan ulang, atau tidak diganti, tidak dapat melaporkan versi yang tidak pernah dipakainya untuk mulai, sehingga pengamatan itu hanya dapat berhasil jika versi baru benar-benar sedang melayani.

Penyelesaian 9.9. Aman: versi yang berjalan (diperlukan untuk memverifikasi rilis); status keseluruhan seperti ok, degraded, atau down (diperlukan pemantau); keadaan per dependensi seperti “basis data ok” (diperlukan untuk menemukan letak kegagalan); lama menyala (memperlihatkan penyalaan ulang yang tak terduga). Tidak boleh: kredensial atau *connection string* (menyerahkan akses kepada penyerang); jalur sistem berkas dan nama mesin (membantu penyerang memetakan sistem); data pribadi atau jumlah pengguna beserta namanya (privasi, UU PDP); ambang keamanan yang persis seperti batas laju (membantu penyerang tetap sedikit di bawahnya). Jika operator memerlukan rincian lebih banyak, tempatnya di balik autentikasi, bukan di *endpoint* publik [10].

Penyelesaian 9.10. *Liveness* bertanya apakah proses sebaiknya dibiarkan berjalan; *readiness* bertanya apakah ia sebaiknya menerima permintaan. Dalam `health.py`, jawaban apa pun berarti *hidup*. `ok` dan `degraded` sama-sama berarti *siap* (degraded berarti siap dengan fitur berkurang); `down` berarti *tidak siap*. Pengelola layanan sebaiknya menyalakan ulang proses hanya ketika proses tidak hidup (tidak ada jawaban, atau mogok), dan itulah yang dilakukan NSSM. Ia tidak sebaiknya menyalakan ulang pada `down`, karena jika basis data tidak dapat dijangkau, penyalaan ulang tidak membantu dan dapat membuat pemulihan makin riuh; kasus itu memerlukan peringatan kepada manusia. Pada `degraded` tidak diperlukan tindakan selain pelaporan, dan justru karena itulah degraded ada sebagai status tersendiri.

Daftar Pustaka

[1] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Eds., *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA, USA: O'Reilly Media, 2016, ch. 3 "Embracing Risk" and ch. 8 "Release Engineering."

[2] Cloudflare, "Cloudflare Tunnel," *Cloudflare One Docs*. [Online]. Available: https://developers.cloudflare.com/cloudflare-one/networks/connectors/cloudflare-tunnel/

[3] NSSM, "NSSM: the Non-Sucking Service Manager." [Online]. Available: https://nssm.cc/

[4] S. Ramírez, "FastAPI," documentation. [Online]. Available: https://fastapi.tiangolo.com/

[5] Encode, "Uvicorn: an ASGI web server for Python," documentation. [Online]. Available: https://www.uvicorn.org/

[6] WHATWG, "Server-sent events," in *HTML Living Standard*, sec. 9.2. [Online]. Available: https://html.spec.whatwg.org/multipage/server-sent-events.html

[7] The PostgreSQL Global Development Group, "Database roles," *PostgreSQL Documentation*, ch. 21. [Online]. Available: https://www.postgresql.org/docs/current/user-manag.html

[8] Profitina, *Profitina Whitepaper, Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[9] N. Harvey and D. DeBellis, "Announcing the 2025 DORA report," *Google Cloud Blog*, Sep. 23, 2025. [Online]. Available: https://cloud.google.com/blog/products/ai-machine-learning/announcing-the-2025-dora-report

[10] Microsoft, "Health Endpoint Monitoring pattern," *Azure Architecture Center*. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/patterns/health-endpoint-monitoring

[11] Mozilla, "AbortController," *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/AbortController

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th ed. Cambridge, MA, USA: MIT Press, 2022, ch. 6.

[13] Python Software Foundation, "heapq: Heap queue algorithm," *Python 3.14 Documentation*. [Online]. Available: https://docs.python.org/3/library/heapq.html

[14] Oracle, "Class PriorityQueue," *Java SE 25 & JDK 25 API Specification*, 2025. [Online]. Available: https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/util/PriorityQueue.html

[15] Ollama, "API reference," documentation. [Online]. Available: https://docs.ollama.com/api

Bab 10

Situs Pemasaran sebagai Studi Kasus GEO

Setelah menyelesaikan bab ini, Anda akan mampu menjelaskan bagaimana profitina.com, situs pemasaran publik Profitina, dibangun dan mengapa dibangun dengan cara itu: sebuah instalasi WordPress dengan tema dwibahasa khusus, bahasa Indonesia dan bahasa Inggris berdampingan, yang berada di belakang dua lapisan *cache*, satu di penyedia *hosting* dan satu lagi di *edge* jaringan penghantar konten (*content delivery network*, CDN). Anda akan mampu membaca berkas `robots.txt` tepat seperti yang ditetapkan RFC 9309, termasuk bagian-bagian yang sering mengejutkan orang: bagaimana grup-grup untuk *crawler* yang sama digabungkan, mengapa *crawler* yang memiliki grup sendiri mengabaikan grup `*`, dan apa yang wajib dilakukan *crawler* ketika berkas itu menjawab dengan galat. Anda akan mengenal insiden nyata yang mengajari tim Profitina untuk memeriksa `robots.txt` yang *tayang*, bukan berkas yang ada di server, dan mengapa insiden itu mudah terulang. Anda akan mampu menjelaskan apa itu `llms.txt`, berapa banyak situs yang menerbitkannya dan betapa sedikit berkas itu yang pernah dibaca, lengkap dengan sumbernya. Anda akan memahami *caching* HTTP cukup dalam untuk menghitung berapa lama sebuah halaman basi dapat bertahan ketika dua *cache* berdiri berlapis, dan mengapa urutan *purge* (pengosongan) keduanya menentukan. Anda akan tahu data terstruktur apa saja yang dibawa situs pemasaran itu, serta apa yang dapat dan tidak dapat dilakukan data itu setelah perubahan Google atas hasil FAQ pada Mei 2026. Terakhir, Anda akan membangun dan menjalankan sebuah auditor, dalam Python dan Java, yang memeriksa `robots.txt`, `llms.txt`, *header cache*, dan JSON-LD pada empat situs contoh kecil yang dijalankan di komputer Anda sendiri, dan yang menemukan, di antara temuan lain, kesalahan yang dulu pernah menyembunyikan profitina.com dari dua *crawler* AI.

10.1 Titik Berangkat Kita

Bab 9 membongkar aplikasi: satu *host* Windows, Cloudflare Tunnel yang tidak membuka satu pun *port* masuk, FastAPI dan PostgreSQL, dua jalur AI lokal, serta gerbang rilis yang baru menyatakan berhasil ketika *endpoint* kesehatan menjawab dengan versi yang baru. Bab itu membahas `app.profitina.com`, produk tempat pengguna masuk.

Bab ini beralih ke separuh merek yang lain, `profitina.com`, situs yang biasanya pertama kali dijumpai pendatang baru. Orang mudah menganggap situs pemasaran sekadar brosur, sesuatu yang dibuat sekali oleh desainer lalu tak dipikirkan lagi. Bagi perusahaan yang produknya adalah *visibilitas AI*, sikap semacam itu justru janggal. Profitina mengatakan kepada penggunanya bahwa mesin penjawab hanya dapat mengutip apa yang boleh diambil oleh *crawler*-nya, apa yang dapat diurainya, dan apa yang masih benar ketika *crawler* itu tiba. Maka situs Profitina sendiri adalah tempat pertama nasihat itu diuji, di depan umum, dan siapa pun dapat memeriksanya.

Karena itu situs pemasaran menjadi studi kasus yang luar biasa jujur. Setiap gagasan dari Bab 2, tempat Subbab 2.7 memperkenalkan data terstruktur, `robots.txt`, dan `llms.txt`, dan tempat lab G6 membangkitkan ketiganya untuk Profitina Laundry, di sini harus bertahan ketika bersentuhan dengan infrastruktur nyata: sistem manajemen konten dengan kebiasaannya sendiri, perusahaan

hosting dengan *cache*-nya sendiri, dan jaringan penghantar konten dengan pendapatnya sendiri tentang *crawler* mana yang boleh masuk. Sebagian gagasan itu bertahan utuh. Satu di antaranya gagal tanpa suara, dan kisah kegagalan itulah jantung bab ini.

Bab ini bergerak dari luar ke dalam. Subbab 10.2 menggambarkan dua permukaan merek dan tumpukan teknologi di balik situs pemasaran. Subbab 10.3 menunjukkan bagaimana tema dwibahasa dibangun dengan benar dan menceritakan sebuah kutu (*bug*) kecil yang sarat pelajaran. Subbab 10.4 dan 10.5 mempelajari `robots.txt` secara mendalam, lalu menceritakan insidennya. Subbab 10.6 menelaah `llms.txt` dengan angka adopsi dan pemakaian terbaru. Subbab 10.7 menjelaskan *caching* HTTP dan urutan *purge* dua lapisan *cache*, disertai sedikit hitungan. Subbab 10.8 membahas data terstruktur di situs itu. Subbab 10.9 membangun auditornya, dan Subbab 10.10 sampai 10.14 menutup bab dengan Profitina dalam praktik, kesalahpahaman yang sering terjadi, ringkasan, soal latihan, dan penyelesaian lengkapnya.

10.2 Dua Permukaan, Satu Merek

10.2.1 Mengapa perlu situs pemasaran tersendiri

Profitina memiliki dua alamat publik, dan keduanya mengerjakan tugas yang berbeda.

`app.profitina.com` adalah aplikasinya. Di sanalah pengguna mendaftar, menjalankan pemindaian, membangkitkan draf, mengamati garis tren, menulis di kanal komunitas, dan menggunakan hak-hak atas datanya. Aplikasi itu ditulis dalam Python dengan FastAPI, menyimpan datanya di PostgreSQL, dan setiap halamannya dihasilkan oleh kode yang ditulis dan diuji oleh tim. Judulnya di tab peramban adalah “Profitina — Dasbor GEO”.

`profitina.com` adalah situs pemasarannya. Situs ini menjelaskan apa itu Profitina, menerbitkan artikel, cerita, kiat, dan materi kursus, memuat tautan *whitepaper* dan pertanyaan yang sering diajukan, serta menyematkan tampilan umpan komunitas yang hanya dapat dibaca. Pada 24 September 2026, judul utamanya berbunyi “A self-reliant GEO platform — from us, for everyone”, dan judul halamannya “Profitina – Gather · Grow · Give” [1].

Mengapa tidak menaruh semuanya dalam satu aplikasi? Karena kedua tugas itu membutuhkan hal yang berlawanan. Aplikasi berubah setiap beberapa hari, diuji oleh lebih dari seribu uji otomatis, dan tidak boleh sekali pun menampilkan data seorang pengguna kepada pengguna lain. Situs pemasaran berubah ketika seseorang menulis artikel, harus dapat disunting oleh orang yang tidak menulis kode, dan menampilkan halaman yang sama kepada semua orang. Sistem manajemen konten adalah alat yang tepat untuk tugas kedua, dan aplikasi buatan sendiri adalah alat yang tepat untuk tugas pertama. Memisahkan keduanya juga memisahkan kegagalannya: tema yang rusak tidak dapat merobohkan pemindai, dan rilis aplikasi yang gagal tidak dapat menyembunyikan artikel.

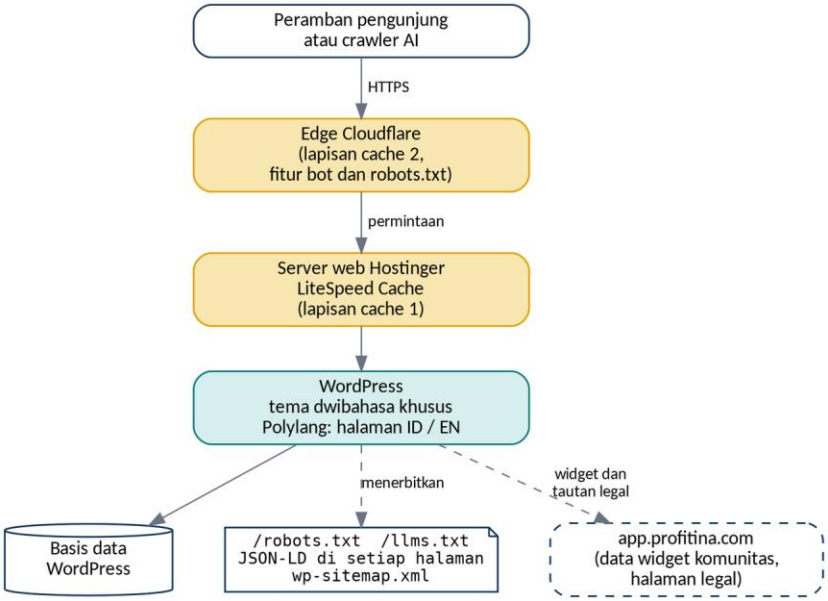
10.2.2 Mengapa WordPress

Situs pemasaran berjalan di atas WordPress. Pada 2026 pilihan itu hampir tidak perlu dibela. Menurut W3Techs, pada September 2026 WordPress dipakai oleh 40,2% dari seluruh situs web dan 58,8% dari situs yang sistem manajemen kontennya diketahui [2]. Bagi tim kecil, popularitas itu bukan soal tren, melainkan soal ketersediaan: perusahaan *hosting* menyetel servernya untuk WordPress, *plugin cache* tersedia untuknya, penerjemah mengenalnya, dan hampir setiap masalah sudah pernah ditemui dan dipecahkan orang lain.

Harga dari popularitas itu adalah bahwa situs WordPress dibentuk oleh konvensi yang tidak dipilih tim. Tema berjalan di dalam *loop* pembangun halaman milik WordPress sendiri dan berbagi variabel globalnya. *Plugin* menambahkan *header*-nya sendiri, berkasnya sendiri, dan *cache*-nya sendiri. Perusahaan *hosting* dapat menambahkan *cache* tingkat server, dan jaringan penghantar konten dapat menambahkan satu lagi. Masing-masing berguna, dan masing-masing adalah tempat di mana apa yang *ditulis* tim dan apa yang *diterima* pengunjung dapat bergeser menjauh. Sisa bab ini, dalam arti tertentu, adalah katalog tempat-tempat itu.

10.2.3 Tumpukan teknologi, dari pengunjung ke dalam

Gambar 10.1 memperlihatkan jalur yang ditempuh sebuah permintaan menuju profitina.com.



Gambar 10.1 Tumpukan teknologi situs pemasaran. Permintaan melewati *edge* Cloudflare (lapisan *cache* 2) dan LiteSpeed Cache milik *host* (lapisan *cache* 1) sebelum WordPress dan tema dwibahasanya menyusun halaman; aplikasi hanya dijangkau melalui widget yang disematkan dan tautan legal.

Dibaca dari atas ke bawah:

1. **Pengunjung.** Seseorang dengan peramban, atau sebuah program: *crawler* mesin pencari, *crawler* AI, bot pratinjau tautan, alat audit SEO. Semuanya meminta URL yang sama, dan semuanya menerima apa pun yang diputuskan oleh lapisan-lapisan di depan WordPress.
2. ***Edge* Cloudflare.** Cloudflare mengakhiri koneksi HTTPS di dekat pengunjung, menyaring lalu lintas yang merugikan, dan dapat menyimpan salinan respons. Cloudflare juga menawarkan fitur yang *mengubah* respons, termasuk, seperti akan diperlihatkan Subbab 10.5, `robots.txt` terkelola. Inilah lapisan *cache* 2, lapisan luar.
3. ***Host* web.** Situs ini di-*hosting* di Hostinger pada server web LiteSpeed, dengan *plugin* LiteSpeed Cache untuk WordPress. LiteSpeed menyimpan salinan halaman yang siap saji sehingga WordPress tidak perlu menyusun ulang setiap halaman bagi setiap pengunjung. Inilah lapisan *cache* 1, lapisan dalam.
4. **WordPress dan tema.** Ketika kedua *cache* tidak punya jawaban, WordPress berjalan, tema khusus menyusun halaman dalam bahasa yang diminta, dan basis data menyediakan kontennya.
5. **Apa yang diterbitkan situs.** Selain halaman, situs menyajikan `/robots.txt`, `/llms.txt`, peta situs XML di `/wp-sitemap.xml`, dan data terstruktur JSON-LD di dalam halamannya.

6. **Tautan ke aplikasi.** Widget komunitas di halaman pemasaran membaca umpan publik dari aplikasi, dan tautan Kebijakan Privasi serta Syarat dan Ketentuan di kaki halaman mengarah ke `app.profitina.com/legal/`, satu-satunya rumah dokumen-dokumen itu.

Butir terakhir itu adalah keputusan rancangan yang disengaja, dengan nama pendek dalam catatan tim: *hapus duplikasinya, jangan menyinkronkannya* [3]. Kebijakan Privasi dulu ada di dua tempat. Dua salinan teks hukum berarti dua peluang untuk keliru, dan cepat atau lambat salah satunya memang keliru. Sejak akhir Agustus 2026 kebijakan itu hanya ada di aplikasi, sementara situs pemasaran meringkas dan menautkannya. Bab 11 kembali ke keputusan itu dari sisi hukum.

> **WAWASAN KUNCI:** Setiap lapisan antara pembuat halaman dan pembacanya adalah tempat versi yang tayang dapat berbeda dari versi yang ditulis. Karena itu audit GEO harus melihat apa yang diterima *dunia*, pada URL publik, bukan berkas di server atau di repositori.

10.2.4 Apa berada di mana

Tabel 10.1 merangkum pembagian kerjanya.

Tabel 10.1 Dua permukaan publik merek Profitina dan tanggung jawab masing-masing.

Urusan	Profitina.com (WordPress)	App.profitina.com (FastAPI)
Khalayak	Siapa pun, termasuk setiap *crawler*	Pengguna terdaftar, ditambah tampilan publik yang hanya-baca
Konten	Artikel, cerita, kiat, materi kursus, FAQ, tautan *whitepaper*	Pemindaian, draf, tren, komunitas, akun, PROFIT
Bahasa	Halaman bahasa Indonesia dan Inggris (Polylang)	Satu antarmuka dengan kamus ID/EN
Berkas yang dibaca mesin	`robots.txt`, `llms.txt`, `wp-sitemap.xml`, JSON-LD	`robots.txt`, `sitemap.xml`, *endpoint* API
Caching	LiteSpeed Cache di *host*, Cloudflare di *edge*	Tingkat aplikasi; *edge* meneruskan terowongan apa adanya
Teks hukum	Hanya ringkasan dan tautan	Satu-satunya rumah: `/legal/privacy`, `/legal/terms`
Perubahan	Ketika penyunting menerbitkan	Melalui gerbang rilis teruji di Bab 9

10.3 Tema Dwibahasa yang Dikerjakan dengan Benar

10.3.1 Dua bahasa, dua himpunan halaman

Profitina mengutamakan Indonesia, dan khalayaknya membaca bahasa Indonesia, bahasa Inggris, atau keduanya. Situs dwibahasa dapat dibangun dengan dua cara yang sangat berbeda. Cara murah adalah menulis setiap halaman sekali lalu menerjemahkannya seketika, di peramban, dengan widget penerjemah mesin. Cara cermat adalah memelihara dua versi nyata untuk setiap halaman, satu per bahasa, masing-masing dengan alamatnya sendiri, judulnya sendiri, dan teksnya sendiri yang ditulis, atau setidaknya diperiksa, oleh manusia.

profitina.com menempuh cara cermat. Temanya adalah tema WordPress buatan sendiri, dan lapisan bahasanya adalah *plugin* Polylang, yang memungkinkan penyunting membuat terjemahan setiap tulisan, halaman, kategori, dan menu lalu menautkan terjemahan-terjemahan itu satu sama lain [4]. Polylang memelihara menu terpisah untuk setiap bahasa. Bagi penyunting itu berarti kerja tambahan, tetapi hasilnya menu yang terbaca wajar dalam setiap bahasa, bukan label terjemahan yang dijejalkan ke tata letak yang dirancang untuk bahasa lain.

Mengapa hal ini penting bagi GEO? Karena mesin penjawab yang membaca sebuah halaman perlu tahu bahasa apa yang sedang dibacanya dan apakah ada versi lain. Tiga sinyal kecil membawa informasi itu:

- atribut `lang` pada elemen `<html>`, `id` atau `en`, yang memberi tahu setiap program bahasa halaman itu;
- alamat setiap versi, yang harus stabil agar dapat dikutip;
- tautan `hreflang`, yang memberi tahu mesin pencari bahwa halaman berbahasa Indonesia dan halaman berbahasa Inggris adalah terjemahan satu sama lain, sehingga pembaca diarahkan ke versi yang tepat [5].

Halaman terjemahan mesin tidak memiliki satu pun dari ketiganya dengan benar: alamatnya hanya satu, bahasa yang dinyatakan hanya satu, dan teks terjemahannya hanya ada di peramban pengunjung. Tidak ada *crawler* yang pernah melihat terjemahan itu, sehingga tidak ada mesin yang dapat mengutipnya.

10.3.2 Kutu `$id`: pelajaran tentang nama bersama

Riwayat tema ini menyimpan satu kutu kecil yang layak diceritakan, karena polanya muncul di setiap *framework* yang menjalankan kode Anda di dalam *loop* miliknya.

Di dalam *loop* WordPress, fungsi yang menyiapkan setiap tulisan untuk ditampilkan, `setup_postdata()` (dipanggil oleh `the_post()`), mengisi sejumlah variabel global, di antaranya `$id`, pengenal numerik tulisan yang sedang diproses [6]. Salah satu versi awal tema profitina.com, pada lini 2.20, juga memakai variabel bernama `$id` di kode templatnya, untuk menampung pengenal label bagian *hero* sesuai bahasa. Dalam PHP, templat yang dimuat WordPress berbagi lingkup tempat variabel-variabel global itu hidup. Dua pemakaian nama yang sama itu pun bertabrakan. Gejalanya termasuk jenis yang paling buruk: bukan *crash*, bukan pesan galat, melainkan label berbahasa Indonesia yang sesekali muncul di *hero* berbahasa Inggris, bergantung pada urutan kejadian di dalam *loop* suatu halaman.

Pandangan pertama atas gejala “bahasanya tercampur” langsung tertuju ke logika bahasa. Logika itu benar. Kesalahannya ada satu tingkat di bawahnya, pada sebuah nama. Perbaikannya sama sederhananya, dan yang penting, merupakan perbaikan akar masalah, bukan tambalan: variabel milik tema diganti namanya menjadi `$pf_id`, dengan awalan yang tidak akan pernah dipakai WordPress, dalam serangkaian rilis kecil tema. Aturan umum yang diajarkannya adalah aturan yang cepat atau lambat dipelajari setiap pengguna *framework*:

> **PERINGATAN:** Ketika kode Anda berjalan di dalam *loop* milik pihak lain, setiap nama tanpa kualifikasi yang Anda pilih mungkin sudah menjadi milik *framework* itu. Beri awalan pada nama Anda sendiri (`pf_`, *namespace*, kelas), dan ketika sebuah kutu muncul sesekali dan bergantung pada urutan kejadian, curigai keadaan bersama (*shared state*) sebelum mencurigai logika Anda.

Kebiasaan yang sama, yakni memberi setiap perubahan tema nomor versi dan satu baris di catatan perubahan, itulah yang membuat kutu ini dapat dilacak. Kalimat “muncul sebelum versi ini, diperbaiki di rilis-rilis ini” hanya dapat ditulis oleh proyek yang berversi.

10.4 robots.txt dari Prinsip Dasar

10.4.1 Apa yang sudah ditetapkan Bab 2

Subbab 2.7.3 memperkenalkan `robots.txt`: berkas teks biasa di akar sebuah situs, dibakukan sebagai *Robots Exclusion Protocol* dalam RFC 9309 pada September 2022 [7], yang tersusun atas grup-grup, masing-masing berupa daftar baris `User-agent` diikuti aturan `Allow` dan `Disallow`. Subbab itu juga memberikan prosedur keputusannya: *crawler* memakai grup yang menyebut namanya, atau grup `*` bila tidak ada; aturan yang cocok paling panjang menang; bila seri, `Allow` menang; dan bila tidak ada aturan yang cocok, jalur itu diizinkan. Tabel *crawler* di Subbab 2.7.4 mendaftar *crawler* milik OpenAI, Anthropic, Perplexity, Google, dan Microsoft, dan lab G6 membangkitkan berkas yang mengizinkan *crawler* yang biasanya diinginkan usaha kecil.

Subbab ini melangkah lebih jauh, ke bagian-bagian RFC 9309 yang penting ketika berkas yang Anda tulis bukan satu-satunya berkas yang terlibat, atau bukan berkas yang sampai ke tujuan. Kita membutuhkan empat gagasan lagi: bagaimana baris-baris berkas menjadi grup, apa yang terjadi bila dua grup menyebut *crawler* yang sama, apa arti karakter pengganti (*wildcard*), dan apa yang dilakukan *crawler* bila berkas itu tidak dapat dibaca.

10.4.2 Dari baris ke grup

Crawler membaca berkas baris demi baris. Segala sesuatu setelah `#` adalah komentar. Setiap baris yang tersisa berbentuk `kunci: nilai`, dan kuncinya tidak membedakan huruf besar dan kecil. **Grup** baru dimulai pada baris `User-agent` yang muncul setelah baris aturan (atau pada baris `User-agent` pertama dalam berkas); baris-baris `User-agent` yang berurutan termasuk grup yang sama. Baris aturan (`Allow`, `Disallow`) melekat pada grup yang sedang terbuka. Baris lain, seperti `Sitemap`, tidak menjadi bagian grup mana pun dan boleh muncul di mana saja.

> **DEFINISI.** *Grup* dalam berkas `robots.txt` adalah rangkaian maksimal satu atau lebih baris `User-agent` yang diikuti baris-baris aturan hingga baris `User-agent` berikutnya yang muncul setelah sebuah aturan. *Token produk* pada baris `User-agent` dicocokkan dengan nama *crawler* tanpa membedakan huruf besar dan kecil.

Ambil berkas contoh Profitina Laundry dari lab bab ini:

```
# Profitina Laundry (sample site for Chapter 10)
User-agent: *
Content-Signal: search=yes, ai-input=yes, ai-train=no
Allow: /
Disallow: /admin/

User-agent: GPTBot
User-agent: OAI-SearchBot
User-agent: ClaudeBot
User-agent: PerplexityBot
User-agent: Google-Extended
Allow: /
Disallow: /admin/

Sitemap: http://127.0.0.1:7100/sitemap.xml
```

Berkas ini memiliki dua grup. Grup pertama hanya menyebut `*` dan memiliki dua aturan. Grup kedua menyebut lima *crawler*, karena lima baris `User-agent` berurutan tanpa aturan di antaranya, dan memiliki dua aturan yang sama. Baris `Sitemap` tidak termasuk grup mana pun. Baris `Content-Signal` sama sekali bukan bagian RFC 9309; Subbab 10.5.4 menjelaskan asal-usulnya, dan pembaca yang patuh standar cukup melewati kunci yang tidak dikenalnya.

10.4.3 Memilih grup: penggabungan dan jalur cadangan

Misalkan a adalah nama sebuah *crawler* yang ditulis dengan huruf kecil, dan agents(g) adalah himpunan token produk grup g dalam huruf kecil. Himpunan grup yang wajib dipatuhi *crawler* a adalah

$$G(a) = \begin{cases} \{\, g : a \in \text{agents}(g) \,\} & \text{jika himpunan ini tidak kosong,} \\ \{\, g : * \in \text{agents}(g) \,\} & \text{jika tidak.} \end{cases} \tag{10.1}$$

Persamaan (10.1) memuat dua aturan yang dinyatakan RFC 9309 secara gamblang [7]. Pertama, bila *lebih dari satu* grup menyebut *crawler* itu, aturan-aturannya digabung menjadi satu daftar; *crawler* tidak berhenti pada grup pertama yang menyebutnya. Kedua, grup `*` adalah *cadangan*, bukan bawaan yang ditambahkan ke semua *crawler*: *crawler* yang disebut oleh grup mana pun mengabaikan grup `*` sepenuhnya.

Aturan kedua inilah yang menggigit. Misalkan sebuah berkas berbunyi

```
User-agent: *
Allow: /

User-agent: GPTBot
Disallow: /private/
```

Pemilik situs yang membacanya mungkin berpikir "semua orang boleh membaca semuanya, dan GPTBot tidak boleh membaca `/private/`". Itu benar, tetapi hanya kebetulan. Ubah grup kedua menjadi `Disallow: /`, maka GPTBot tidak boleh membaca apa pun, walaupun grup pertama berkata `Allow: /` untuk "semua orang". GPTBot bukan lagi bagian dari "semua orang": ia punya grup sendiri. Subbab 10.5 memperlihatkan bagaimana tepat aturan ini, digabung dengan berkas yang tidak pernah ditulis pemilik situs, menyembunyikan profitina.com dari dua *crawler*.

10.4.4 Memilih aturan: kecocokan terpanjang, `Allow` bila seri

Setelah grup dipilih, aturan-aturannya membentuk satu daftar $R(a)$. Untuk sebuah jalur p, misalnya `/blog/post-1`, misalkan $M(a, p)$ adalah aturan-aturan di $R(a)$ yang polanya cocok dengan p, dan $\ell(r)$ adalah panjang pola aturan r dalam oktet. Aturan penentunya adalah

$$r^{*} = \arg\max_{r \in M(a,p)} \left(\ell(r),\ [\, r \text{ adalah Allow} \,] \right) \tag{10.2}$$

dengan pasangan-pasangan dibandingkan secara leksikografis: pertama menurut panjang, dan hanya di antara pola yang sama panjang menurut apakah aturan itu `Allow` (dihitung 1) atau `Disallow` (dihitung 0). Bila $M(a, p)$ kosong, jalur itu diizinkan. Bila tidak, jalur itu diizinkan tepat ketika r^{*} adalah aturan `Allow`. Persamaan (10.2) adalah seluruh aturan prioritas RFC 9309 dalam satu baris: "kecocokan paling spesifik yang ditemukan WAJIB digunakan", dan "bila aturan allow dan aturan disallow setara, aturan allow SEBAIKNYA digunakan" [7]. Dokumentasi Google menyatakan hal yang sama untuk *crawler*-nya [8].

> **CONTOH:** Untuk grup `User-agent: *` dengan `Disallow: /` dan `Allow: /blog/`, serta jalur `/blog/post-1`: kedua pola cocok, dengan panjang 1 dan 6. Pasangan untuk `Disallow: /` adalah (1,0) dan pasangan untuk `Allow: /blog/` adalah (6,1); pasangan yang lebih besar adalah (6,1), sehingga jalur itu diizinkan. Untuk jalur `/about`, hanya `Disallow: /` yang cocok, M memiliki satu unsur, dan jalur itu diblokir.

Urutan baris dalam berkas tidak berperan sama sekali dalam Persamaan (10.2). Inilah perbedaan nyata dari cara baca `robots.txt` yang lebih lama dan tidak resmi, yang membiarkan aturan

pertama yang cocok memutuskan, dan perbedaan itulah yang selama bertahun-tahun diterapkan oleh `urllib.robotparser` milik Python. Subbab 10.9.6 menjalankan kedua cara baca itu berdampingan.

10.4.5 Karakter pengganti

RFC 9309 memberi dua karakter makna khusus di dalam pola [7]:

- `*` cocok dengan rangkaian karakter apa pun, termasuk rangkaian kosong, sehingga `Disallow: /*.pdf` memblokir `/files/report.pdf` dan `/a.pdf`;
- `$` di ujung paling akhir menambatkan pola pada akhir jalur, sehingga `Disallow: /*.pdf$` memblokir `/a.pdf` tetapi tidak `/a.pdf?download=1`.

Tanpa `$` di akhir, sebuah pola cocok dengan setiap jalur yang *diawali* olehnya; itulah sebabnya `Disallow: /admin` juga memblokir `/administrator`. Panjang $\ell(r)$ dalam Persamaan (10.2) adalah panjang pola sebagaimana tertulis, termasuk karakter penggantinya.

Pembaca dalam lab ini menerapkan kedua karakter pengganti dengan mengubah pola menjadi ekspresi reguler: bagian-bagian di antara karakter `*` di-*escape* apa adanya, `*` menjadi `.*`, dan pola ditambatkan di akhir hanya bila semula diakhiri `$`. Subbab 10.9.3 memperlihatkan belasan baris yang mengerjakannya.

10.4.6 Ketika berkas tidak dapat dibaca

Crawler yang meminta `/robots.txt` tidak selalu mendapat berkas. RFC 9309 membedakan kasus-kasusnya menurut status HTTP, dan **Tabel 10.2** merangkumnya [7], [8].

Tabel 10.2 Cara RFC 9309 meminta *crawler* memperlakukan setiap jenis respons atas permintaan `robots.txt`. Google juga memperlakukan status 429 seperti galat server.

Respons atas GET `/robots.txt`	Apa yang wajib atau boleh dilakukan *crawler*	Alasannya
2xx dengan berkas	Mengurai dan mematuhinya	Kasus normal
Pengalihan 3xx	Mengikuti setidaknya lima pengalihan, lalu memperlakukannya sebagai tidak tersedia	Situs memindahkan berkas
4xx, "tidak tersedia" (404, 410, 403…)	Boleh merayapi semuanya, seolah berkas tidak ada	Pemilik tidak menerbitkan aturan
5xx atau tanpa jawaban, "tidak terjangkau"	Wajib menganggap semuanya dilarang	Aturan mungkin ada tetapi tak terbaca
Respons lebih besar dari 500 KiB	Boleh mengabaikan sisanya; wajib mengurai setidaknya 500 KiB	Batas beban penguraian

Asimetri antara 4xx dan 5xx disengaja, dan mudah terbalik dalam ingatan. Status 404 berarti "tidak ada aturan": situs tidak menerbitkan berkas, jadi tidak ada yang dilarang. Status 500 berarti "ada aturan, tetapi saya tidak dapat menunjukkannya": anggapan yang aman adalah bahwa semuanya dilarang. Situs yang `robots.txt`-nya rusak dengan galat server pun lenyap dari *crawler* yang santun, sementara tidak ada yang tampak salah di halaman-halamannya. Google menambahkan satu rincian praktis: selama 12 jam pertama galat semacam itu, Google berhenti merayapi situs sambil terus mencoba mengambil berkasnya [8]. Salah satu dari empat situs contoh di lab, `broken`, mengidap kesalahan ini persis.

RFC 9309 juga mengizinkan *crawler* menyimpan berkas itu dalam *cache*, tetapi menyatakan bahwa *crawler* "SEBAIKNYA TIDAK memakai versi *cache* lebih dari 24 jam, kecuali berkas robots.txt tidak terjangkau" [7], dan Google menyatakan bahwa ia "umumnya menyimpan isi

berkas robots.txt dalam *cache* hingga 24 jam" [8]. Hal ini penting di Subbab 10.7: perbaikan `robots.txt` dapat memerlukan hingga satu hari untuk sampai ke sebuah *crawler* walaupun *cache* Anda sendiri kosong, sehingga satu hari *caching* tambahan di atasnya adalah satu hari yang sebenarnya tidak perlu hilang.

10.4.7 Membaca dan akses adalah dua hal yang berbeda

Dua batas `robots.txt` yang sudah dinyatakan di Bab 2 layak diulang, karena insiden di subbab berikut justru tentang kesalahan sebaliknya. `robots.txt` adalah *permintaan* yang dihormati *crawler* yang santun, bukan kendali akses: halaman yang harus privat memerlukan autentikasi. Dan berkas itu publik: baris `Disallow` justru mengumumkan jalur yang hendak disembunyikannya. Kesalahan di Subbab 10.5 bukanlah halaman yang terbuka; melainkan halaman-halaman yang ditujukan bagi semua orang diam-diam ditahan dari sebagian pembaca yang paling penting.

10.5 Pelajaran robots.txt: Ketika Berkas yang Tayang Bukan Berkas Anda

10.5.1 Apa yang terjadi

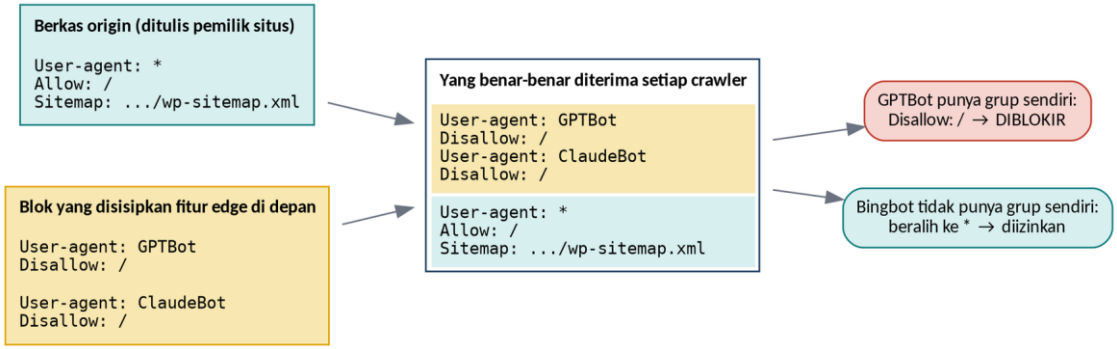
profitina.com memiliki `robots.txt` yang mengerjakan apa yang semestinya dikerjakan situs milik sebuah platform GEO: mengizinkan *crawler* mesin pencari dan *crawler* AI dengan menyebut namanya satu per satu. Berkas itu benar di tema dan benar di server. Namun untuk sementara waktu, dua *crawler* yang disambutnya, `GPTBot` milik OpenAI dan `ClaudeBot` milik Anthropic, justru disuruh menjauh.

Penyebabnya adalah fitur jaringan penghantar konten di depan situs itu, yaitu `robots.txt` *terkelola* (*managed*). Ketika fitur itu dinyalakan, *edge* tidak sekadar meneruskan berkas milik situs. *Edge* menulis blok aturannya sendiri, menaruhnya di depan aturan situs, dan setiap pengunjung, termasuk setiap *crawler*, menerima gabungannya. Dokumentasi Cloudflare sendiri menggambarkan perilakunya dengan tepat: "Cloudflare akan menyisipkan `robots.txt` terkelola kami di depan `robots.txt` Anda yang sudah ada, menggabungkan keduanya menjadi satu respons" [9]. Blok terkelola itu menyebut sederet *crawler* AI, antara lain `GPTBot`, `ClaudeBot`, `Google-Extended`, `Applebot-Extended`, `CCBot`, `Bytespider`, `Amazonbot`, dan `meta-externalagent`, lalu melarang semuanya [9].

Tidak ada yang berubah di halaman-halaman situs. Tidak ada yang salah di repositori tim. Orang yang membuka `robots.txt` di tema atau berkas di server melihat aturan yang dimaksud. Hanya permintaan ke alamat publik, `https://profitina.com/robots.txt`, yang memperlihatkan apa yang sebenarnya diterima dunia. Aturan tim yang lahir dari peristiwa itu pendek saja: *selalu verifikasi URL yang tayang, dengan melewati setiap cache* [3].

10.5.2 Mengapa penyisipan di depan sudah cukup untuk memblokir

Menyisipkan di depan kedengarannya tidak berbahaya. Bagaimanapun, aturan milik situs masih ada di dalam berkas. **Gambar 10.2** memperlihatkan mengapa hal itu tetap berbahaya, dengan memakai Persamaan (10.1).



Gambar 10.2 Blok terkelola yang disisipkan di *edge*. Grup `*` milik origin masih mengizinkan semua pihak, tetapi *crawler* yang disebut di blok terkelola kini memiliki grup sendiri dan tidak pernah melihat `*`; *crawler* yang tidak disebut tetap kembali ke `*`.

Misalkan berkas milik situs hanya berbunyi `User-agent: *` dan `Allow: /`, cara yang umum dan sepenuhnya wajar untuk mengatakan "semua pihak dipersilakan". Setelah penyisipan, berkas yang tayang memiliki grup untuk `GPTBot` dengan `Disallow: /`, grup untuk `ClaudeBot` dengan `Disallow: /`, dan grup `*` milik situs di bagian akhir. Untuk `GPTBot`, kasus pertama Persamaan (10.1) berlaku: ada grup yang menyebut namanya, sehingga G(gptbot) hanyalah grup itu, dan grup `*` tidak pernah dirujuk. Satu-satunya aturannya, `Disallow: /`, cocok dengan setiap jalur. `GPTBot` diblokir di mana-mana. `Bingbot`, yang tidak disebut oleh blok terkelola, jatuh ke kasus kedua Persamaan (10.1), mematuhi grup `*`, dan diizinkan.

Bagaimana jika berkas milik situs menyebut `GPTBot` secara eksplisit, seperti berkas profitina.com saat ini? Maka Persamaan (10.1) menggabungkan kedua grup yang menyebut namanya, dan daftar aturannya memuat `Disallow: /` dari *edge* sekaligus `Allow: /` dari situs. Kedua pola sama-sama sepanjang 1, sehingga menurut Persamaan (10.2) seri itu dimenangkan `Allow`, dan *crawler* yang mematuhi RFC 9309 kata demi kata akan diizinkan. Namun tiga fakta membuat hal itu bukan penghiburan. Pertama, aturan situs mungkin lebih spesifik, misalnya `Allow: /blog/`, dan kemudian `Disallow: /` dari *edge* menang untuk setiap jalur di luar `/blog/`. Kedua, tidak setiap *crawler* menerapkan penggabungan dan aturan seri; *crawler* yang berhenti pada grup pertama yang menyebut namanya hanya melihat `Disallow: /` dari *edge*. Ketiga, dan yang terpenting, berkas yang sekaligus berkata "pergilah" dan "selamat datang" kepada pembaca yang sama adalah berkas yang maknanya bergantung pada pengurai si pembaca. Situs yang ingin dikutip tidak boleh menyerahkan hal itu pada untung-untungan.

> **PERINGATAN:** Grup yang menyebut sebuah *crawler* mengeluarkan *crawler* itu dari grup `*` sepenuhnya. Lapisan mana pun yang menambahkan grup untuk sebuah *crawler*, entah fitur CDN, *plugin* keamanan, atau *plugin* SEO, diam-diam menimpa apa yang dikatakan grup `*` Anda tentang *crawler* itu.

10.5.3 Bawaan pun bergeser

`robots.txt` terkelola bukanlah pengaturan yang eksotis. Cloudflare memperkenalkannya pada 1 Juli 2025, sebagai bagian dari serangkaian kendali baru atas *crawler* AI [10]. Setahun kemudian, pada 1 Juli 2026, Cloudflare mengumumkan kendali yang lebih halus yang memisahkan lalu lintas AI menjadi pemakaian untuk pencarian, agen, dan pelatihan, dengan bawaan baru sejak 15 September 2026 bagi domain yang baru bergabung dan menampilkan iklan: bot pelatihan dan bot agen diblokir di halaman-halaman itu, bot pencarian diizinkan [11]. Apakah semua itu tepat bagi suatu situs adalah keputusan pemiliknya; yang penting bagi seorang insinyur adalah bahwa perilaku lapisan di depan situs Anda sebagian ditetapkan oleh pihak lain, dan berubah dari waktu

ke waktu. Pemeriksaan yang lolos pada bulan Juni dapat gagal pada bulan September tanpa seorang pun di tim Anda menyentuh apa pun.

10.5.4 Berkas yang disajikan profitina.com hari ini

Pada 24 September 2026, berkas yang tayang di `https://profitina.com/robots.txt` menutup `/wp-admin/` (dengan pengecualian lazim untuk `/wp-admin/admin-ajax.php`), secara eksplisit mengizinkan sederet panjang *crawler* mesin pencari dan AI, termasuk `GPTBot`, `ClaudeBot`, `Claude-Web`, dan `PerplexityBot`, menunjuk peta situs di `/wp-sitemap.xml`, dan menyebut `llms.txt` milik situs itu [12]. Berkas itu juga memuat satu baris yang tidak didefinisikan RFC 9309:

```
Content-Signal: search=yes, ai-input=yes, ai-train=no
```

Inilah *sinyal konten* (*content signal*), pernyataan yang dapat dibaca mesin tentang bagaimana pemilik ingin kontennya *dipakai*, berbeda dari boleh tidaknya konten itu *diambil*. Kosakatanya, yang diusulkan Cloudflare, memiliki tiga kunci: `search` (membangun indeks pencarian dan menampilkan hasil), `ai-input` (memasukkan konten ke model saat menjawab, seperti pada *retrieval-augmented generation*), dan `ai-train` (melatih atau menyetel ulang model) [9]. Baris milik profitina.com berkata: indekslah kami, kutiplah kami dalam jawaban, tetapi mohon jangan berlatih dengan konten kami. Kombinasi itu selaras dengan tujuan sebuah platform GEO, dan memperlihatkan bahwa "izinkan AI atau tidak" sebenarnya adalah beberapa pertanyaan yang terpisah.

Dua catatan menjaga sinyal konten tetap pada proporsinya. Sinyal itu pernyataan preferensi yang mungkin dibaca, mungkin tidak, oleh sebuah *crawler*; dengan sendirinya ia tidak memberi maupun menahan apa pun. Dan aturan akses tetap yang menentukan: *crawler* yang dilarang tidak pernah membaca kontennya, apa pun bunyi sinyalnya.

10.5.5 Pelajaran umumnya

Insiden itu dapat digeneralisasi menjadi daftar periksa yang dapat diterapkan situs mana pun dan yang diotomatiskan oleh lab di Subbab 10.9:

1. **Ambil URL publiknya, bukan berkasnya.** Yang penting adalah apa yang dikembalikan `https://situs-anda/robots.txt` kepada orang asing.
2. **Ambil dua kali, sekali dengan melewati *cache*.** Permintaan dengan *query string* unik, seperti `?audit-bypass=1`, biasanya luput dari setiap *cache* yang berkuncikan URL lengkap dan sampai ke origin. Bila kedua jawaban berbeda, ada *cache* yang menyimpan salinan lama.
3. **Bandingkan dengan origin itu sendiri.** Bila penyedia *hosting* menyediakan cara menjangkau server origin secara langsung, bandingkan pula dengannya. Bila jawaban yang melewati *cache* berbeda dari jawaban origin, ada sesuatu di antara keduanya yang *menulis ulang* berkas itu; ini sama sekali bukan soal salinan basi.
4. **Evaluasi aturannya seperti *crawler* mengevaluasinya.** Jangan menilai teksnya dengan mata; jalankan melalui pembaca yang menerapkan Persamaan (10.1) dan Persamaan (10.2), untuk setiap *crawler* yang Anda pedulikan.
5. **Ulangi secara terjadwal.** Lapisan-lapisan mengubah bawaannya (Subbab 10.5.3). Penjadwal di Bab 9 adalah tempat yang wajar bagi pemeriksaan semacam ini.

10.6 llms.txt: Apa Itu, dan Apa yang Bukan

10.6.1 Usulannya

Subbab 2.7.5 memperkenalkan `llms.txt`, yang diusulkan pada 3 September 2024 oleh Jeremy Howard sebagai berkas Markdown di akar situs yang memberi model bahasa peta situs yang singkat dan terkurasi [13]. Formatnya mudah dinyatakan dengan tepat:

1. baris H1 berisi nama situs, "satu-satunya bagian yang wajib";
2. kutipan blok (*blockquote*) berisi ringkasan singkat;
3. secara opsional, paragraf atau daftar berisi rincian lebih lanjut, tetapi tanpa judul;
4. secara opsional, bagian-bagian berjudul H2 yang memuat daftar tautan, masing-masing `- [judul](url): catatan`;
5. menurut konvensi, bagian `## Optional` untuk tautan yang boleh dilewati ketika konteks terbatas [13].

Usulan itu juga menyarankan agar halaman-halaman penting disediakan dalam versi Markdown yang bersih di alamat yang sama dengan tambahan `.md`. Gagasannya masuk akal: jendela konteks model bahasa kecil dibandingkan sebuah situs web, dan HTML sebuah halaman penuh dengan navigasi, skrip, dan penataan gaya yang bagi model hanyalah derau.

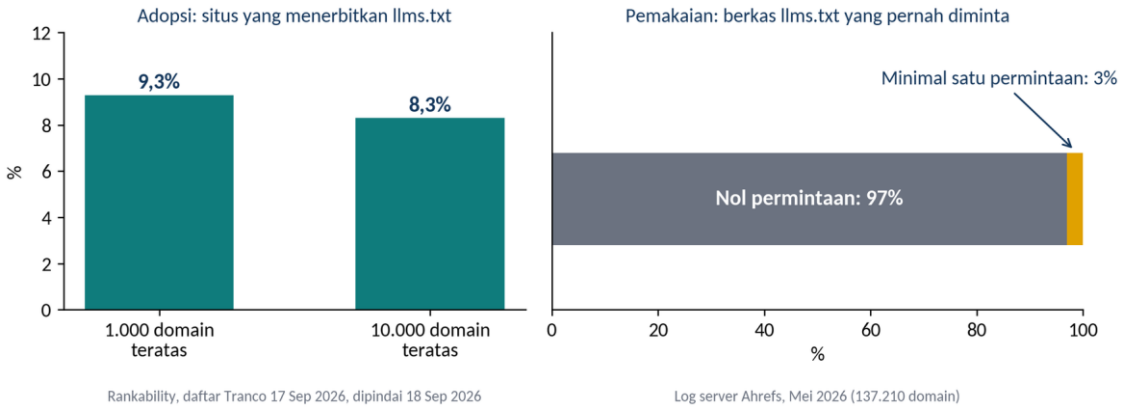
profitina.com menerbitkan sebuah `llms.txt` [14]. Isinya menggambarkan Profitina sebagai platform GEO yang mengukur seberapa siap sebuah merek dikutip mesin penjawab AI berdasarkan sinyal nyata situs, bukan dengan menanyai mesin-mesin itu, menyebut model Qwen lokal dua jalur, rancangan yang mengutamakan Indonesia dan kepatuhan pada UU PDP, serta menyatakan terus terang bahwa PROFIT adalah token utilitas di Solana devnet, bukan efek dan bukan investasi. Dengan kata lain, berkas itu adalah ringkasan satu halaman yang setia pada apa yang dikatakan bagian lain situs.

10.6.2 Adopsi dan pemakaian

Di sini angka yang jujur lebih penting daripada gagasannya, dan ada dua angka yang sangat berbeda.

Angka pertama adalah **adopsi**: berapa banyak situs yang menerbitkan berkas itu. Pemindaian pada 18 September 2026 atas daftar Tranco tertanggal 17 September 2026 menemukan `llms.txt` (atau `llms-full.txt`) pada 93 dari 1.000 domain teratas, 9,3%, dan pada 830 dari 10.000 domain teratas, 8,3% [15]. Di antara lebih dari tiga juta situs yang dipantau, satu pelacak menghitung 36.120 situs dengan berkas itu pada Mei 2026, dibandingkan 4.088 setahun sebelumnya, kenaikan 8,8 kali lipat [16].

Angka kedua adalah **pemakaian**: seberapa sering ada yang membaca berkas itu. Ahrefs menelaah log server untuk 137.210 domain pada Mei 2026. Dari berkas-berkas yang ada, 97% tidak menerima satu permintaan pun sepanjang bulan itu [17]. Di antara berkas yang diminta, 96% permintaan datang dari bot, bukan manusia, dan 77% bot itu sama sekali bukan alat AI, melainkan alat audit SEO, *crawler* umum, dan pemrofil teknologi; bot penelusuran di balik produk pencarian AI hanya menyumbang 1,1% dari seluruh permintaan [17]. Studi itu juga mengamati bahwa bot AI tidak mencari-cari berkas `llms.txt` yang tidak ada; berkas itu diambil ketika ada sesuatu yang menunjuk ke sana, bukan atas prakarsa bot sendiri [17]. **Gambar 10.3** menempatkan kedua sisi itu berdampingan.



Gambar 10.3 Adopsi dan pemakaian `llms.txt` pada 2026. Sekitar satu dari sebelas situs teratas menerbitkan berkas itu, tetapi pada Mei 2026 hampir tidak ada berkas yang menerima satu permintaan pun.

Angka adopsi itu dapat diperiksa dengan satu baris hitungan, kebiasaan yang layak dipelihara setiap kali persentase dan jumlah dikutip bersamaan: $93/1000 = 0.093 = 9.3\%$ dan $830/10\,000 = 0.083 = 8.3\%$. Demikian pula faktor pertumbuhannya: $36\,120/4\,088 \approx 8.84$.

Lembaga yang menjalankan pemindaian adopsi itu menyatakan batas datanya sendiri dalam satu kalimat yang layak dikutip utuh: menerbitkan berkas itu "tidak membuktikan peningkatan peringkat pencarian, kutipan AI, maupun izin pelatihan" [15].

10.6.3 Sikap Profitina: kebersihan murah, tanpa janji

Mengapa menerbitkan berkas yang hampir tidak pernah dibaca? Karena biayanya hampir nol, karena berkas itu menggambarkan situs dengan jujur, dan karena sebagian alat, seperti alat bantu pemrograman dan agen yang diarahkan seseorang ke sebuah situs, memang membacanya bila diminta. Yang tidak dilakukan Profitina adalah mengklaim bahwa berkas itu menyebabkan kutipan. Pengekangan diri itu sama dengan yang mengalir di seluruh produk: hasil visibilitas adalah "estimasi metodologis, bukan jaminan" [3], dan platform yang mengukur kesiapan sambil membesar-besarkan sebuah berkas akan membantah dirinya sendiri. Auditor di lab mencetak catatan yang sama di bawah setiap laporan.

> **WAWASAN KUNCI:** Bedakan *adopsi* (berapa yang menerbitkan) dari *pemakaian* (berapa yang dibaca), dan keduanya dari *dampak* (apakah hasilnya berubah). Untuk `llms.txt` pada September 2026, adopsinya sedang, pemakaiannya amat kecil, dan dampaknya belum terbukti. Sebaliknya, `robots.txt` berdampak setiap kali sebuah *crawler* datang.

10.7 Dua Lapisan Cache dan Urutan Purge

10.7.1 Mengapa *cache* ada

Cache adalah simpanan jawaban siap saji. Menyusun halaman WordPress berarti menjalankan PHP, menanyai basis data, dan merakit templat; menyajikan salinan tersimpan berarti membaca sebuah berkas. Di *host* kecil perbedaannya adalah antara puluhan milidetik dan ratusan, dan di tengah lonjakan lalu lintas perbedaannya adalah antara situs yang tetap berdiri dan situs yang tumbang. Jaringan penghantar konten menambahkan manfaat kedua: salinan yang disimpan di *edge* disajikan dari lokasi yang dekat dengan pengunjung, sehingga permintaan tidak perlu menyeberangi samudra.

Harga sebuah *cache* adalah bahwa salinannya dapat *basi*: origin sudah berubah, sedangkan salinannya belum. Seluruh subbab ini membahas cara mengendalikan harga itu.

10.7.2 Kesegaran dalam HTTP

Caching HTTP dispesifikasikan dalam RFC 9111 [18]. Sebuah respons disimpan bersama **usia**-nya (*age*), yaitu waktu sejak origin menghasilkannya, dan boleh disajikan tanpa bertanya kepada origin selama ia **segar**:

$$\text{segar} \Leftrightarrow \text{current_age} < \text{freshness_lifetime} \tag{10.3}$$

Masa hidup kesegaran (*freshness lifetime*) berasal dari *header* `Cache-Control` yang dikirim origin (atau suatu lapisan). **Tabel 10.3** mendaftar direktif yang penting bagi situs pemasaran.

Tabel 10.3 Direktif dan *header caching* HTTP yang dibaca auditor di lab, sebagaimana didefinisikan RFC 9111 untuk *cache* bersama.

Direktif	Arti bagi *cache* bersama (CDN, *cache host*)
`max-age=N`	Segar selama *N* detik; berlaku bagi setiap *cache*, termasuk peramban
`s-maxage=N`	Segar selama *N* detik hanya di *cache bersama*; di sana menimpa `max-age`
`no-cache`	Boleh disimpan, tetapi wajib divalidasi ulang ke origin sebelum setiap pemakaian
`no-store`	Sama sekali tidak boleh disimpan
`private`	Hanya peramban milik pengguna yang boleh menyimpannya; *cache* bersama tidak boleh
`public`	*Cache* bersama boleh menyimpannya walaupun biasanya tidak
`ETag`, `Last-Modified` (*header*)	*Validator*: memungkinkan *cache* bertanya "apakah sudah berubah?" dan menerima jawaban singkat 304
`Age` (*header*)	Jumlah detik yang sudah dihabiskan salinan itu di dalam *cache*

Jadi, bagi *cache* bersama, masa hidup L sebuah respons adalah

$$L = \begin{cases} s & \text{jika } \texttt{s-maxage} = s \text{ ada,} \\ m & \text{jika tidak, tetapi } \texttt{max-age} = m \text{ ada,} \\ \text{heuristik} & \text{jika tidak keduanya.} \end{cases} \tag{10.4}$$

Baris terakhir Persamaan (10.4) layak diperhatikan: respons tanpa `Cache-Control` sama sekali tidak berarti "jangan disimpan". RFC 9111 membolehkan *cache menebak* masa hidupnya, biasanya dari berapa lama sejak sumber daya itu terakhir diubah [18]. Situs yang tidak mengirim *header caching* telah menyerahkan keputusan itu kepada pihak lain.

Dua *header* yang lebih baru memperhalus gambaran ini. RFC 9213 mendefinisikan `CDN-Cache-Control`, *header* dengan sintaks yang sama dengan `Cache-Control` tetapi hanya dipatuhi *cache* CDN; bila *header* itu ada, *cache* CDN "WAJIB mengabaikan *field header* Cache-Control dan Expires" [19]. Dengan begitu origin dapat berkata "peramban: satu menit; CDN saya: satu jam" dalam satu respons. Selain itu, setiap lapisan biasanya melaporkan apa yang dilakukannya lewat *header*-nya sendiri: Cloudflare mengirim `CF-Cache-Status` dengan nilai seperti `HIT`, `MISS`, `EXPIRED`, `BYPASS`, dan `DYNAMIC` [20], sedangkan *plugin* LiteSpeed Cache mengirim `X-LiteSpeed-Cache: hit` atau `miss` [21].

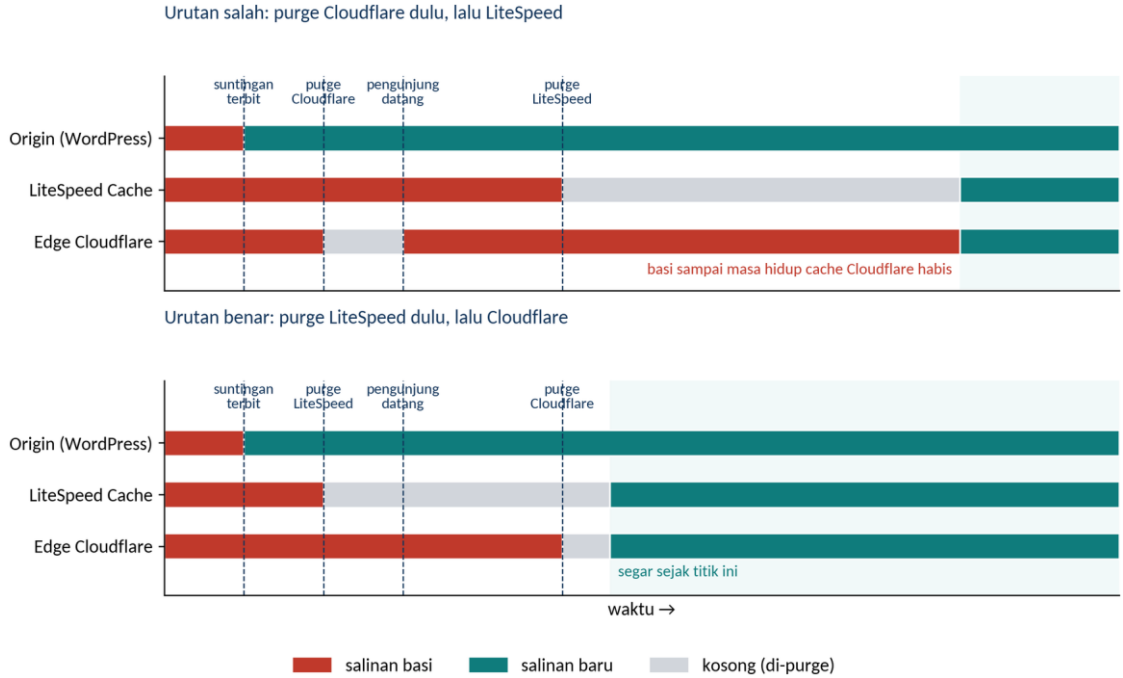
10.7.3 Apa yang di-*cache* setiap lapisan secara bawaan

Bawaan itu penting karena tidak ada orang yang mengatur segalanya. Dokumentasi Cloudflare menyatakan bahwa CDN-nya "tidak menyimpan HTML atau JSON secara bawaan", menyimpan daftar tetap ekstensi berkas statis seperti gambar, skrip, *style sheet*, dan fon, serta bahwa "secara bawaan Cloudflare menyimpan robots.txt sebuah situs web" [22]. Cloudflare juga tidak menyimpan respons yang `Cache-Control`-nya `private`, `no-store`, `no-cache`, atau `max-age=0` [22]. Situs yang ingin halaman HTML-nya disimpan di *edge* harus menyatakannya lewat sebuah aturan, dan begitu itu dilakukan, halaman-halaman tersebut tunduk pada risiko basi yang sama seperti hal lain.

Sebaliknya, *plugin* LiteSpeed Cache ada justru untuk menyimpan halaman WordPress di *host*, dan *plugin* itu mengosongkan salinannya secara otomatis ketika sebuah tulisan diterbitkan atau diperbarui [21]. Kedua lapisan itu karenanya punya kebiasaan yang berbeda: lapisan dalam mengenal WordPress dan mengosongkan dirinya sendiri ketika konten berubah; lapisan luar tidak tahu apa-apa tentang WordPress dan menyimpan apa yang diperintahkan kepadanya, selama yang diperintahkan.

10.7.4 Urutan *purge*

Setelah perubahan yang tidak ditangani *purge* otomatis, seperti versi tema baru atau perubahan `robots.txt`, kedua *cache* harus dikosongkan secara manual. Catatan penerapan tim Profitina menuliskan urutannya dengan huruf kapital: **LiteSpeed dulu, baru Cloudflare**. **Gambar 10.4** memperlihatkan alasannya.



Gambar 10.4 Mengosongkan dua lapisan *cache*. Dengan urutan yang salah, pengunjung yang datang di antara dua *purge* membuat *edge* mengisi dirinya kembali dari *cache* dalam yang masih basi, dan salinan basi itu hidup selama satu masa hidup *edge* penuh; dengan urutan yang benar, jendela basi berakhir pada *purge* kedua.

Bacalah panel atas terlebih dahulu. Suntingan diterbitkan di origin. Operator mengosongkan Cloudflare. Sebelum operator sempat mengosongkan LiteSpeed, seorang pengunjung datang. Cloudflare tidak punya apa-apa, maka ia bertanya ke *host*; LiteSpeed masih menyimpan halaman lama dan menyajikannya; Cloudflare menyimpan halaman lama itu sebagai salinan segar, dengan

masa hidup penuh di depannya. Kini operator mengosongkan LiteSpeed. *Host* sudah bersih, tetapi *edge* belum, dan *edge* akan terus menyajikan halaman lama sampai masa hidupnya habis. Mengosongkan dengan urutan ini bukan hanya tidak menolong: ia justru *memperbarui* salinan basi itu.

Pada panel bawah, LiteSpeed dikosongkan lebih dulu. Pengunjung yang datang di antara kedua *purge* masih mendapat halaman lama dari Cloudflare, yang belum dikosongkan, tetapi tidak ada yang baru tersimpan di mana pun, karena Cloudflare tidak perlu bertanya. Ketika Cloudflare dikosongkan, permintaan berikutnya luput di *edge*, luput di *host*, sampai ke WordPress, dan halaman baru mengisi kedua *cache* dalam perjalanan pulangnya.

Kita dapat memberi angka pada perbedaan itu. Misalkan suntingan terjadi pada waktu t_0, *purge* pertama pada t_1, *purge* kedua pada t_2, dan T_C adalah masa hidup halaman itu di *edge*. Dengan urutan benar, jendela basi berakhir pada *purge* kedua:

$$W_{\text{benar}} = t_2 - t_0 \tag{10.5}$$

Dengan urutan salah, sebuah permintaan pada waktu t_r mana pun dengan $t_1 < t_r < t_2$ mengisi kembali *edge* dengan salinan lama, yang kemudian hidup selama T_C:

$$W_{\text{salah}} = t_r + T_C - t_0 \;\leq\; t_2 + T_C - t_0 \tag{10.6}$$

CONTOH: Perbaikan `robots.txt` diterbitkan pada $t_0 = 0$. Operator mengosongkan satu *cache* pada $t_1 = 2$ menit dan yang lain pada $t_2 = 5$ menit. *Edge* menyimpan `robots.txt` selama $T_C = 4$ jam, yaitu 240 menit. Dengan urutan benar, Persamaan (10.5) memberikan $W = 5 - 0 = 5$ menit. Dengan urutan salah, *crawler* yang kebetulan bertanya pada $t_r = 3$ menit memberikan, menurut Persamaan (10.6), $W = 3 + 240 - 0 = 243$ menit, sedikit di atas empat jam. Tambahkan hingga 24 jam selama *crawler* boleh menyimpan salinan `robots.txt` miliknya sendiri (Subbab 10.4.6), dan kekeliruan urutan dua menit berubah menjadi lebih dari sehari aturan lama.

Permintaan di antara dua *purge* bukanlah peristiwa langka di situs publik. *Crawler*, pemantau *uptime*, dan bot pratinjau tautan meminta halaman terus-menerus. Di situs yang punya lalu lintas sedikit saja, panel atas adalah hasil yang biasa dari urutan yang salah, bukan nasib sial.

DEFINISI. Dua *cache* membentuk *lapisan* ketika salah satunya (lapisan luar) mengambil dari yang lain (lapisan dalam) saat luput. Aturan *urutan purge* untuk *cache* berlapis adalah: kosongkan dari dalam ke luar, lapisan yang paling dekat dengan origin lebih dulu dan lapisan yang paling dekat dengan pengunjung paling akhir.

10.7.5 Memilih masa hidup

Urutan *purge* membatasi kerusakan akibat perubahan; masa hidup yang wajar membatasi seberapa sering Anda perlu mengosongkan *cache*. Tiga pegangan praktis mengikuti pembahasan sejauh ini, dan auditor di lab memeriksa dua yang pertama:

- **Halaman HTML:** masa hidup dalam hitungan menit hingga jam, bukan hari, kecuali setiap penerbitan dapat diandalkan mengosongkan setiap lapisan. `s-maxage` sebulan pada HTML berarti *purge* yang terlewat menjadi kekeliruan selama sebulan.
- **`robots.txt`:** tidak lebih dari sehari di *cache* bersama mana pun, karena *crawler* sudah menyimpan salinannya sendiri hingga 24 jam [7], [8].
- **Validator:** kirim `ETag` atau `Last-Modified`, agar *cache* dapat memvalidasi ulang dengan murah alih-alih mengunduh seluruh halaman lagi.

Aset statis yang namanya berubah ketika isinya berubah, seperti `style.3f2a.css`, dapat disimpan selama setahun; itulah yang dicapai *cache busting* dalam proses rilis aplikasi di Bab 9.

10.8 Data Terstruktur di Situs Pemasaran

10.8.1 Apa yang dibawa halaman-halamannya

Bab 2 menjelaskan schema.org dan JSON-LD serta membangkitkan keduanya untuk Profitina Laundry. profitina.com memakai kosakata yang sama di halamannya sendiri: data terstruktur FAQ untuk pertanyaan yang sering diajukan dan tanggal `dateModified` pada kontennya, sehingga setiap program yang membaca sebuah halaman tahu apa halaman itu, pertanyaan apa yang dijawabnya, dan seberapa baru isinya. Kosakata itu sendiri terus berkembang; rilis schema.org terbaru saat buku ini disusun adalah versi 30.1, tertanggal 16 September 2026 [23].

Tiga jenis simpul paling penting bagi situs semacam ini:

- simpul **organisasi** (`Organization`, atau `LocalBusiness` yang lebih spesifik untuk toko), setidaknya dengan `name` dan `url`, agar entitas di balik situs itu tidak ambigu;
- simpul **FAQ** (`FAQPage` dengan `Question` dan `Answer`), yang mengemas pernyataan-pernyataan pendek, mandiri, dan siap kutip;
- simpul **halaman** (`WebPage` atau `Article`) dengan `dateModified`, yang memberi tahu pembaca seberapa segar fakta-faktanya.

10.8.2 Setelah Mei 2026: *markup* FAQ tanpa hasil FAQ

Selama bertahun-tahun, banyak situs menambahkan *markup* `FAQPage` terutama karena Google menampilkan *rich result* FAQ, pertanyaan yang dapat dibuka di bawah hasil pencarian. Insentif itu sudah hilang. Google berhenti menampilkan *rich result* FAQ pada 7 Mei 2026, disusul penghentian dukungan pelaporan dan pengujian pada Juni dan dukungan API pada Agustus 2026 [24]. Google juga menyatakan bahwa *markup* itu boleh tetap dipasang: “data terstruktur yang tidak dipakai tidak menimbulkan masalah bagi Search, dan FAQPage tetap merupakan tipe Schema.org yang sah” [24].

Bagi GEO, penalarannya tidak pernah bergantung pada fitur visual satu mesin pencari. Pertanyaan dengan jawaban yang eksplisit dan pendek adalah jenis bagian teks yang menurut studi GEO di balik Bab 2 paling mungkin dipakai dalam jawaban yang dibangkitkan, dengan penambahan kutipan, petikan, dan statistik menaikkan visibilitas sebuah sumber hingga sekitar 40% [25]. Data terstruktur membuat bagian teks semacam itu tidak ambigu bagi mesin. Data terstruktur tidak mewajibkan mesin mana pun mengutipnya, dan Profitina tidak pernah mengatakan demikian.

10.8.3 Kesegaran termasuk bagian data

Properti `dateModified` menghubungkan data terstruktur dengan kisah *caching* di Subbab 10.7. Halaman yang di-*cache* membawa `dateModified` pada saat ia disimpan. Bila salinan basi bertahan di *edge*, tanggal di dalamnya pun bertahan, dan program yang membaca halaman itu diberi tahu, dengan jujur sejauh menyangkut salinan itu, bahwa tidak ada yang berubah sejak saat itu. Lab ini memanfaatkan hal tersebut: ia membandingkan `dateModified` halaman yang tayang dengan `dateModified` halaman di origin, dan perbedaannya adalah bukti *cache* basi yang tidak dapat disembunyikan *header* mana pun.

10.8.4 Tiga kesalahan yang membuat data terstruktur tak berguna

Data terstruktur gagal tanpa suara, seperti `robots.txt`. Peramban tidak pernah menampilkannya, sehingga blok yang rusak tidak disadari sampai ada yang memeriksanya. Tiga kesalahan yang dicari lab adalah yang paling umum:

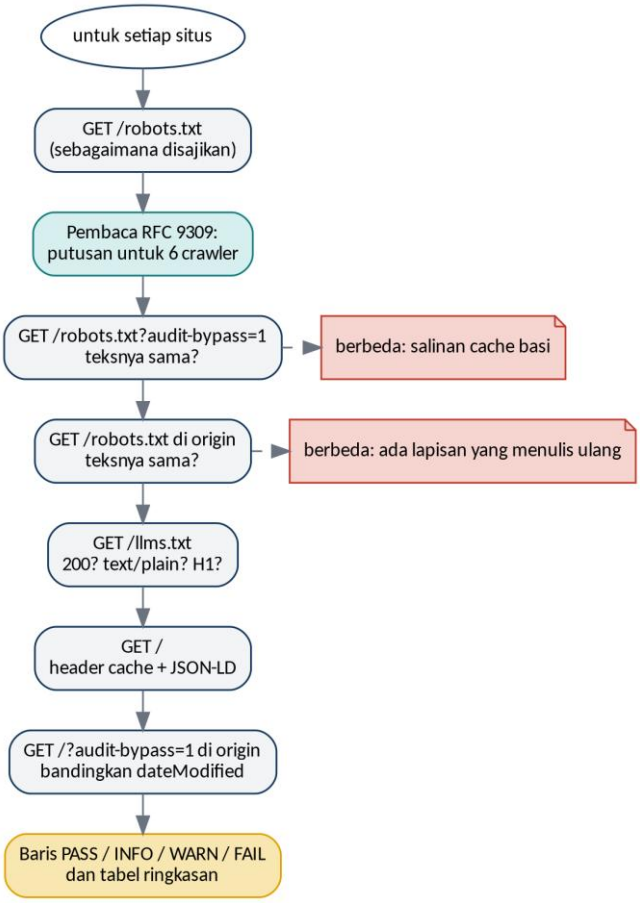
1. **JSON yang tidak sah.** Satu koma di akhir, satu tanda kutip yang tidak di-*escape*, dan seluruh blok ditolak pengurai yang ketat. Situs contoh `broken` di lab memiliki koma di akhir.
2. **Identitas yang hilang.** Organisasi tanpa `url` adalah organisasi yang tidak dapat ditautkan dengan pasti ke situs webnya.
3. **Tanggal yang basi.** `dateModified` yang berbeda antara halaman yang tayang dan origin (Subbab 10.8.3).

Keabsahan hanyalah lantai. Aturan Bab 2, bahwa data terstruktur harus menyatakan persis apa yang dinyatakan halaman yang terlihat, tetap berlaku, dan tidak ada pemeriksaan otomatis yang dapat memastikannya; seseorang yang membaca keduanya dapat.

10.9 Lab: Auditor Kebersihan GEO dalam Python dan Java

10.9.1 Apa yang akan kita bangun

Lab ini mengubah daftar periksa Subbab 10.5.5, aturan *caching* Subbab 10.7, dan kesalahan data terstruktur Subbab 10.8.4 menjadi sebuah program. Program itu adalah *auditor*: diberi alamat sebuah situs, ia mengambil segelintir berkas publik, menerapkan aturan-aturan bab ini, dan mencetak satu baris untuk setiap temuan, masing-masing bertanda `PASS`, `INFO`, `WARN`, atau `FAIL`, diikuti tabel ringkasan. **Gambar 10.5** memperlihatkan urutan permintaannya.



Gambar 10.5 Permintaan-permintaan auditor untuk satu situs. Dua perbandingan membedakan salinan *cache* yang basi dari berkas yang ditulis ulang oleh suatu lapisan; permintaan terakhir mengambil halaman beranda yang segar untuk membandingkan `dateModified`.

Program yang mengambil halaman milik orang lain harus santun, dan auditor ini santun: ia membuat satu permintaan per sumber daya, menunggu jawaban paling lama lima detik, dan menyebutkan jati dirinya di *header* `User-Agent`. Ia tidak pernah dijalankan secara massal terhadap situs pihak ketiga. Agar ada sesuatu yang realistis untuk diaudit tanpa menyentuh server siapa pun, lab ini menyertakan empat **situs contoh** kecil dan sebuah server yang menjalankannya di komputer Anda sendiri, pada alamat `127.0.0.1`, yang tidak dapat dijangkau komputer lain. **Tabel 10.4** memperkenalkan keempatnya.

Tabel 10.4 Empat situs contoh di lab. Dua di antaranya juga membuka origin-nya di *port* kedua, sehingga auditor dapat membandingkan apa yang diterima pengunjung dengan apa yang disimpan origin.

Situs	*Port* publik	*Port* origin	Apa yang salah dengannya (disengaja)
`laundry`	7100	—	Tidak ada: Profitina Laundry, dikonfigurasi seperti yang dianjurkan bab ini
`cdnblock`	7101	7105	Lapisan *edge* menyisipkan blok terkelola di depan `robots.txt` (Subbab 10.5)
`stale`	7102	7106	HTML disimpan sebulan; beranda di *cache* sudah usang; berkas yang tidak ada dijawab dengan beranda berstatus 200
`broken`	7103	—	`robots.txt` menjawab 500; satu blok JSON-LD bukan JSON yang sah; organisasinya tidak punya `url`

Seluruh kode hanya memakai pustaka standar masing-masing bahasa. Versi Python memerlukan Python 3.11 atau yang lebih baru dan ditulis untuk 3.14; versi Java dikompilasi dengan JDK 21

dan berjalan di JDK 25. Susunan folder, perintah untuk Windows, macOS, dan Linux, serta pengaturan VS Code ada di `README.md` bab ini:

```
Chapter10/
  sites/     laundry/  cdnblock/  stale/  broken/   (shared by both languages)
  python/    site_server.py  robots_rules.py  auditor.py  robots_quirk.py
  java/      SiteServer.java  RobotsRules.java  MiniJson.java  Auditor.java
```

10.9.2 Situs-situs contoh dan servernya

Setiap situs contoh adalah folder berisi berkas-berkas biasa ditambah satu berkas konfigurasi kecil, `_site.conf`. Inilah berkas untuk `stale`:

```
# HTML cached for a month at the edge; the cached home page is out of date.
# Missing files come back as the home page with status 200 (a "soft 404").
port = 7102
origin_port = 7106
soft404 = yes
header * = Cache-Control: public, max-age=604800, s-maxage=2592000
header /robots.txt = Cache-Control: public, max-age=604800
```

Baris `header PATH = Nama: nilai` menambahkan *header* respons untuk satu jalur, atau untuk setiap jalur bila `PATH` adalah `*`; baris yang lebih spesifik menimpa baris `*` untuk *header* yang sama. Baris `status PATH = KODE` memaksakan kode status; begitulah `broken` membuat `robots.txt`-nya menjawab 500. `etag = yes` membuat server mengirim `ETag` yang dihitung dari isi berkas.

Dua subfolder meniru lapisan-lapisan di **Gambar 10.1**. Berkas di `_edge/` adalah apa yang disajikan lapisan *edge sebagai pengganti* berkas origin yang bernama sama, pada setiap permintaan ke *port* publik; begitulah `cdnblock` mendapat blok terkelolanya. Berkas di `_cache/` adalah salinan basi yang masih disimpan sebuah *cache*; berkas itu disajikan di *port* publik hanya bila URL-nya *tidak* memiliki *query string*, dengan *header* `X-Cache: HIT` dan `Age: 5400`, persis seperti perilaku *cache* nyata yang berkuncikan URL lengkap. Permintaan apa pun yang memiliki *query string* luput dari *cache* dan dijawab dari origin dengan `X-Cache: MISS`. Situs yang memiliki `origin_port` juga menjawab di *port* kedua itu dengan berkas-berkas origin yang polos, tidak pernah dengan salinan `_edge` atau `_cache`.

Kode 10.1 adalah server versi Python. Server itu membaca konfigurasi setiap situs, membangun *handler* permintaan untuk setiap lapisan, dan menjalankan satu `ThreadingHTTPServer` per *port*, masing-masing di *thread* latar belakang.

Kode 10.1 `site_server.py`: menyajikan situs-situs contoh, meniru lapisan *edge* (`_edge/`) dan *cache* basi (`_cache/`) di depan setiap origin.

```
"""Serve the sample sites of Chapter 10 on 127.0.0.1 (ports 7100-7109).

Each folder under sites/ is one site. Its _site.conf sets the port, extra
headers and status codes. Files in _edge/ are what an edge layer (a CDN)
serves instead of the origin file; files in _cache/ are stale cached copies,
served only when the URL has no query string.
usage: python site_server.py [sites_dir]
"""
import hashlib
import os
import sys
import threading
from http.server import BaseHTTPRequestHandler, ThreadingHTTPServer
from urllib.parse import urlsplit

TYPES = {".html": "text/html; charset=utf-8",
         ".txt": "text/plain; charset=utf-8",
         ".xml": "application/xml"}
CACHED_AGE = "5400"            # seconds a cached copy has sat at the edge


def read_conf(site_dir):
    conf = {"port": None, "origin_port": None, "etag": False,
            "soft404": False, "headers": [], "status": {}}
    with open(os.path.join(site_dir, "_site.conf"), encoding="utf-8") as f:
        for raw in f:
            line = raw.strip()
            if not line or line.startswith("#"):
                continue
            key, value = [s.strip() for s in line.split("=", 1)]
            if key.startswith("header "):
                name, val = value.split(":", 1)
                conf["headers"].append((key[7:].strip(), name, val.strip()))
            elif key.startswith("status "):
                conf["status"][key[7:].strip()] = int(value)
            elif key in ("port", "origin_port"):
                conf[key] = int(value)
            else:
                conf[key] = (value == "yes")
    return conf


def make_handler(site_dir, conf, layer):
    two_layers = conf["origin_port"] is not None
```

Kode 10.1 (Lanjutan) *site_server.py: menyajikan situs-situs contoh, meniru lapisan edge (_edge/) dan cache basi (_cache/) di depan setiap origin.*

```
    class Handler(BaseHTTPRequestHandler):
        def log_message(self, *args):          # keep the terminal quiet
            pass

        def do_GET(self):
            parts = urlsplit(self.path)
            path = parts.path
            name = "index.html" if path == "/" else path.lstrip("/")
            if name.startswith("_") or ".." in name:
                return self.reply(404, "text/plain", b"Not Found", path, {})
            if path in conf["status"]:
                code = conf["status"][path]
                return self.reply(code, "text/plain", b"Server Error", path, {})
            extra, file = {}, os.path.join(site_dir, name)
            edge = os.path.join(site_dir, "_edge", name)
            cached = os.path.join(site_dir, "_cache", name)
            if layer == "edge" and os.path.isfile(edge):
                file = edge                    # the edge rewrites this file
            elif layer == "edge" and not parts.query and os.path.isfile(cached):
                file = cached                  # a stale copy in the cache
                extra = {"X-Cache": "HIT", "Age": CACHED_AGE}
            elif layer == "edge" and two_layers:
                extra = {"X-Cache": "MISS"}
            if not os.path.isfile(file):
                if not conf["soft404"]:
                    return self.reply(404, "text/plain", b"Not Found", path, {})
                file = os.path.join(site_dir, "index.html")   # "soft 404"
            with open(file, "rb") as f:
                body = f.read()
            ctype = TYPES.get(os.path.splitext(file)[1], "text/plain")
            self.reply(200, ctype, body, path, extra)

        def do_HEAD(self):                     # same headers, no body
            self.do_GET()

        def reply(self, code, ctype, body, path, extra):
            headers = {"Content-Type": ctype}
            for pattern, hname, hval in conf["headers"]:
                if pattern == "*" or pattern == path:
                    headers[hname] = hval      # a path rule overrides "*"
            if conf["etag"] and code == 200:
                digest = hashlib.sha256(body).hexdigest()[:16]
                headers["ETag"] = '"' + digest + '"'
            headers.update(extra)
            headers["Content-Length"] = str(len(body))
            self.send_response(code)
```

Kode 10.1 (Lanjutan) *site_server.py: menyajikan situs-situs contoh, meniru lapisan edge (_edge/) dan cache basi (_cache/) di depan setiap origin.*

```
            for hname, hval in headers.items():
                self.send_header(hname, hval)
            self.end_headers()
            if self.command != "HEAD":
                self.wfile.write(body)

    return Handler


def start_all(sites_dir):
    """Start every site (and its origin, if any); return [(label, server)]."""
    started = []
    for site in sorted(os.listdir(sites_dir)):
        site_dir = os.path.join(sites_dir, site)
        if not os.path.isfile(os.path.join(site_dir, "_site.conf")):
            continue
        conf = read_conf(site_dir)
        layers = [("edge", conf["port"])]
        if conf["origin_port"] is not None:
            layers.append(("origin", conf["origin_port"]))
        for layer, port in layers:
            handler = make_handler(site_dir, conf, layer)
            server = ThreadingHTTPServer(("127.0.0.1", port), handler)
            threading.Thread(target=server.serve_forever, daemon=True).start()
            started.append((site + " " + layer, server))
    return started


def main():
    sites_dir = sys.argv[1] if len(sys.argv) > 1 else "../sites"
    servers = start_all(sites_dir)
    for label, server in servers:
        print("%-16s http://127.0.0.1:%d/" % (label, server.server_port))
    print("Serving; press Ctrl+C to stop.")
    try:
        threading.Event().wait()
    except KeyboardInterrupt:
        for _, server in servers:
            server.shutdown()


if __name__ == "__main__":
    main()
```

Jantung server ini adalah keputusan di `do_GET`, dan layak dibaca sesuai urutan pengujiannya dalam kode. Jalur yang diawali garis bawah atau mengandung `..` ditolak, sehingga pengunjung tidak pernah dapat membaca `_site.conf` atau keluar dari folder situs. Status paksaan diperiksa berikutnya. Lalu, hanya di *port* publik, berkas `_edge` mengalahkan segalanya; berkas `_cache` menang bila URL tidak memiliki *query string*; dan situs dua lapis menandai setiap jawaban lainnya sebagai `MISS`. Terakhir, berkas yang tidak ada dijawab dengan 404 biasa atau, bila `soft404 = yes`, dengan beranda berstatus 200, perilaku banyak sistem manajemen konten yang salah konfigurasi dan jebakan bagi siapa pun yang hanya memeriksa kode status.

Kode 10.2 adalah server yang sama dalam Java, dibangun di atas `com.sun.net.httpserver.HttpServer` milik JDK. Logikanya sama baris demi baris; tambahannya hanya kelas `Conf` yang eksplisit untuk pengaturan dan pabrik *thread* yang menandai *thread* server sebagai *thread daemon*, agar Java Virtual Machine dapat berhenti begitu `main` selesai.

Kode 10.2 `SiteServer.java`: server situs contoh yang sama di atas server HTTP bawaan JDK.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.security.MessageDigest;
import java.util.*;
import java.util.concurrent.Executors;

/** Serves the sample sites of Chapter 10 on 127.0.0.1 (ports 7100-7109).
 *  Files in _edge/ are what an edge layer serves instead of the origin file;
 *  files in _cache/ are stale copies, served only without a query string.
 *  usage: java SiteServer [sites_dir] */
public class SiteServer {
    static final String CACHED_AGE = "5400";   // seconds at the edge

    /** The settings read from one site's _site.conf. */
    static class Conf {
        Integer port, originPort;
        boolean etag, soft404;
        List<String[]> headers = new ArrayList<>();   // {pattern, name, value}
        Map<String, Integer> status = new HashMap<>();
    }

    static Conf readConf(Path siteDir) throws IOException {
        Conf c = new Conf();
        for (String raw : Files.readAllLines(siteDir.resolve("_site.conf"))) {
            String line = raw.strip();
            if (line.isEmpty() || line.startsWith("#")) continue;
            int eq = line.indexOf('=');
            String key = line.substring(0, eq).strip();
            String value = line.substring(eq + 1).strip();
            if (key.startsWith("header ")) {
                int colon = value.indexOf(':');
                c.headers.add(new String[] {key.substring(7).strip(),
                        value.substring(0, colon),
                        value.substring(colon + 1).strip()});
            } else if (key.startsWith("status ")) {
                c.status.put(key.substring(7).strip(), Integer.parseInt(value));
            } else if (key.equals("port")) {
                c.port = Integer.parseInt(value);
            } else if (key.equals("origin_port")) {
                c.originPort = Integer.parseInt(value);
```

Kode 10.2 (Lanjutan) *SiteServer.java: server situs contoh yang sama di atas server HTTP bawaan JDK.*

```
            } else if (key.equals("etag")) {
                c.etag = value.equals("yes");
            } else if (key.equals("soft404")) {
                c.soft404 = value.equals("yes");
            }
        }
        return c;
    }

    static void handle(HttpExchange ex, Path dir, Conf conf, String layer)
            throws IOException {
        String path = ex.getRequestURI().getPath();
        String query = ex.getRequestURI().getRawQuery();
        String name = path.equals("/") ? "index.html" : path.substring(1);
        if (name.startsWith("_") || name.contains("..")) {
            reply(ex, 404, "text/plain", bytes("Not Found"), path, conf, Map.of());
            return;
        }
        if (conf.status.containsKey(path)) {
            reply(ex, conf.status.get(path), "text/plain", bytes("Server Error"),
                  path, conf, Map.of());
            return;
        }
        Map<String, String> extra = new LinkedHashMap<>();
        Path file = dir.resolve(name);
        Path edge = dir.resolve("_edge").resolve(name);
        Path cached = dir.resolve("_cache").resolve(name);
        boolean atEdge = layer.equals("edge");
        if (atEdge && Files.isRegularFile(edge)) {
            file = edge;                                  // the edge rewrites it
        } else if (atEdge && query == null && Files.isRegularFile(cached)) {
            file = cached;                                // a stale cached copy
            extra.put("X-Cache", "HIT");
            extra.put("Age", CACHED_AGE);
        } else if (atEdge && conf.originPort != null) {
            extra.put("X-Cache", "MISS");
        }
        if (!Files.isRegularFile(file)) {
            if (!conf.soft404) {
                reply(ex, 404, "text/plain", bytes("Not Found"), path, conf,
                      Map.of());
                return;
            }
            file = dir.resolve("index.html");             // a "soft 404"
        }
        String f = file.toString();
```

Kode 10.2 (Lanjutan) *SiteServer.java: server situs contoh yang sama di atas server HTTP bawaan JDK.*

```
        String type = f.endsWith(".html") ? "text/html; charset=utf-8"
                : f.endsWith(".xml") ? "application/xml"
                : "text/plain; charset=utf-8";
        reply(ex, 200, type, Files.readAllBytes(file), path, conf, extra);
    }

    static void reply(HttpExchange ex, int code, String type, byte[] body,
                      String path, Conf conf, Map<String, String> extra)
            throws IOException {
        Map<String, String> h = new LinkedHashMap<>();
        h.put("Content-Type", type);
        for (String[] rule : conf.headers) {
            if (rule[0].equals("*") || rule[0].equals(path)) {
                h.put(rule[1], rule[2]);              // a path rule overrides *
            }
        }
        if (conf.etag && code == 200) {
            h.put("ETag", "\"" + sha256(body).substring(0, 16) + "\"");
        }
        h.putAll(extra);
        h.forEach((k, v) -> ex.getResponseHeaders().set(k, v));
        if (ex.getRequestMethod().equals("HEAD")) {   // same headers, no body
            ex.sendResponseHeaders(code, -1);
            return;
        }
        ex.sendResponseHeaders(code, body.length);
        try (OutputStream out = ex.getResponseBody()) {
            out.write(body);
        }
    }

    static byte[] bytes(String s) {
        return s.getBytes(StandardCharsets.UTF_8);
    }

    static String sha256(byte[] data) {
        try {
            byte[] d = MessageDigest.getInstance("SHA-256").digest(data);
            return HexFormat.of().formatHex(d);
        } catch (Exception e) {
            throw new IllegalStateException(e);
        }
    }
```

Kode 10.2 (Lanjutan) *SiteServer.java: server situs contoh yang sama di atas server HTTP bawaan JDK.*

```
    /** Starts every site (and its origin, if any); returns label -> server. */
    static Map<String, HttpServer> startAll(Path sitesDir) throws IOException {
        Map<String, HttpServer> started = new LinkedHashMap<>();
        List<Path> sites = new ArrayList<>();
        try (var list = Files.list(sitesDir)) {
            list.filter(p -> Files.isRegularFile(p.resolve("_site.conf")))
                .sorted().forEach(sites::add);
        }
        for (Path dir : sites) {
            Conf conf = readConf(dir);
            Map<String, Integer> layers = new LinkedHashMap<>();
            layers.put("edge", conf.port);
            if (conf.originPort != null) layers.put("origin", conf.originPort);
            for (var layer : layers.entrySet()) {
                HttpServer server = HttpServer.create(
                        new InetSocketAddress("127.0.0.1", layer.getValue()), 0);
                server.createContext("/", ex -> {
                    try {
                        handle(ex, dir, conf, layer.getKey());
                    } finally {
                        ex.close();
                    }
                });
                server.setExecutor(Executors.newCachedThreadPool(r -> {
                    Thread t = new Thread(r);
                    t.setDaemon(true);              // never keeps the JVM alive
                    return t;
                }));
                server.start();
                started.put(dir.getFileName() + " " + layer.getKey(), server);
            }
        }
        return started;
    }

    public static void main(String[] args) throws Exception {
        Path sitesDir = Path.of(args.length > 0 ? args[0] : "../sites");
        Map<String, HttpServer> servers = startAll(sitesDir);
        servers.forEach((label, s) -> System.out.printf(
                "%-16s http://127.0.0.1:%d/%n", label, s.getAddress().getPort()));
        System.out.println("Serving; press Ctrl+C to stop.");
        Thread.currentThread().join();              // until Ctrl+C
    }
}
```

Jalankan server Python di satu terminal dengan `python site_server.py ../sites` (di Windows, `py site_server.py ..\sites`). Server mencetak alamat setiap lapisan yang dijalankannya:

```
broken edge      http://127.0.0.1:7103/
cdnblock edge    http://127.0.0.1:7101/
cdnblock origin  http://127.0.0.1:7105/
laundry edge     http://127.0.0.1:7100/
stale edge       http://127.0.0.1:7102/
stale origin     http://127.0.0.1:7106/
Serving; press Ctrl+C to stop.
```

Di terminal kedua, `curl -i` menampilkan respons beserta *header*-nya. Pertama `robots.txt` publik milik `cdnblock`, lalu berkas yang sama di origin-nya:

```
$ curl -i http://127.0.0.1:7101/robots.txt
HTTP/1.0 200 OK
Server: BaseHTTP/0.6 Python/3.11.15
Date: Thu, 24 Sep 2026 04:56:58 GMT
Content-Type: text/plain; charset=utf-8
Cache-Control: public, max-age=14400
ETag: "a75bf8f54f3e4133"
Content-Length: 294

# BEGIN block managed by the edge layer
User-agent: GPTBot
Disallow: /

User-agent: ClaudeBot
Disallow: /

User-agent: Google-Extended
Disallow: /
# END block managed by the edge layer

# Origin copy, written by the site owner
User-agent: *
Allow: /

Sitemap: http://127.0.0.1:7101/sitemap.xml

$ curl -i http://127.0.0.1:7105/robots.txt
HTTP/1.0 200 OK
Server: BaseHTTP/0.6 Python/3.11.15
Date: Thu, 24 Sep 2026 04:56:58 GMT
Content-Type: text/plain; charset=utf-8
Cache-Control: public, max-age=14400
ETag: "692d92fde37dbec1"
Content-Length: 108

# Origin copy, written by the site owner
User-agent: *
Allow: /

Sitemap: http://127.0.0.1:7101/sitemap.xml
```

Inilah Subbab 10.5 dalam bentuk mini: berkas milik pemilik situs tidak bersalah, dan berkas yang diterima dunia memblokir tiga *crawler*. Situs `stale` memperlihatkan kesalahan yang lain. `curl -I` hanya meminta *header*; tanpa *query string* yang menjawab adalah *cache*, dan dengan *query string* yang menjawab adalah origin:

```
$ curl -I http://127.0.0.1:7102/
HTTP/1.0 200 OK
Server: BaseHTTP/0.6 Python/3.11.15
Date: Thu, 24 Sep 2026 04:56:58 GMT
Content-Type: text/html; charset=utf-8
Cache-Control: public, max-age=604800, s-maxage=2592000
X-Cache: HIT
Age: 5400
Content-Length: 511

$ curl -I "http://127.0.0.1:7102/?nocache=1"
HTTP/1.0 200 OK
Server: BaseHTTP/0.6 Python/3.11.15
Date: Thu, 24 Sep 2026 04:56:58 GMT
Content-Type: text/html; charset=utf-8
Cache-Control: public, max-age=604800, s-maxage=2592000
X-Cache: MISS
Content-Length: 517
```

Kedua panjang itu, 511 dan 517 bait, sudah memberi tahu kita bahwa kedua jawaban adalah halaman yang berbeda. Tekan Ctrl+C di terminal pertama untuk menghentikan server; auditor di bawah tidak memerlukannya, karena auditor dapat menjalankan situs-situs itu sendiri.

10.9.3 Pembaca robots.txt yang mengikuti RFC 9309

Kelas `Robots` di Bab 2 menerapkan pemilihan grup dan aturan kecocokan terpanjang tanpa karakter pengganti. Bab ini memerlukan sisa RFC 9309: grup untuk *crawler* yang sama digabungkan, grup `*` sebagai cadangan murni, serta karakter pengganti `*` dan `$`. **Kode 10.3** mengerjakannya dalam sekitar enam puluh baris.

Kode 10.3 `robots_rules.py`: pembaca yang menerapkan Persamaan (10.1) (penggabungan, kembali ke `*`) dan Persamaan (10.2) (kecocokan terpanjang, `Allow` bila seri), lengkap dengan karakter pengganti `*` dan `$`.

```
"""A small robots.txt reader that follows RFC 9309.

Groups that name the same crawler are merged; a crawler with no group of
its own obeys the "*" group; the longest matching path wins, and on a tie
"allow" wins. Python's urllib.robotparser uses the first match instead.
"""
import re


class Robots:
    def __init__(self, text):
        self.groups = []           # [(set of agents, [(allow, path), ...])]
        self.sitemaps = []
        self.signals = []          # Content-Signal lines (not in RFC 9309)
        agents, rules, last_was_agent = None, None, False
        for raw in text.splitlines():
            line = raw.split("#", 1)[0].strip()
            if ":" not in line:
                continue
            key, value = line.split(":", 1)
            key, value = key.strip().lower(), value.strip()
            if key == "user-agent":
                if not last_was_agent:      # a new group starts here
                    agents, rules = set(), []
                    self.groups.append((agents, rules))
                agents.add(value.lower())
                last_was_agent = True
                continue
            last_was_agent = False
            if key in ("allow", "disallow") and rules is not None and value:
                rules.append((key == "allow", value))
            elif key == "sitemap":
                self.sitemaps.append(value)
            elif key == "content-signal":
                self.signals.append(value)

    def rules_for(self, agent):
        agent = agent.lower()
        mine = [r for agents, rules in self.groups if agent in agents
                for r in rules]
        if mine:
            return mine
        return [r for agents, rules in self.groups if "*" in agents
                for r in rules]
```

Kode 10.3 (Lanjutan) *robots_rules.py: pembaca yang menerapkan Persamaan (10.1) (penggabungan, kembali ke *) dan Persamaan (10.2) (kecocokan terpanjang, Allow bila seri), lengkap dengan karakter pengganti * dan $.*

```
    def allowed(self, agent, path="/"):
        best_len, best_allow = -1, True     # no matching rule: allowed
        for allow, pattern in self.rules_for(agent):
            if matches(pattern, path):
                n = len(pattern.encode("utf-8"))
                if n > best_len or (n == best_len and allow):
                    best_len, best_allow = n, allow
        return best_allow


def matches(pattern, path):
    """'*' matches any run of characters; a final '$' anchors the end."""
    anchored = pattern.endswith("$")
    body = pattern[:-1] if anchored else pattern
    regex = ".*".join(re.escape(part) for part in body.split("*"))
    return re.match(regex + ("$" if anchored else ""), path) is not None
```

Tiga rincian layak dikomentari. Konstruktornya menyimpan penanda `last_was_agent`, karena baris `User-agent` hanya memulai grup baru bila baris bermakna sebelumnya *bukan* baris `User-agent` juga; begitulah lima baris `User-agent` yang berurutan menjadi satu grup, seperti di Subbab 10.4.2. Metode `rules_for` adalah Persamaan (10.1) hampir kata demi kata: kumpulkan aturan setiap grup yang menyebut *crawler* itu, dan hanya bila tidak ada, kumpulkan aturan grup-grup `*`. Dan `allowed` adalah Persamaan (10.2): metode itu menelusuri aturan-aturan, menyimpan pasangan terbaik (panjang, apakah-allow) sejauh ini, dan menggantinya ketika pola yang lebih panjang cocok atau ketika `Allow` yang sama panjang cocok. Baris `Disallow:` yang kosong, yang dalam `robots.txt` berarti "tidak ada yang dilarang", tidak ditambahkan sebagai aturan.

Fungsi `matches` menangani karakter pengganti. `Disallow: /*.pdf$` menjadi ekspresi reguler `/.*\.pdf$`, dan `re.match` menambatkannya di awal jalur, karena setiap pola dalam `robots.txt` dicocokkan dari awal. **Kode 10.4** adalah versi Java-nya, dengan *record* untuk aturan dan grup.

Kode 10.4 `RobotsRules.java`: pembaca RFC 9309 yang sama dalam Java.

```java
import java.nio.charset.StandardCharsets;
import java.util.*;
import java.util.regex.Pattern;

/** A small robots.txt reader that follows RFC 9309. Groups naming the same
 *  crawler are merged; a crawler with no group of its own obeys "*"; the
 *  longest matching path wins, and on a tie "allow" wins. */
public class RobotsRules {
    record Rule(boolean allow, String path) {}
    record Group(Set<String> agents, List<Rule> rules) {}

    final List<Group> groups = new ArrayList<>();
    final List<String> sitemaps = new ArrayList<>();
    final List<String> signals = new ArrayList<>();   // Content-Signal lines

    RobotsRules(String text) {
        Group current = null;
        boolean lastWasAgent = false;
        for (String raw : text.split("\r?\n")) {
            String line = raw.split("#", 2)[0].strip();
            int colon = line.indexOf(':');
            if (colon < 0) continue;
            String key = line.substring(0, colon).strip().toLowerCase();
            String value = line.substring(colon + 1).strip();
            if (key.equals("user-agent")) {
                if (!lastWasAgent) {                  // a new group starts
                    current = new Group(new HashSet<>(), new ArrayList<>());
                    groups.add(current);
                }
                current.agents().add(value.toLowerCase());
                lastWasAgent = true;
                continue;
            }
            lastWasAgent = false;
            if ((key.equals("allow") || key.equals("disallow"))
                    && current != null && !value.isEmpty()) {
                current.rules().add(new Rule(key.equals("allow"), value));
            } else if (key.equals("sitemap")) {
                sitemaps.add(value);
            } else if (key.equals("content-signal")) {
                signals.add(value);
            }
        }
    }
```

Kode 10.4 (Lanjutan) *RobotsRules.java: pembaca RFC 9309 yang sama dalam Java.*

```
    List<Rule> rulesFor(String agent) {
        String a = agent.toLowerCase();
        List<Rule> mine = new ArrayList<>();
        for (Group g : groups) if (g.agents().contains(a)) mine.addAll(g.rules());
        if (!mine.isEmpty()) return mine;
        for (Group g : groups) if (g.agents().contains("*")) mine.addAll(g.rules());
        return mine;
    }

    boolean allowed(String agent, String path) {
        int bestLen = -1;
        boolean bestAllow = true;                       // no rule: allowed
        for (Rule r : rulesFor(agent)) {
            if (matches(r.path(), path)) {
                int n = r.path().getBytes(StandardCharsets.UTF_8).length;
                if (n > bestLen || (n == bestLen && r.allow())) {
                    bestLen = n;
                    bestAllow = r.allow();
                }
            }
        }
        return bestAllow;
    }

    boolean allowed(String agent) {
        return allowed(agent, "/");
    }

    /** '*' matches any run of characters; a final '$' anchors the end. */
    static boolean matches(String pattern, String path) {
        boolean anchored = pattern.endsWith("$");
        String body = anchored ? pattern.substring(0, pattern.length() - 1)
                               : pattern;
        StringBuilder regex = new StringBuilder();
        String[] parts = body.split("\\*", -1);
        for (int i = 0; i < parts.length; i++) {
            if (i > 0) regex.append(".*");
            regex.append(Pattern.quote(parts[i]));
        }
        if (!anchored) regex.append(".*");
        return Pattern.compile(regex.toString(), Pattern.DOTALL)
                      .matcher(path).matches();
    }
}
```

10.9.4 Auditornya

Kode 10.5 adalah auditornya. Inilah program terpanjang di bab ini, tetapi ia hanyalah rangkaian pemeriksaan kecil yang saling lepas, masing-masing berupa fungsi yang mengambil apa yang diperlukannya dan mencetak temuannya melalui objek `Report` yang mengingat status terburuk setiap kategori.

Kode 10.5 `auditor.py`: memeriksa `robots.txt`, `llms.txt`, *header cache*, dan JSON-LD setiap situs, lalu mencetak laporan dan tabel ringkasan.

```
"""Audit robots.txt, llms.txt, cache headers and JSON-LD of web sites.

usage: python auditor.py --local ../sites      (start the sample sites, audit)
       python auditor.py https://your-own-site.example ...
One request per resource, a timeout, and an honest User-Agent.
"""
import json
import os
import re
import sys
import urllib.error
import urllib.request

from robots_rules import Robots

BOTS = ["GPTBot", "OAI-SearchBot", "ClaudeBot", "PerplexityBot",
        "Google-Extended", "Bingbot"]
AGENT = "Chapter10-Auditor/1.0 (Profitina book lab)"
RANK = {"PASS": 0, "INFO": 1, "WARN": 2, "FAIL": 3}
LDJSON = re.compile(r'<script[^>]*type\s*=\s*["\']application/ld\+json["\']'
                    r'[^>]*>(.*?)</script>', re.I | re.S)
ENTITIES = {"Organization", "LocalBusiness", "Restaurant", "Store"}


def fetch(url):
    """Return (status, headers with lower-case names, text); 0 if no reply."""
    request = urllib.request.Request(url, headers={"User-Agent": AGENT})
    try:
        with urllib.request.urlopen(request, timeout=5) as r:
            status, headers, body = r.status, r.headers, r.read()
    except urllib.error.HTTPError as e:
        status, headers, body = e.code, e.headers, e.read()
    except (urllib.error.URLError, OSError):
        return 0, {}, ""
    lower = {k.lower(): v for k, v in headers.items()}
    return status, lower, body.decode("utf-8", "replace")


class Report:
    def __init__(self):
        self.worst = {}
```

Kode 10.5 (Lanjutan) *auditor.py: memeriksa robots.txt, llms.txt, header cache, dan JSON-LD setiap situs, lalu mencetak laporan dan tabel ringkasan.*

```
    def line(self, category, status, text, *details):
        print("[%s] %s" % (status, text))
        for d in details:
            print("       " + d)
        old = self.worst.get(category)
        if old is None or RANK[status] > RANK[old]:
            self.worst[category] = status


def media_type(headers):
    return headers.get("content-type", "").split(";")[0].strip()


def check_robots(rep, base, origin):
    status, _, text = fetch(base + "/robots.txt")
    if status >= 500 or status == 0:
        rep.line("robots", "FAIL", "robots.txt: HTTP %d; crawlers must assume "
                 '"disallow all"' % status)
        return
    if status >= 400:
        rep.line("robots", "WARN", "robots.txt: HTTP %d; crawlers assume "
                 '"allow all"' % status)
        return
    robots = Robots(text)
    blocked = [b for b in BOTS if not robots.allowed(b, "/")]
    extra = ["sitemap: " + s for s in robots.sitemaps] or ["no Sitemap line"]
    extra += ["content signal: " + s for s in robots.signals]
    if blocked:
        rep.line("robots", "FAIL", "robots.txt: HTTP 200; blocked for "
                 + ", ".join(blocked), *extra)
    else:
        rep.line("robots", "PASS", "robots.txt: HTTP 200; all %d crawlers "
                 "may fetch /" % len(BOTS), *extra)
    _, _, bypass = fetch(base + "/robots.txt?audit-bypass=1")
    if bypass != text:
        rep.line("robots", "WARN", "robots.txt: served copy differs from a "
                 "cache-bypass copy", "a stale copy is cached; purge the cache")
        return
    if origin:
        _, _, at_origin = fetch(origin + "/robots.txt")
        if at_origin != text:
            o = Robots(at_origin)
            diff = [b for b in BOTS if o.allowed(b) != robots.allowed(b)]
            rep.line("robots", "WARN", "robots.txt: served copy differs from "
                     "the origin copy", "origin allows " + ", ".join(diff),
                     "something in front of the origin rewrites the file")
```

Kode 10.5 (Lanjutan) *auditor.py: memeriksa robots.txt, llms.txt, header cache, dan JSON-LD setiap situs, lalu mencetak laporan dan tabel ringkasan.*

```
            return
    agree = "served, cache-bypass and origin" if origin else \
        "served and cache-bypass"
    rep.line("robots", "PASS", "robots.txt: %s copies agree" % agree)


def check_llms(rep, base):
    status, headers, text = fetch(base + "/llms.txt")
    kind = media_type(headers)
    if status == 404:
        rep.line("llms", "INFO", "llms.txt: absent (HTTP 404); optional")
        return
    if status != 200:
        rep.line("llms", "WARN", "llms.txt: HTTP %d" % status)
        return
    if kind not in ("text/plain", "text/markdown"):
        rep.line("llms", "WARN", "llms.txt: HTTP 200 but %s" % kind,
                 'probably a "soft 404" page, not an llms.txt file')
        return
    lines = [s for s in text.splitlines() if s.strip()]
    if not lines or not lines[0].startswith("# "):
        rep.line("llms", "WARN", "llms.txt: first line is not an H1 title")
        return
    summary = "summary" if any(s.startswith(">") for s in lines) \
        else "no summary"
    links = len(re.findall(r"\[[^\]]+\]\([^)]+\)", text))
    rep.line("llms", "PASS", 'llms.txt: HTTP 200 %s; H1 "%s", %s, %s'
             % (kind, lines[0][2:].strip(), summary, plural(links, "link")))


def plural(n, word):
    return "%d %s%s" % (n, word, "" if n == 1 else "s")


def human(seconds):
    if seconds < 3600:
        return "%d min" % (seconds // 60)
    if seconds < 86400:
        return "%d h" % (seconds // 3600)
    return "%d days" % (seconds // 86400)
```

Kode 10.5 (Lanjutan) *auditor.py: memeriksa robots.txt, llms.txt, header cache, dan JSON-LD setiap situs, lalu mencetak laporan dan tabel ringkasan.*

```
def check_cache(rep, path, headers):
    cc = headers.get("cache-control", "")
    directives = {}
    for part in cc.split(","):
        name, _, value = part.strip().lower().partition("=")
        if name:
            directives[name] = value
    validator = "ETag" if "etag" in headers else \
        "Last-Modified" if "last-modified" in headers else "no validator"
    layers = ["%s %s" % (n, headers[n.lower()]) for n in
              ("X-Cache", "CF-Cache-Status", "X-LiteSpeed-Cache", "Age")
              if n.lower() in headers]
    extra = ["layer headers: " + ", ".join(layers)] if layers else []
    if not cc:
        rep.line("cache", "INFO", "cache %s: no Cache-Control; %s" %
                 (path, validator), "caches may guess a lifetime", *extra)
        return
    if {"no-store", "no-cache", "private"} & directives.keys():
        rep.line("cache", "PASS", "cache %s: not kept by shared caches; %s"
                 % (path, validator), *extra)
        return
    ttl = int(directives.get("s-maxage") or directives.get("max-age") or 0)
    text = "cache %s: shared caches keep it %s; %s" % (path, human(ttl),
                                                       validator)
    if ttl > 86400 and path == "/robots.txt":
        rep.line("cache", "WARN", text, "crawlers re-read robots.txt within "
                 "24 h; a long cache delays every fix", *extra)
    elif ttl > 86400:
        rep.line("cache", "WARN", text, "HTML cached this long needs a purge "
                 "of every layer after each edit", *extra)
    else:
        rep.line("cache", "PASS", text, *extra)


def walk(node, found):
    """Collect every schema.org node (a dict with @type) in document order."""
    if isinstance(node, dict):
        if "@type" in node:
            found.append(node)
        for value in node.values():
            walk(value, found)
    elif isinstance(node, list):
        for value in node:
            walk(value, found)
```

Kode 10.5 (Lanjutan) *auditor.py: memeriksa robots.txt, llms.txt, header cache, dan JSON-LD setiap situs, lalu mencetak laporan dan tabel ringkasan.*

```
def types_of(node):
    t = node["@type"]
    return t if isinstance(t, list) else [t]


def date_modified(html):
    for block in LDJSON.findall(html):
        try:
            found = []
            walk(json.loads(block), found)
        except ValueError:
            continue
        for node in found:
            if "dateModified" in node:
                return node["dateModified"]
    return None


def check_jsonld(rep, html, fresh_html):
    blocks = LDJSON.findall(html)
    if not blocks:
        rep.line("jsonld", "WARN", "JSON-LD: none found on the home page")
        return
    nodes, bad = [], 0
    for block in blocks:
        try:
            walk(json.loads(block), nodes)
        except ValueError:
            bad += 1
    types = []
    for node in nodes:
        for t in types_of(node):
            if t not in types:
                types.append(t)
    details = ["types: " + (", ".join(types) or "none")]
    for node in nodes:
        if "FAQPage" in types_of(node):
            n = len(node.get("mainEntity", []))
            details.append("FAQPage with %d questions" % n)
    served = date_modified(html)
    details.append("dateModified " + (served or "absent"))
    if bad:
        rep.line("jsonld", "FAIL", "JSON-LD: %d of %d blocks are not valid "
                 "JSON; engines skip them" % (bad, len(blocks)), *details)
    else:
        rep.line("jsonld", "PASS", "JSON-LD: %s, all valid JSON"
```

Kode 10.5 (Lanjutan) *auditor.py: memeriksa robots.txt, llms.txt, header cache, dan JSON-LD setiap situs, lalu mencetak laporan dan tabel ringkasan.*

```
                % plural(len(blocks), "block"), *details)
    for node in nodes:
        kinds = [t for t in types_of(node) if t in ENTITIES]
        for key in ("name", "url"):
            if kinds and key not in node:
                rep.line("jsonld", "WARN", 'JSON-LD: %s has no "%s"'
                         % (kinds[0], key))
    fresh = date_modified(fresh_html)
    if served and fresh and served != fresh:
        rep.line("jsonld", "WARN", "JSON-LD: served page says dateModified "
                 "%s, origin says %s" % (served, fresh),
                 "a stale copy is served; purge the cache layers")


def audit(name, base, origin=None):
    print("== %s  %s" % (name, base))
    rep = Report()
    check_robots(rep, base, origin)
    check_llms(rep, base)
    _, home_headers, html = fetch(base + "/")
    _, robots_headers, _ = fetch(base + "/robots.txt")
    check_cache(rep, "/", home_headers)
    check_cache(rep, "/robots.txt", robots_headers)
    _, _, fresh = fetch((origin or base) + "/?audit-bypass=1")
    check_jsonld(rep, html, fresh)
    print()
    return rep.worst


def main():
    args = sys.argv[1:]
    targets, servers = [], []
    if args[:1] == ["--local"]:
        from site_server import read_conf, start_all
        sites_dir = args[1] if len(args) > 1 else "../sites"
        servers = start_all(sites_dir)
        confs = [(read_conf(os.path.join(sites_dir, site)), site)
                 for site in os.listdir(sites_dir)
                 if os.path.isfile(os.path.join(sites_dir, site, "_site.conf"))]
        for conf, site in sorted(confs, key=lambda c: c[0]["port"]):
            base = "http://127.0.0.1:%d" % conf["port"]
            origin = conf["origin_port"] and \
                "http://127.0.0.1:%d" % conf["origin_port"]
            targets.append((site, base, origin))
    else:
        for url in args:
```

Kode 10.5 (Lanjutan) *auditor.py: memeriksa robots.txt, llms.txt, header cache, dan JSON-LD setiap situs, lalu mencetak laporan dan tabel ringkasan.*

```
            url = url.rstrip("/")
            targets.append((url.split("//")[-1], url, None))
    if not targets:
        print(__doc__.strip())
        return
    results = [(name, audit(name, base, origin))
               for name, base, origin in targets]
    for _, server in servers:
        server.shutdown()
    print("%-10s %-7s %-7s %-7s %s" % ("site", "robots", "llms", "cache",
                                       "json-ld"))
    for name, worst in results:
        print("%-10s %-7s %-7s %-7s %s" % (name, *[worst.get(c, "-") for c in
              ("robots", "llms", "cache", "jsonld")]))
    print("\nNote on llms.txt: cheap hygiene, not a proven citation signal. In")
    print("May 2026, 97% of llms.txt files got zero requests (Ahrefs log study)")
    print("of 137,210 domains).")


if __name__ == "__main__":
    main()
```

Membaca pemeriksaan-pemeriksaannya sesuai urutan pemanggilan oleh `audit`:

- **`fetch`** mengembalikan status, *header* dengan nama berhuruf kecil, dan isi respons. Galat HTTP seperti 404 atau 500 bukan eksepsi bagi auditor melainkan jawaban yang perlu dilaporkan, sehingga ditangkap dan dikembalikan seperti jawaban lain; hanya situs yang sama sekali tidak menjawab yang menjadi status 0.
- **`check_robots`** pertama-tama menerapkan **Tabel 10.2**: galat server atau ketiadaan jawaban bernilai `FAIL`, karena *crawler* lalu wajib menganggap semuanya dilarang; 4xx hanya `WARN`, karena *crawler* lalu boleh membaca semuanya. Untuk 2xx, fungsi itu mengurai berkas dengan `Robots` dan menanyakan, untuk masing-masing dari enam *crawler*, apakah *crawler* itu boleh mengambil `/`. Lalu datang dua perbandingan dari Subbab 10.5.5: salinan yang tayang dibandingkan dengan salinan yang diambil dengan `?audit-bypass=1`, dan, bila alamat origin diketahui, salinan yang tayang dibandingkan dengan salinan origin. Perbedaan pertama berarti *cache* basi; perbedaan kedua berarti ada lapisan yang menulis ulang berkas itu, dan auditor menyebutkan *crawler* yang aturannya berbeda antara origin dan berkas yang tayang.
- **`check_llms`** memisahkan empat keadaan: tidak ada berkas (404, hanya `INFO`, karena berkas itu opsional), galat, "berkas" yang sebenarnya halaman HTML berstatus 200 (*soft 404* di situs `stale`), dan berkas sungguhan, yang harus diawali baris H1. Untuk berkas sungguhan, fungsi itu melaporkan judulnya, ada tidaknya ringkasan berupa kutipan blok, dan jumlah tautan Markdown.
- **`check_cache`** mengurai `Cache-Control` menjadi direktif-direktif dan menghitung masa hidup *cache* bersama menurut Persamaan (10.4): `s-maxage` bila ada, jika tidak `max-age`. Tanpa *header* sama sekali bernilai `INFO`, karena *cache* akan menebak; `no-store`, `no-cache`, atau `private` bernilai `PASS`; masa hidup lebih dari sehari bernilai `WARN`, dengan penjelasan berbeda untuk HTML dan untuk `robots.txt`. Fungsi itu juga mendaftar *header* lapisan *cache* yang dikenalnya (`X-Cache`, `CF-Cache-Status`, `X-LiteSpeed-Cache`, `Age`), dan dari situlah Anda dapat membedakan `HIT` dari `MISS` di situs nyata.
- **`check_jsonld`** mengekstrak setiap blok `<script type="application/ld+json">` dengan ekspresi reguler, mengurai setiap blok dengan pengurai JSON yang ketat, dan menelusuri hasilnya untuk mengumpulkan setiap simpul yang memiliki `@type`, termasuk simpul yang bersarang di dalam simpul lain, seperti `PostalAddress` di dalam `LocalBusiness`. Fungsi itu

melaporkan blok yang tidak sah, tipe-tipe yang ditemukan, jumlah pertanyaan `FAQPage` bila ada, dan `dateModified`; memperingatkan simpul mirip organisasi yang tidak punya `name` atau `url`; dan membandingkan `dateModified` yang tayang dengan milik origin, pemeriksaan dari Subbab 10.8.3.

Fungsi `main` punya dua mode. Dengan `--local`, fungsi itu membaca berkas konfigurasi situs-situs contoh, menjalankan semua server mereka di dalam proses yang sama, mengaudit situs-situs itu sesuai urutan *port*, dan mematikan server-server itu sebelum mencetak ringkasan. Dengan satu URL atau lebih, fungsi itu mengaudit situs-situs tersebut; begitulah Anda mengarahkannya ke situs Anda sendiri.

Versi Java memerlukan satu hal yang sudah tersedia di Python: pengurai JSON. **Kode 10.6** adalah pengurai kecil yang ketat, cukup untuk memutuskan apakah sebuah blok adalah JSON yang sah dan untuk menelusuri objek dan larik di dalamnya. Pengurai ini sengaja menolak koma di akhir, sebagaimana wajib dilakukan setiap pengurai yang patuh standar.

Kode 10.6 `MiniJson.java`: pembaca JSON kecil dan ketat untuk auditor Java; apa pun yang bukan JSON sah, seperti koma di akhir, melemparkan eksepsi.

```
import java.util.*;

/** A tiny, strict JSON reader: objects become LinkedHashMap, arrays become
 *  ArrayList, numbers Double, plus String, Boolean and null. Anything that
 *  is not valid JSON (a trailing comma, for example) throws. */
public class MiniJson {
    private final String s;
    private int i;

    private MiniJson(String s) {
        this.s = s;
    }

    public static Object parse(String text) {
        MiniJson p = new MiniJson(text);
        Object value = p.value();
        p.space();
        if (p.i != p.s.length()) throw p.error("extra text");
        return value;
    }

    private IllegalArgumentException error(String what) {
        return new IllegalArgumentException(what + " at offset " + i);
    }

    private void space() {
        while (i < s.length() && " \t\r\n".indexOf(s.charAt(i)) >= 0) i++;
    }

    private char peek() {
        space();
        if (i >= s.length()) throw error("unexpected end");
        return s.charAt(i);
    }

    private void expect(char c) {
        if (peek() != c) throw error("expected '" + c + "'");
        i++;
    }
```

Kode 10.6 (Lanjutan) *MiniJson.java: pembaca JSON kecil dan ketat untuk auditor Java; apa pun yang bukan JSON sah, seperti koma di akhir, melemparkan eksepsi.*

```
    private Object value() {
        char c = peek();
        if (c == '{') return object();
        if (c == '[') return array();
        if (c == '"') return string();
        for (String word : new String[] {"true", "false", "null"}) {
            if (s.startsWith(word, i)) {
                i += word.length();
                return word.equals("null") ? null : Boolean.valueOf(word);
            }
        }
        return number();
    }

    private Map<String, Object> object() {
        Map<String, Object> map = new LinkedHashMap<>();
        expect('{');
        if (peek() == '}') {
            i++;
            return map;
        }
        while (true) {
            String key = string();
            expect(':');
            map.put(key, value());
            if (peek() == ',') {
                i++;
                continue;                    // the next key must follow
            }
            expect('}');
            return map;
        }
    }

    private List<Object> array() {
        List<Object> list = new ArrayList<>();
        expect('[');
        if (peek() == ']') {
            i++;
            return list;
        }
        while (true) {
            list.add(value());
            if (peek() == ',') {
                i++;
                continue;
```

Kode 10.6 (Lanjutan) *MiniJson.java: pembaca JSON kecil dan ketat untuk auditor Java; apa pun yang bukan JSON sah, seperti koma di akhir, melemparkan eksepsi.*

```
            }
            expect(']');
            return list;
        }
    }

    private String string() {
        expect('"');
        StringBuilder b = new StringBuilder();
        while (true) {
            if (i >= s.length()) throw error("unterminated string");
            char c = s.charAt(i++);
            if (c == '"') return b.toString();
            if (c < 0x20) throw error("control character in string");
            if (c != '\\') {
                b.append(c);
                continue;
            }
            if (i >= s.length()) throw error("unterminated string");
            char e = s.charAt(i++);
            switch (e) {
                case '"', '\\', '/' -> b.append(e);
                case 'b' -> b.append('\b');
                case 'f' -> b.append('\f');
                case 'n' -> b.append('\n');
                case 'r' -> b.append('\r');
                case 't' -> b.append('\t');
                case 'u' -> {
                    if (i + 4 > s.length()) throw error("bad \\u escape");
                    b.append((char) Integer.parseInt(s.substring(i, i + 4), 16));
                    i += 4;
                }
                default -> throw error("bad escape");
            }
        }
    }
```

Kode 10.6 (Lanjutan) *MiniJson.java: pembaca JSON kecil dan ketat untuk auditor Java; apa pun yang bukan JSON sah, seperti koma di akhir, melemparkan eksepsi.*

```
    private Double number() {
        int start = i;
        while (i < s.length() && "+-0123456789.eE".indexOf(s.charAt(i)) >= 0) i++;
        try {
            return Double.valueOf(s.substring(start, i));
        } catch (NumberFormatException e) {
            i = start;
            throw error("not a JSON value");
        }
    }
}
```

Kode 10.7 adalah auditor versi Java. Auditor ini memakai `java.net.http.HttpClient` untuk permintaannya, dipatok ke HTTP/1.1 agar berbicara dengan server contoh dengan cara yang sama seperti `urllib`, dan mencetak laporan yang persis sama dengan versi Python, karakter demi karakter.

Kode 10.7 `Auditor.java`: auditor yang sama dalam Java; mencetak laporan yang persis sama dengan **Kode 10.5**.

```
import com.sun.net.httpserver.HttpServer;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Files;
import java.nio.file.Path;
import java.time.Duration;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** Audits robots.txt, llms.txt, cache headers and JSON-LD of web sites.
 *  usage: java Auditor --local ../sites   (start the sample sites, audit)
 *         java Auditor https://your-own-site.example ...
 *  One request per resource, a timeout, and an honest User-Agent. */
public class Auditor {
    static final List<String> BOTS = List.of("GPTBot", "OAI-SearchBot",
            "ClaudeBot", "PerplexityBot", "Google-Extended", "Bingbot");
    static final String AGENT = "Chapter10-Auditor/1.0 (Profitina book lab)";
    static final List<String> RANK = List.of("PASS", "INFO", "WARN", "FAIL");
    static final Pattern LDJSON = Pattern.compile(
            "<script[^>]*type\\s*=\\s*[\"']application/ld\\+json[\"']"
            + "[^>]*>(.*?)</script>", Pattern.CASE_INSENSITIVE | Pattern.DOTALL);
    static final Set<String> ENTITIES = Set.of("Organization", "LocalBusiness",
            "Restaurant", "Store");
    static final HttpClient HTTP = HttpClient.newBuilder()
            .version(HttpClient.Version.HTTP_1_1)
            .connectTimeout(Duration.ofSeconds(5)).build();

    record Page(int status, Map<String, String> headers, String text) {}

    /** Returns status, headers with lower-case names, and text; 0 if no reply. */
    static Page fetch(String url) {
        try {
            HttpRequest req = HttpRequest.newBuilder(URI.create(url))
                    .header("User-Agent", AGENT)
                    .timeout(Duration.ofSeconds(5)).build();
            HttpResponse<String> r = HTTP.send(req,
                    HttpResponse.BodyHandlers.ofString());
            Map<String, String> h = new HashMap<>();
            r.headers().map().forEach((k, v) -> h.put(k.toLowerCase(), v.get(0)));
            return new Page(r.statusCode(), h, r.body());
        } catch (Exception e) {
            return new Page(0, Map.of(), "");
        }
```

Kode 10.7 (Lanjutan) *Auditor.java: auditor yang sama dalam Java; mencetak laporan yang persis sama dengan Kode 10.5.*

```
    }

    /** Prints report lines and remembers the worst status per category. */
    static class Report {
        final Map<String, String> worst = new HashMap<>();

        void line(String category, String status, String text, String... details) {
            System.out.println("[" + status + "] " + text);
            for (String d : details) System.out.println("       " + d);
            String old = worst.get(category);
            if (old == null || RANK.indexOf(status) > RANK.indexOf(old)) {
                worst.put(category, status);
            }
        }
    }

    static String mediaType(Map<String, String> headers) {
        return headers.getOrDefault("content-type", "").split(";")[0].strip();
    }

    static List<String> blockedBy(RobotsRules robots) {
        List<String> out = new ArrayList<>();
        for (String b : BOTS) if (!robots.allowed(b)) out.add(b);
        return out;
    }

    static void checkRobots(Report rep, String base, String origin) {
        Page p = fetch(base + "/robots.txt");
        if (p.status() >= 500 || p.status() == 0) {
            rep.line("robots", "FAIL", "robots.txt: HTTP " + p.status()
                    + "; crawlers must assume \"disallow all\"");
            return;
        }
        if (p.status() >= 400) {
            rep.line("robots", "WARN", "robots.txt: HTTP " + p.status()
                    + "; crawlers assume \"allow all\"");
            return;
        }
        RobotsRules robots = new RobotsRules(p.text());
        List<String> blocked = blockedBy(robots);
        List<String> extra = new ArrayList<>();
        for (String s : robots.sitemaps) extra.add("sitemap: " + s);
        if (extra.isEmpty()) extra.add("no Sitemap line");
        for (String s : robots.signals) extra.add("content signal: " + s);
        String[] ex = extra.toArray(new String[0]);
        if (!blocked.isEmpty()) {
```

Kode 10.7 (Lanjutan) *Auditor.java: auditor yang sama dalam Java; mencetak laporan yang persis sama dengan Kode 10.5.*

```
            rep.line("robots", "FAIL", "robots.txt: HTTP 200; blocked for "
                    + String.join(", ", blocked), ex);
        } else {
            rep.line("robots", "PASS", "robots.txt: HTTP 200; all " + BOTS.size()
                    + " crawlers may fetch /", ex);
        }
        String bypass = fetch(base + "/robots.txt?audit-bypass=1").text();
        if (!bypass.equals(p.text())) {
            rep.line("robots", "WARN", "robots.txt: served copy differs from a "
                    + "cache-bypass copy",
                    "a stale copy is cached; purge the cache");
            return;
        }
        if (origin != null) {
            String atOrigin = fetch(origin + "/robots.txt").text();
            if (!atOrigin.equals(p.text())) {
                RobotsRules o = new RobotsRules(atOrigin);
                List<String> diff = new ArrayList<>();
                for (String b : BOTS) {
                    if (o.allowed(b) != robots.allowed(b)) diff.add(b);
                }
                rep.line("robots", "WARN", "robots.txt: served copy differs from "
                        + "the origin copy",
                        "origin allows " + String.join(", ", diff),
                        "something in front of the origin rewrites the file");
                return;
            }
        }
        String agree = origin != null ? "served, cache-bypass and origin"
                                      : "served and cache-bypass";
        rep.line("robots", "PASS", "robots.txt: " + agree + " copies agree");
    }

    static void checkLlms(Report rep, String base) {
        Page p = fetch(base + "/llms.txt");
        String kind = mediaType(p.headers());
        if (p.status() == 404) {
            rep.line("llms", "INFO", "llms.txt: absent (HTTP 404); optional");
            return;
        }
        if (p.status() != 200) {
            rep.line("llms", "WARN", "llms.txt: HTTP " + p.status());
            return;
        }
        if (!kind.equals("text/plain") && !kind.equals("text/markdown")) {
            rep.line("llms", "WARN", "llms.txt: HTTP 200 but " + kind,
```

Kode 10.7 (Lanjutan) *Auditor.java: auditor yang sama dalam Java; mencetak laporan yang persis sama dengan Kode 10.5.*

```
                    "probably a \"soft 404\" page, not an llms.txt file");
            return;
        }
        List<String> lines = new ArrayList<>();
        for (String s : p.text().split("\r?\n")) if (!s.isBlank()) lines.add(s);
        if (lines.isEmpty() || !lines.get(0).startsWith("# ")) {
            rep.line("llms", "WARN", "llms.txt: first line is not an H1 title");
            return;
        }
        boolean hasSummary = lines.stream().anyMatch(s -> s.startsWith(">"));
        int links = 0;
        Matcher m = Pattern.compile("\\[[^\\]]+\\]\\([^)]+\\)").matcher(p.text());
        while (m.find()) links++;
        rep.line("llms", "PASS", "llms.txt: HTTP 200 " + kind + "; H1 \""
                + lines.get(0).substring(2).strip() + "\", "
                + (hasSummary ? "summary" : "no summary") + ", "
                + plural(links, "link"));
    }

    static String plural(int n, String word) {
        return n + " " + word + (n == 1 ? "" : "s");
    }

    static String human(long seconds) {
        if (seconds < 3600) return (seconds / 60) + " min";
        if (seconds < 86400) return (seconds / 3600) + " h";
        return (seconds / 86400) + " days";
    }

    static void checkCache(Report rep, String path, Map<String, String> headers) {
        String cc = headers.getOrDefault("cache-control", "");
        Map<String, String> directives = new HashMap<>();
        for (String part : cc.split(",")) {
            String d = part.strip().toLowerCase();
            if (d.isEmpty()) continue;
            int eq = d.indexOf('=');
            directives.put(eq < 0 ? d : d.substring(0, eq),
                           eq < 0 ? "" : d.substring(eq + 1));
        }
        String validator = headers.containsKey("etag") ? "ETag"
                : headers.containsKey("last-modified") ? "Last-Modified"
                : "no validator";
        List<String> layers = new ArrayList<>();
        for (String n : List.of("X-Cache", "CF-Cache-Status", "X-LiteSpeed-Cache",
                                "Age")) {
            String v = headers.get(n.toLowerCase());
```

Kode 10.7 (Lanjutan) *Auditor.java: auditor yang sama dalam Java; mencetak laporan yang persis sama dengan Kode 10.5.*

```
            if (v != null) layers.add(n + " " + v);
        }
        List<String> extra = new ArrayList<>();
        if (!layers.isEmpty()) {
            extra.add("layer headers: " + String.join(", ", layers));
        }
        if (cc.isEmpty()) {
            extra.add(0, "caches may guess a lifetime");
            rep.line("cache", "INFO", "cache " + path + ": no Cache-Control; "
                    + validator, extra.toArray(new String[0]));
            return;
        }
        if (directives.containsKey("no-store") || directives.containsKey("no-cache")
                || directives.containsKey("private")) {
            rep.line("cache", "PASS", "cache " + path + ": not kept by shared "
                    + "caches; " + validator, extra.toArray(new String[0]));
            return;
        }
        String age = directives.containsKey("s-maxage") ? directives.get("s-maxage")
                : directives.getOrDefault("max-age", "0");
        long ttl = Long.parseLong(age.isEmpty() ? "0" : age);
        String text = "cache " + path + ": shared caches keep it " + human(ttl)
                + "; " + validator;
        if (ttl > 86400 && path.equals("/robots.txt")) {
            extra.add(0, "crawlers re-read robots.txt within 24 h; a long cache "
                    + "delays every fix");
            rep.line("cache", "WARN", text, extra.toArray(new String[0]));
        } else if (ttl > 86400) {
            extra.add(0, "HTML cached this long needs a purge of every layer "
                    + "after each edit");
            rep.line("cache", "WARN", text, extra.toArray(new String[0]));
        } else {
            rep.line("cache", "PASS", text, extra.toArray(new String[0]));
        }
    }
```

Kode 10.7 (Lanjutan) *Auditor.java: auditor yang sama dalam Java; mencetak laporan yang persis sama dengan Kode 10.5.*

```
    /** Collects every schema.org node (a map with @type) in document order. */
    @SuppressWarnings("unchecked")
    static void walk(Object node, List<Map<String, Object>> found) {
        if (node instanceof Map<?, ?> map) {
            if (map.containsKey("@type")) found.add((Map<String, Object>) map);
            for (Object v : map.values()) walk(v, found);
        } else if (node instanceof List<?> list) {
            for (Object v : list) walk(v, found);
        }
    }

    static List<String> typesOf(Map<String, Object> node) {
        Object t = node.get("@type");
        List<String> out = new ArrayList<>();
        if (t instanceof List<?> list) {
            for (Object x : list) out.add(String.valueOf(x));
        } else {
            out.add(String.valueOf(t));
        }
        return out;
    }

    static List<String> blocks(String html) {
        List<String> out = new ArrayList<>();
        Matcher m = LDJSON.matcher(html);
        while (m.find()) out.add(m.group(1));
        return out;
    }

    static String dateModified(String html) {
        for (String block : blocks(html)) {
            List<Map<String, Object>> found = new ArrayList<>();
            try {
                walk(MiniJson.parse(block), found);
            } catch (IllegalArgumentException e) {
                continue;
            }
            for (var node : found) {
                if (node.containsKey("dateModified")) {
                    return String.valueOf(node.get("dateModified"));
                }
            }
        }
        return null;
    }
```

Kode 10.7 (Lanjutan) *Auditor.java: auditor yang sama dalam Java; mencetak laporan yang persis sama dengan Kode 10.5.*

```
    static void checkJsonLd(Report rep, String html, String freshHtml) {
        List<String> blocks = blocks(html);
        if (blocks.isEmpty()) {
            rep.line("jsonld", "WARN", "JSON-LD: none found on the home page");
            return;
        }
        List<Map<String, Object>> nodes = new ArrayList<>();
        int bad = 0;
        for (String block : blocks) {
            try {
                walk(MiniJson.parse(block), nodes);
            } catch (IllegalArgumentException e) {
                bad++;
            }
        }
        List<String> types = new ArrayList<>();
        for (var node : nodes) {
            for (String t : typesOf(node)) if (!types.contains(t)) types.add(t);
        }
        List<String> details = new ArrayList<>();
        details.add("types: "
                + (types.isEmpty() ? "none" : String.join(", ", types)));
        for (var node : nodes) {
            if (typesOf(node).contains("FAQPage")) {
                Object q = node.get("mainEntity");
                int n = q instanceof List<?> list ? list.size() : 0;
                details.add("FAQPage with " + n + " questions");
            }
        }
        String served = dateModified(html);
        details.add("dateModified " + (served != null ? served : "absent"));
        String[] d = details.toArray(new String[0]);
        if (bad > 0) {
            rep.line("jsonld", "FAIL", "JSON-LD: " + bad + " of " + blocks.size()
                    + " blocks are not valid JSON; engines skip them", d);
        } else {
            rep.line("jsonld", "PASS", "JSON-LD: " + plural(blocks.size(), "block")
                    + ", all valid JSON", d);
        }
        for (var node : nodes) {
            List<String> kinds = new ArrayList<>(typesOf(node));
            kinds.retainAll(ENTITIES);
            for (String key : List.of("name", "url")) {
                if (!kinds.isEmpty() && !node.containsKey(key)) {
                    rep.line("jsonld", "WARN", "JSON-LD: " + kinds.get(0)
                            + " has no \"" + key + "\"");
```

Kode 10.7 (Lanjutan) *Auditor.java: auditor yang sama dalam Java; mencetak laporan yang persis sama dengan Kode 10.5.*

```
                }
            }
        }
        String fresh = dateModified(freshHtml);
        if (served != null && fresh != null && !served.equals(fresh)) {
            rep.line("jsonld", "WARN", "JSON-LD: served page says dateModified "
                    + served + ", origin says " + fresh,
                    "a stale copy is served; purge the cache layers");
        }
    }

    static Map<String, String> audit(String name, String base, String origin) {
        System.out.println("== " + name + "  " + base);
        Report rep = new Report();
        checkRobots(rep, base, origin);
        checkLlms(rep, base);
        Page home = fetch(base + "/");
        Page robots = fetch(base + "/robots.txt");
        checkCache(rep, "/", home.headers());
        checkCache(rep, "/robots.txt", robots.headers());
        String fresh = fetch((origin != null ? origin : base)
                + "/?audit-bypass=1").text();
        checkJsonLd(rep, home.text(), fresh);
        System.out.println();
        return rep.worst;
    }

    public static void main(String[] args) throws Exception {
        List<String[]> targets = new ArrayList<>();       // {name, base, origin}
        Map<String, HttpServer> servers = Map.of();
        if (args.length > 0 && args[0].equals("--local")) {
            Path sitesDir = Path.of(args.length > 1 ? args[1] : "../sites");
            servers = SiteServer.startAll(sitesDir);
            List<Path> sites = new ArrayList<>();
            try (var list = Files.list(sitesDir)) {
                list.filter(p -> Files.isRegularFile(p.resolve("_site.conf")))
                    .forEach(sites::add);
            }
            Map<Integer, String[]> byPort = new TreeMap<>();
            for (Path dir : sites) {
                SiteServer.Conf conf = SiteServer.readConf(dir);
                byPort.put(conf.port, new String[] {dir.getFileName().toString(),
                        "http://127.0.0.1:" + conf.port, conf.originPort == null
                        ? null : "http://127.0.0.1:" + conf.originPort});
            }
            targets.addAll(byPort.values());
```

Kode 10.7 (Lanjutan) *Auditor.java: auditor yang sama dalam Java; mencetak laporan yang persis sama dengan Kode 10.5.*

```
        } else {
            for (String url : args) {
                String u = url.replaceAll("/+$", "");
                String name = u.substring(u.indexOf("//") + 2);
                targets.add(new String[] {name, u, null});
            }
        }
        if (targets.isEmpty()) {
            System.out.println("usage: java Auditor --local ../sites | URL ...");
            return;
        }
        List<Map<String, String>> results = new ArrayList<>();
        for (String[] t : targets) results.add(audit(t[0], t[1], t[2]));
        servers.values().forEach(s -> s.stop(0));
        System.out.printf("%-10s %-7s %-7s %-7s %s%n", "site", "robots", "llms",
                "cache", "json-ld");
        for (int k = 0; k < targets.size(); k++) {
            Map<String, String> w = results.get(k);
            System.out.printf("%-10s %-7s %-7s %-7s %s%n", targets.get(k)[0],
                    w.getOrDefault("robots", "-"), w.getOrDefault("llms", "-"),
                    w.getOrDefault("cache", "-"), w.getOrDefault("jsonld", "-"));
        }
        System.out.println("\nNote on llms.txt: cheap hygiene, not a proven "
                + "citation signal. In");
        System.out.println("May 2026, 97% of llms.txt files got zero requests "
                + "(Ahrefs log study");
        System.out.println("of 137,210 domains).");
    }
}
```

10.9.5 Menjalankan auditor

Dalam Python, dari folder `python`:

```
python auditor.py --local ../sites
```

Dalam Java, dari folder `java`:

```
javac -d out *.java
java -cp out Auditor --local ../sites
```

Keduanya mencetak laporan berikut. (Kedua keluaran dibandingkan dengan `diff` dan identik.)

```
== laundry  http://127.0.0.1:7100
[PASS] robots.txt: HTTP 200; all 6 crawlers may fetch /
       sitemap: http://127.0.0.1:7100/sitemap.xml
       content signal: search=yes, ai-input=yes, ai-train=no
[PASS] robots.txt: served and cache-bypass copies agree
[PASS] llms.txt: HTTP 200 text/plain; H1 "Profitina Laundry", summary, 3 links
[PASS] cache /: shared caches keep it 5 min; ETag
[PASS] cache /robots.txt: shared caches keep it 1 h; ETag
[PASS] JSON-LD: 3 blocks, all valid JSON
       types: LocalBusiness, PostalAddress, FAQPage, Question, Answer, WebPage
       FAQPage with 2 questions
       dateModified 2026-09-20

== cdnblock  http://127.0.0.1:7101
[FAIL] robots.txt: HTTP 200; blocked for GPTBot, ClaudeBot, Google-Extended
       sitemap: http://127.0.0.1:7101/sitemap.xml
[WARN] robots.txt: served copy differs from the origin copy
       origin allows GPTBot, ClaudeBot, Google-Extended
       something in front of the origin rewrites the file
[PASS] llms.txt: HTTP 200 text/plain; H1 "Kopi Kita Roastery", summary, 1 link
[PASS] cache /: shared caches keep it 10 min; ETag
       layer headers: X-Cache MISS
[PASS] cache /robots.txt: shared caches keep it 4 h; ETag
[PASS] JSON-LD: 1 block, all valid JSON
       types: Organization
       dateModified absent

== stale  http://127.0.0.1:7102
[PASS] robots.txt: HTTP 200; all 6 crawlers may fetch /
       no Sitemap line
[PASS] robots.txt: served, cache-bypass and origin copies agree
[WARN] llms.txt: HTTP 200 but text/html
       probably a "soft 404" page, not an llms.txt file
[WARN] cache /: shared caches keep it 30 days; no validator
       HTML cached this long needs a purge of every layer after each edit
       layer headers: X-Cache HIT, Age 5400
[WARN] cache /robots.txt: shared caches keep it 7 days; no validator
       crawlers re-read robots.txt within 24 h; a long cache delays every fix
       layer headers: X-Cache MISS
[PASS] JSON-LD: 2 blocks, all valid JSON
       types: Organization, WebPage
       dateModified 2026-08-01
[WARN] JSON-LD: served page says dateModified 2026-08-01, origin says 2026-09-18
       a stale copy is served; purge the cache layers

== broken  http://127.0.0.1:7103
[FAIL] robots.txt: HTTP 500; crawlers must assume "disallow all"
[INFO] llms.txt: absent (HTTP 404); optional
[INFO] cache /: no Cache-Control; no validator
       caches may guess a lifetime
[INFO] cache /robots.txt: no Cache-Control; no validator
       caches may guess a lifetime
[FAIL] JSON-LD: 1 of 2 blocks are not valid JSON; engines skip them
       types: Organization
       dateModified absent
[WARN] JSON-LD: Organization has no "url"

site       robots  llms    cache   json-ld
laundry    PASS    PASS    PASS    PASS
cdnblock   FAIL    PASS    PASS    PASS
stale      PASS    WARN    WARN    WARN
broken     FAIL    INFO    INFO    FAIL

Note on llms.txt: cheap hygiene, not a proven citation signal. In
May 2026, 97% of llms.txt files got zero requests (Ahrefs log study
of 137,210 domains).
```

Bacalah situs demi situs, dan bandingkan setiap temuan dengan isi bab.

`laundry` lolos di semua pemeriksaan. Keenam *crawler* boleh mengambil `/`; salinan `robots.txt` yang tayang sama dengan salinan yang melewati *cache*; `llms.txt` memiliki judul, ringkasan, dan tiga tautan; HTML disimpan 5 menit dan `robots.txt` 1 jam, keduanya dengan `ETag`; dan JSON-LD-nya terdiri atas tiga blok sah, dengan `LocalBusiness` beserta `PostalAddress`-nya, `FAQPage` dengan dua pertanyaan, dan `WebPage` dengan `dateModified`.

`cdnblock` gagal pada `robots.txt` persis seperti profitina.com dulu. `GPTBot`, `ClaudeBot`, dan `Google-Extended` diblokir, padahal origin mengizinkan ketiganya, dan auditor menunjukkan ke mana harus mencari: salinan yang tayang sama dengan salinan yang melewati *cache*, jadi ini bukan *cache* basi, tetapi berbeda dari salinan origin, jadi "sesuatu di depan origin menulis ulang berkas itu". Segala hal lain di situs itu sehat, dan justru itulah yang membuat kesalahan ini sulit disadari dari peramban.

`stale` lolos pada `robots.txt` tetapi mendapat tiga peringatan. `llms.txt`-nya adalah halaman HTML berstatus 200, sebuah *soft 404*. Berandanya boleh disimpan *cache* bersama selama 30 hari dan tidak memiliki validator, sedangkan `robots.txt`-nya 7 hari, melampaui 24 jam yang diperkenankan bagi *crawler* sendiri. Dan halaman yang benar-benar disajikan, sebuah `HIT` *cache* yang berusia 5.400 detik, menyatakan `dateModified 2026-08-01`, sementara origin menyatakan `2026-09-18`: pengunjung dan *crawler* sedang membaca halaman "segera hadir" dari enam minggu lalu.

`broken` gagal dua kali. `robots.txt`-nya menjawab 500, sehingga setiap *crawler* yang santun wajib memperlakukan seluruh situs sebagai terlarang. Satu dari dua blok JSON-LD-nya memiliki koma di akhir dan tidak sah; blok yang sah menggambarkan `Organization` tanpa `url`. Situs itu sama sekali tidak mengirim *header caching*, yang dilaporkan auditor sebagai `INFO`: *cache* boleh menebak.

Tabel ringkasan meringkas setiap kategori menjadi temuan terburuknya, dan catatan di bagian bawah mengulang peringatan Subbab 10.6.2, agar tidak ada yang membaca `PASS` untuk `llms.txt` sebagai sebuah janji.

10.9.6 Kejutan di pustaka standar Python

Python menyertakan pengurai `robots.txt`, yaitu `urllib.robotparser`. Mengapa lab ini menulis pengurainya sendiri? **Kode 10.8** menjawabnya dengan mengajukan dua pertanyaan yang sama kepada keduanya.

Kode 10.8 `robots_quirk.py`: dua pertanyaan yang sama diajukan kepada `urllib.robotparser` dan kepada pembaca RFC 9309 di **Kode 10.3**.

```
"""Compare urllib.robotparser (first match) with RFC 9309 (longest match)."""
import urllib.robotparser

from robots_rules import Robots

CASES = [
    ("longest match", "User-agent: *\nDisallow: /\nAllow: /blog/\n",
     "/blog/post-1"),
    ("merged groups", "User-agent: GPTBot\nDisallow: /private/\n\n"
     "User-agent: *\nAllow: /\n\nUser-agent: GPTBot\nAllow: /private/faq\n",
     "/private/faq"),
]

for title, text, path in CASES:
    stdlib = urllib.robotparser.RobotFileParser()
    stdlib.parse(text.splitlines())
    print("%-14s %-14s robotparser: %-5s RFC 9309: %s" % (
        title, path, stdlib.can_fetch("GPTBot", path),
        Robots(text).allowed("GPTBot", path)))
```

Pada Python 3.11, versi yang dipakai untuk menghasilkan keluaran di buku ini, program itu mencetak:

```
longest match  /blog/post-1   robotparser: False RFC 9309: True
merged groups  /private/faq   robotparser: False RFC 9309: True
```

Pada kasus pertama, `Disallow: /` muncul sebelum `Allow: /blog/` di dalam berkas, dan `robotparser` versi lama membiarkan aturan *pertama* yang cocok memutuskan; kecocokan terpanjang RFC 9309 mengizinkan jalur itu. Pada kasus kedua, `GPTBot` disebut oleh dua grup, dan pengurai lama hanya memakai grup pertama. Selama bertahun-tahun inilah perilaku pustaka standar, dan masih itulah yang Anda dapat pada *interpreter* yang lebih tua. Rilis pemeliharaan terbaru menulis ulang modul itu: sumber terkini di cabang 3.13 dan 3.14 mengurutkan aturan menurut panjang kecocokan, menangani `*` dan `$`, dan menggabungkan grup untuk agen yang sama [26], sehingga pada Python 3.14 terkini kedua kolom mencetak `True`. Pelajarannya bukan tentang Python semata. Perilaku sebuah pustaka adalah fakta tentang sebuah *versi*, dan program yang harus mengikuti standar sebaiknya mematok versi yang diandalkannya atau menguji perilaku yang dibutuhkannya, seperti yang dilakukan kode ini. Pustaka standar Java sama sekali tidak memiliki pengurai `robots.txt`, dan itulah sebabnya kedua versi lab membawa pengurainya sendiri.

10.9.7 Mengarahkan auditor ke situs nyata

Berikan satu alamat atau lebih kepada auditor sebagai ganti `--local`:

```
python auditor.py https://your-own-site.example
java -cp out Auditor https://your-own-site.example
```

Tanpa alamat origin, perbandingan `robots.txt` hanya dapat mendeteksi salinan *cache* yang basi, bukan lapisan yang menulis ulang; bila penyedia *hosting* memberi Anda cara menjangkau origin, misalnya alamat pratinjau atau nama server itu sendiri, bandingkan pula dengannya. Auditlah situs Anda sendiri, atau situs yang pemiliknya setuju, dan pertahankan laju permintaan auditor, satu permintaan per sumber daya. Ingat juga bahwa putusan auditor adalah tentang kebersihan: situs yang lolos semua pemeriksaan telah menyingkirkan penghalang untuk dibaca dan dipahami, yang perlu untuk dikutip tetapi tidak cukup.

10.10 Profitina dalam Praktik

Semua yang ada di bab ini menggambarkan situs pemasaran sebagaimana adanya pada 24 September 2026. Apa yang sudah berjalan dan apa yang hanya pelajaran mudah dipisahkan.

Berjalan di profitina.com. Situs WordPress dengan tema dwibahasa khusus dan Polylang, halaman bahasa Indonesia dan Inggris berdampingan, di-*hosting* dengan LiteSpeed Cache di belakang Cloudflare. `robots.txt` yang mengizinkan *crawler* mesin pencari dan AI dengan menyebut namanya, menutup `/wp-admin/`, menunjuk `/wp-sitemap.xml`, menyebut `llms.txt`, dan menyatakan sinyal konten `search=yes, ai-input=yes, ai-train=no` [12]. `llms.txt` yang meringkas platform dengan jujur, termasuk bahwa PROFIT adalah token utilitas di devnet dan bukan investasi [14]. Data terstruktur FAQ dan `dateModified` pada halaman konten. Widget komunitas yang disematkan dan hanya dapat dibaca, serta tautan legal yang mengarah ke satu-satunya rumah Kebijakan Privasi serta Syarat dan Ketentuan di aplikasi [1].

Berjalan sebagai aturan tim. "Selalu verifikasi URL yang tayang, dengan melewati *cache*" setelah setiap perubahan yang menyangkut akses *crawler* [3]. *Purge* LiteSpeed dulu, Cloudflare kemudian. Beri nomor versi pada setiap perubahan tema. Beri awalan pada nama-nama milik tema.

Di dalam produk. Pengetahuan yang sama sampai ke pengguna Profitina melalui aplikasi. Langkah *Pindai* membaca sinyal nyata kesiapan sebuah situs, di antaranya apakah *crawler* AI boleh membacanya dan apakah data terstrukturnya sah; fitur *Technical Assistant* pada langkah *Optimasi* membantu urusan `robots.txt`, khususnya mengizinkan *crawler* AI, dan pengiriman peta situs ke Bing; dan setiap hasil membawa pengingat bahwa estimasi kesiapan bukan jaminan atas apa yang akan dilakukan mesin pihak ketiga [3]. Bab ini mengajarkan landasan publik dan baku tempat sinyal-sinyal itu berpijak, dengan sebuah lab yang dapat dijalankan siapa saja.

Tidak diklaim. Profitina tidak mengklaim bahwa `llms.txt` menyebabkan kutipan, bahwa *markup* FAQ menghasilkan fitur pencarian setelah Mei 2026, atau bahwa berkas apa pun menjamin sebuah mesin penjawab AI akan mengutip suatu situs. Profitina tidak menanyai ChatGPT, Gemini, atau Perplexity untuk mengukur kutipan; mesin-mesin itu adalah sasaran yang kesiapannya diestimasi, tidak pernah komponen tempat platform ini bergantung [3].

10.11 Kesalahpahaman yang Sering Terjadi

"`robots.txt` saya benar; saya sudah memeriksa berkasnya." Anda memeriksa *sebuah* berkas. *Crawler* menerima apa pun yang dikembalikan URL publik setelah setiap lapisan di depan server Anda ikut bicara. Periksalah URL-nya, dengan dan tanpa *query string* yang melewati *cache* (Subbab 10.5.5).

"`User-agent: *` dengan `Allow: /` menyambut setiap *crawler*." Baris itu menyambut setiap *crawler* yang tidak punya grup sendiri. Satu grup saja di mana pun dalam berkas yang menyebut `GPTBot`, termasuk yang ditambahkan CDN atau *plugin*, mengeluarkan `GPTBot` dari grup `*` sepenuhnya (Persamaan (10.1)).

"Aturan pertama yang cocok yang menang." Menurut RFC 9309, aturan cocok yang *terpanjang* yang menang, dan `Allow` menang bila seri (Persamaan (10.2)). Urutan baris tidak berpengaruh. Beberapa pengurai lama, termasuk `urllib.robotparser` Python versi lama, memang memakai kecocokan pertama (Subbab 10.9.6).

"Kalau `robots.txt` rusak, *crawler* toh akan membaca semuanya." Hanya bila rusaknya berupa 4xx. Status 5xx, atau ketiadaan jawaban, memberi tahu *crawler* untuk menganggap *semuanya* dilarang (**Tabel 10.2**).

"Tanpa *header* `Cache-Control` berarti tidak di-*cache*." Artinya *cache* memilih masa hidupnya sendiri, secara heuristik (Persamaan (10.4)).

"Mengosongkan kedua *cache* sudah cukup; urutannya hanya rincian." Dengan urutan yang salah, satu permintaan di antara dua *purge* mengisi kembali *cache* luar dengan salinan lama selama satu masa hidup penuh (Persamaan (10.6)).

"`llms.txt` adalah cara AI menemukan situs saya." Pada Mei 2026, 97% berkas `llms.txt` tidak menerima satu permintaan pun, dan bot penelusuran AI hanya menyumbang 1,1% dari permintaan yang datang [17]. Berkas itu kebersihan yang murah; akses ditentukan oleh `robots.txt`.

"*Markup* FAQ tak ada gunanya sejak Google menghentikan *rich result* FAQ." Fitur visualnya hilang; *markup*-nya tetap sah, tetap tidak berbahaya, dan tetap merupakan pernyataan tanya-jawab yang tidak ambigu bagi setiap program yang membaca halaman itu [24].

"Baris `Content-Signal` memblokir pelatihan AI." Baris itu menyatakan preferensi tentang pemakaian. Boleh tidaknya sebuah *crawler* mengambil halaman tetap ditentukan aturan `Allow` dan `Disallow`, dan apakah sebuah perusahaan menghormati sinyal itu terserah perusahaan tersebut.

10.12 Ringkasan Bab

- profitina.com adalah situs WordPress dengan tema dwibahasa khusus (Polylang, halaman bahasa Indonesia dan Inggris yang nyata) di belakang dua lapisan *cache*: LiteSpeed Cache di *host* dan Cloudflare di *edge*. Aplikasi di app.profitina.com terpisah, dan teks-teks hukum hanya ada di sana.
- Satu nama global bersama di *loop* WordPress (`$id`) pernah mencampur bahasa pada label *hero* situs; perbaikan akar masalahnya adalah nama berawalan (`$pf_id`). Kode yang berjalan di dalam *loop* sebuah *framework* tidak boleh meminjam nama milik *framework* itu.
- RFC 9309 memilih grup menurut Persamaan (10.1): setiap grup yang menyebut *crawler* itu, digabungkan, atau bila tidak ada, grup `*`; dan memilih aturan menurut Persamaan (10.2): kecocokan terpanjang, dengan `Allow` menang bila seri. Karakter pengganti `*` dan `$` termasuk bagian standar. Status 4xx untuk `robots.txt` berarti "tidak ada aturan"; status 5xx berarti "anggap semuanya dilarang".
- `robots.txt` terkelola milik CDN menyisipkan grup untuk `GPTBot` dan `ClaudeBot` di depan, yang diam-diam menimpa sambutan situs. Aturan yang lahir darinya: selalu verifikasi URL yang tayang, dengan melewati *cache*, dan bandingkan dengan origin.
- `llms.txt` diterbitkan oleh 9,3% dari 1.000 domain teratas dan 8,3% dari 10.000 domain teratas (September 2026), tetapi 97% berkas itu tidak menerima permintaan pada Mei 2026. Profitina menerbitkannya sebagai kebersihan dan tidak mengklaim lebih.
- Kesegaran HTTP adalah Persamaan (10.3), dengan masa hidup *cache* bersama menurut Persamaan (10.4). Dengan dua *cache* berlapis, kosongkan dari dalam ke luar; jendela basinya kemudian Persamaan (10.5), bukan Persamaan (10.6) yang jauh lebih panjang.
- Simpul FAQ, organisasi, dan halaman dengan `dateModified` membuat fakta situs tidak ambigu; sejak 7 Mei 2026 *markup* FAQ tidak lagi menghasilkan *rich result* Google tetapi tetap sah.

- Auditor di lab, dalam Python dan Java, menemukan `robots.txt` yang ditulis ulang, galat server pada `robots.txt`, `llms.txt` berupa *soft 404*, *caching* HTML selama sebulan, halaman *cache* yang basi, dan JSON-LD yang tidak sah pada empat situs contoh lokal, serta mencetak laporan yang sama dalam kedua bahasa.

10.13 Soal Latihan

Soal 10.1. Bagilah berkas berikut menjadi grup-grup, lalu untuk setiap grup sebutkan token produk dan aturan-aturannya. Baris mana yang tidak termasuk grup mana pun?

```
User-agent: Bingbot
User-agent: Googlebot
Disallow: /search
Sitemap: https://shop.example/sitemap.xml
User-agent: *
Disallow: /cart/
Allow: /cart/help
```

Soal 10.2. Untuk berkas pada Soal 10.1, tentukan secara manual, dengan memakai Persamaan (10.1) dan Persamaan (10.2), apakah setiap permintaan berikut diizinkan: `Googlebot` untuk `/search?q=batik`; `Googlebot` untuk `/cart/`; `GPTBot` untuk `/cart/`; `GPTBot` untuk `/cart/help`; `GPTBot` untuk `/cart/helpdesk`.

Soal 10.3. Sebuah grup memuat `Disallow: /*.pdf$` dan `Allow: /public/`. Dengan menghitung panjang setiap pola, tentukan apakah `/public/report.pdf`, `/public/report.pdf?v=2`, dan `/files/a.pdf` diizinkan.

Soal 10.4. Berkas milik seorang pemilik situs berbunyi `User-agent: GPTBot` / `Allow: /blog/`, diikuti `User-agent: *` / `Allow: /`. Sebuah CDN menyisipkan `User-agent: GPTBot` / `Disallow: /` di depannya. (a) Menurut RFC 9309, bolehkah `GPTBot` mengambil `/blog/post-1`? `/about`? (b) Apa keputusan *crawler* yang hanya mematuhi grup *pertama* yang menyebut namanya? (c) Apa yang sebaiknya diubah pemilik situs?

Soal 10.5. Untuk setiap respons atas `GET /robots.txt`, sebutkan apa yang dilakukan *crawler* yang patuh RFC 9309: (a) 404; (b) 503; (c) 200 dengan isi kosong; (d) 200 dengan halaman galat HTML yang bertuliskan “Page not found”. Untuk (d), apa kesimpulan kelas `Robots` di lab, dan mengapa hal itu bermasalah?

Soal 10.6. Sebuah halaman dikirim dengan `Cache-Control: public, max-age=600, s-maxage=86400`. (a) Berapa lama halaman itu segar di peramban, dan di CDN? (b) Respons yang sama juga membawa `CDN-Cache-Control: max-age=60`. Berapa lama sekarang halaman itu segar di CDN, menurut RFC 9213? (c) Masa hidup mana yang dilaporkan auditor di lab untuk *header* aslinya, dan mengapa?

Soal 10.7. Sebuah suntingan diterbitkan pada $t_0 = 0$. Satu *cache* dikosongkan pada $t_1 = 1$ menit dan yang lain pada $t_2 = 10$ menit. *Edge* menyimpan halaman itu selama $T_C = 2$ jam. Hitung jendela basi pada urutan yang benar, dan pada urutan yang salah untuk permintaan pada $t_r = 6$ menit serta untuk waktu permintaan terburuk.

Soal 10.8. Ahrefs menemukan berkas `llms.txt` yang sah pada 38.360 dari 137.210 domain, dan 97% berkas itu tidak menerima permintaan pada Mei 2026. (a) Berapa bagian dari seluruh domain yang diteliti yang memiliki berkas sah? (b) Kira-kira berapa berkas yang menerima setidaknya satu permintaan? (c) Mengapa keliru menyimpulkan bahwa “3% situs mendapat manfaat dari `llms.txt`”?

Soal 10.9. Tambahkan `Claude-SearchBot` milik Anthropic ke daftar *crawler* yang diperiksa auditor, dalam kedua bahasa. Ramalkan baris `robots.txt` setiap situs contoh sebelum menjalankannya, lalu jalankan auditor dan bandingkan.

Soal 10.10. Ubahlah auditor, dalam kedua bahasa, sehingga `robots.txt` tanpa satu pun baris `Sitemap` menghasilkan satu baris `WARN` tambahan. Ringkasan situs contoh mana yang berubah, dan menjadi apa?

Soal 10.11. Mengapa `check_jsonld` mengambil beranda origin dengan `?audit-bypass=1`, bukan tanpa *query string*? Apa yang akan salah untuk situs `stale` bila fungsi itu memakai alamat publik tanpa *query string*?

Soal 10.12. Tulislah `llms.txt` untuk pemanggang kopi fiktif milik situs `cdnblock`, Kopi Kita Roastery, yang diterima auditor dengan `PASS` dan memiliki ringkasan, dua bagian tautan, serta bagian `Optional`. Bagian mana dari berkas itu yang diperiksa auditor, dan bagian mana yang hanya dapat diperiksa manusia?

10.14 Penyelesaian Soal Latihan

Penyelesaian 10.1. Ada dua grup. Grup 1 memiliki token produk `bingbot` dan `googlebot` (dua baris `User-agent` yang berurutan) dan satu aturan, `Disallow: /search`. Baris `Sitemap` tidak termasuk grup mana pun; baris itu juga tidak menutup grup 1, tetapi baris `User-agent` berikutnya memulai grup baru karena baris bermakna sebelumnya di grup 1 adalah aturan. Grup 2 memiliki token `*` dan aturan `Disallow: /cart/` serta `Allow: /cart/help`.

Penyelesaian 10.2. `Googlebot` disebut oleh grup 1, sehingga menurut Persamaan (10.1) ia hanya mematuhi grup 1 dan tidak pernah grup `*`. `/search?q=batik` diawali `/search`, jadi `Disallow: /search` cocok dan permintaan itu diblokir. `/cart/` tidak cocok dengan aturan mana pun di grup 1, sehingga *diizinkan* bagi `Googlebot`: `Disallow: /cart/` milik grup `*` tidak berlaku bagi *crawler* yang punya grup sendiri. `GPTBot` tidak disebut grup mana pun, sehingga mematuhi grup 2. `/cart/` hanya cocok dengan `Disallow: /cart/` (panjang 6): diblokir. `/cart/help` cocok dengan `Disallow: /cart/` (6) dan `Allow: /cart/help` (10); yang lebih panjang menang: diizinkan. `/cart/helpdesk` juga diawali `/cart/help`, sehingga kedua aturan yang sama cocok dan jalur itu pun diizinkan, pengingat bahwa pola mencocokkan awalan.

Penyelesaian 10.3. Pola `/*.pdf$` sepanjang 7 oktet dan `/public/` sepanjang 8. Untuk `/public/report.pdf` keduanya cocok (yang pertama karena jalurnya berakhir dengan `.pdf`), dan yang lebih panjang, `Allow: /public/` dengan 8 oktet, menang: diizinkan. Untuk `/public/report.pdf?v=2` pola `$` tidak cocok, karena jalurnya tidak berakhir dengan `.pdf`; hanya `Allow: /public/` yang cocok: diizinkan. Untuk `/files/a.pdf` hanya `Disallow: /*.pdf$` yang cocok: diblokir.

Penyelesaian 10.4. (a) Berkas yang tayang memiliki dua grup yang menyebut `GPTBot`, grup sisipan dan grup milik pemilik. Menurut Persamaan (10.1) keduanya digabung menjadi aturan `Disallow: /` (panjang 1) dan `Allow: /blog/` (panjang 6), dan grup `*` diabaikan. Untuk `/blog/post-1` keduanya cocok dan `Allow` yang lebih panjang menang: diizinkan. Untuk `/about` hanya `Disallow: /` yang cocok: diblokir. Grup `*` milik pemilik, yang berbunyi `Allow: /`, sama sekali tidak berperan. (b) *Crawler* yang hanya mematuhi grup pertama yang menyebut namanya melihat `Disallow: /` saja dan diblokir di mana-mana, termasuk di `/blog/`. (c) Matikan fitur `robots.txt` terkelola milik CDN (atau atur agar tidak mengusik *crawler* yang diinginkan pemilik), dan nyatakan maksud pemilik untuk setiap *crawler* secara lengkap di grupnya sendiri, misalnya

`Allow: /` untuk `GPTBot`, sehingga tidak ada cara baca gabungan yang dapat menyempitkannya. Lalu verifikasi URL publiknya.

Penyelesaian 10.5. (a) Status 404 berarti berkas "tidak tersedia": *crawler* boleh membaca semuanya. (b) Status 503 adalah galat server, "tidak terjangkau": *crawler* wajib menganggap semuanya dilarang (dan Google menghentikan perayapan selama 12 jam pertama sambil mencoba lagi). (c) Berkas kosong tidak memiliki grup maupun aturan, sehingga setiap jalur diizinkan. (d) *Crawler* menerima halaman HTML berstatus 200 dan mengurainya sebagai `robots.txt`. Baris-baris HTML tidak memiliki struktur `kunci: nilai` yang membentuk grup, sehingga `Robots` tidak menemukan grup apa pun dan menyimpulkan bahwa semuanya diizinkan. Di sini kebetulan tidak berbahaya, tetapi artinya situs itu tidak menyatakan apa yang dimaksudkannya; bila teks halaman itu kebetulan memuat baris `User-agent: *` dan `Disallow: /`, baris-baris itu malah akan dipatuhi. Kode status menceritakan kisah yang keliru kepada *crawler*, persis masalah *soft 404* yang ditangkap `check_llms` untuk `llms.txt`.

Penyelesaian 10.6. (a) Peramban, sebuah *cache* privat, memakai `max-age`: 600 detik, 10 menit. CDN, sebuah *cache* bersama, memakai `s-maxage` bila ada, menurut Persamaan (10.4): 86.400 detik, satu hari. (b) RFC 9213 menyatakan bahwa *cache* CDN yang menemukan `CDN-Cache-Control` yang sah wajib memakainya dan mengabaikan `Cache-Control`: halaman itu segar di CDN selama 60 detik, sementara peramban tetap menyimpannya 10 menit. (c) Auditor hanya membaca `Cache-Control` dan melaporkan masa hidup *cache* bersama, `s-maxage`: ia mencetak `shared caches keep it 1 days` (fungsi `human`-nya yang kecil tidak repot membedakan bentuk tunggal), dan karena 86.400 detik tidak *lebih* dari satu hari, baris itu bernilai `PASS`. Menambahkan dukungan `CDN-Cache-Control` adalah perluasan yang wajar: baca *header* itu lebih dulu bila ada, sebagaimana dinyatakan RFC.

Penyelesaian 10.7. Urutan benar, Persamaan (10.5): $W = t_2 - t_0 = 10 - 0 = 10$ menit. Urutan salah, Persamaan (10.6), dengan $t_r = 6$: $W = 6 + 120 - 0 = 126$ menit. Kasus terburuk adalah permintaan tepat sebelum *purge* kedua, $t_r \to t_2 = 10$: $W \to 10 + 120 = 130$ menit, sedikit di atas dua jam, alih-alih sepuluh menit.

Penyelesaian 10.8. (a) $38\,360/137\,210 \approx 0.280$, sekitar 28% dari domain yang diteliti. (b) $3\% \times 38\,360 = 0.03 \times 38\,360 \approx 1\,151$ berkas. (c) Sebuah permintaan bukanlah manfaat. Sebagian besar permintaan yang datang berasal dari bot yang sama sekali bukan alat AI, seperti alat audit SEO, dan hanya 1,1% berasal dari bot penelusuran AI [17]; lagi pula, permintaan dari sistem AI sekalipun tidak mengatakan apa-apa tentang apakah situs itu kemudian dikutip. Pemakaian adalah syarat perlu bagi dampak, bukan buktinya.

Penyelesaian 10.9. Dalam Python, tambahkan `"Claude-SearchBot"` ke `BOTS`; dalam Java, tambahkan ke `List.of(...)` pada `BOTS`. Ramalannya: tidak ada situs contoh yang menyebut `Claude-SearchBot`, sehingga *crawler* itu selalu kembali ke grup `*`. `laundry` dan `stale` mengizinkan semua pihak di bawah `*`, sehingga barisnya menjadi `all 7 crawlers may fetch /`. Grup `*` milik `cdnblock` mengizinkan semua pihak dan blok terkelolanya tidak menyebut `Claude-SearchBot`, sehingga daftar yang diblokir tetap `GPTBot, ClaudeBot, Google-Extended`. `broken` tetap gagal dengan HTTP 500 sebelum *crawler* mana pun ditimbang. Menjalankan auditor Python yang telah diubah mengonfirmasinya:

```
== laundry  http://127.0.0.1:7100
[PASS] robots.txt: HTTP 200; all 7 crawlers may fetch /
== cdnblock  http://127.0.0.1:7101
[FAIL] robots.txt: HTTP 200; blocked for GPTBot, ClaudeBot, Google-Extended
== stale  http://127.0.0.1:7102
[PASS] robots.txt: HTTP 200; all 7 crawlers may fetch /
== broken  http://127.0.0.1:7103
[FAIL] robots.txt: HTTP 500; crawlers must assume "disallow all"
```

Penyelesaian 10.10. Di `check_robots` pada **Kode 10.5**, sisipkan sebelum baris yang mengambil `?audit-bypass=1`:

```
    if not robots.sitemaps:
        rep.line("robots", "WARN", "robots.txt: no Sitemap line; add one so "
                 "crawlers find every page")
```

dan di `checkRobots` pada **Kode 10.7**, sebelum baris yang mengambil salinan *bypass*:

```
        if (robots.sitemaps.isEmpty()) {
            rep.line("robots", "WARN", "robots.txt: no Sitemap line; add one so "
                    + "crawlers find every page");
        }
```

Hanya `stale` yang memiliki `robots.txt` tanpa baris `Sitemap` (`broken` tidak pernah sampai ke tahap itu, karena berkasnya menjawab 500). Bagian laporannya bertambah satu baris, dan kolom `robots`-nya di ringkasan berubah dari `PASS` menjadi `WARN`; kedua bahasa mencetak:

```
== stale  http://127.0.0.1:7102
[PASS] robots.txt: HTTP 200; all 6 crawlers may fetch /
       no Sitemap line
[WARN] robots.txt: no Sitemap line; add one so crawlers find every page
[PASS] robots.txt: served, cache-bypass and origin copies agree
```

Penyelesaian 10.11. Perbandingan itu dimaksudkan untuk mempertentangkan apa yang diterima pengunjung dengan apa yang *benar saat ini*. Untuk situs yang alamat origin-nya diketahui, origin tidak pernah menyajikan salinan *cache*, sehingga *query string* hanyalah pengaman; untuk situs tanpa alamat origin, *query string* adalah satu-satunya cara auditor melewati *cache* yang berkuncikan URL lengkap. Bila `check_jsonld` mengambil alamat publik `stale` tanpa *query string*, ia akan menerima `HIT` *cache* yang sama dengan halaman yang tayang, kedua tanggal akan sama-sama `2026-08-01`, dan halaman basi itu lolos tanpa terdeteksi. Inti pemeriksaan itu adalah mengajukan pertanyaan yang tidak dapat dijawab oleh *cache*.

Penyelesaian 10.12. Misalnya:

```
# Kopi Kita Roastery

> Kopi Kita Roastery is a fictional coffee roaster in Malang that sells
> single-origin beans from Java, roasted every Monday.

Orders close on Sunday evening; delivery is within Java only.

## Products

- [Beans](http://127.0.0.1:7101/beans.html): origins, roast levels, prices
- [Brewing guide](http://127.0.0.1:7101/brew.html): grind and ratio per method

## Company

- [About](http://127.0.0.1:7101/about.html): who roasts, and where

## Optional

- [Blog](http://127.0.0.1:7101/blog.html): harvest notes and events
```

Auditor memeriksa bahwa berkas itu disajikan dengan status 200 sebagai `text/plain` (atau `text/markdown`), bahwa baris tak-kosong pertamanya adalah H1 (`#`), apakah ada ringkasan berupa kutipan blok, dan berapa tautan Markdown yang dimuatnya; berkas ini memperoleh `PASS` dengan "summary, 4 links". Hanya manusia yang dapat memeriksa bahwa ringkasannya benar, bahwa tautan-tautannya berfungsi dan mengarah ke halaman yang menyatakan apa yang dijanjikan catatannya, dan bahwa berkas itu selaras dengan bagian lain situs, aturan "jaga agar fakta tetap benar" yang sama yang diterapkan Bab 2 pada data terstruktur.

Daftar Pustaka

[1] Profitina, "Profitina – Gather · Grow · Give," marketing site. [Online]. Available: https://profitina.com (accessed Sep. 24, 2026).

[2] W3Techs, "Usage statistics and market share of WordPress," Sep. 2026. [Online]. Available: https://w3techs.com/technologies/details/cm-wordpress (accessed Sep. 24, 2026).

[3] Profitina, *Profitina Whitepaper — Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[4] WP SYNTEX, "Polylang," WordPress plugin directory, version 3.8.9. [Online]. Available: https://wordpress.org/plugins/polylang/ (accessed Sep. 24, 2026).

[5] Google Search Central, "Tell Google about localized versions of your page." [Online]. Available: https://developers.google.com/search/docs/specialty/international/localized-versions (accessed Sep. 24, 2026).

[6] WordPress Developer Resources, "setup_postdata()." [Online]. Available: https://developer.wordpress.org/reference/functions/setup_postdata/ (accessed Sep. 24, 2026).

[7] M. Koster, G. Illyes, H. Zeller, and L. Sassman, "Robots Exclusion Protocol," IETF RFC 9309, Sep. 2022.

[8] Google Search Central, "How Google interprets the robots.txt specification." [Online]. Available: https://developers.google.com/search/docs/crawling-indexing/robots/robots_txt (accessed Sep. 24, 2026).

[9] Cloudflare, "Managed robots.txt," Cloudflare Bots documentation. [Online]. Available: https://developers.cloudflare.com/bots/additional-configurations/managed-robots-txt/ (accessed Sep. 24, 2026).

[10] Cloudflare, "Control content use for AI training with Cloudflare's managed robots.txt and blocking for monetized content," *The Cloudflare Blog*, Jul. 1, 2025. [Online]. Available: https://blog.cloudflare.com/control-content-use-for-ai-training/

[11] Cloudflare, "Your site, your rules: new AI traffic options for all customers," *The Cloudflare Blog*, Jul. 1, 2026. [Online]. Available: https://blog.cloudflare.com/content-independence-day-ai-options/

[12] Profitina, “robots.txt,” profitina.com. [Online]. Available: https://profitina.com/robots.txt (accessed Sep. 24, 2026).

[13] J. Howard, “The /llms.txt file,” llmstxt.org, proposed Sep. 3, 2024. [Online]. Available: https://llmstxt.org/

[14] Profitina, “llms.txt,” profitina.com. [Online]. Available: https://profitina.com/llms.txt (accessed Sep. 24, 2026).

[15] Rankability, “LLMS.txt adoption,” data page, scan of Sep. 18, 2026 (Tranco list of Sep. 17, 2026). [Online]. Available: https://www.rankability.com/data/llms-txt-adoption/

[16] PPC Land, “llms.txt adoption rises 8.8x but 97% of files get zero AI requests,” Jul. 2, 2026. [Online]. Available: https://ppc.land/llms-txt-adoption-rises-8-8x-but-97-of-files-get-zero-ai-requests/

[17] L. Linehan and X. Guan, “We analyzed 137K sites: 97% of llms.txt files never get read,” *Ahrefs Blog*, Jun. 15, 2026. [Online]. Available: https://ahrefs.com/blog/llmstxt-study/

[18] R. Fielding, M. Nottingham, and J. Reschke, “HTTP Caching,” IETF RFC 9111, Jun. 2022.

[19] S. Ludin, M. Nottingham, and Y. Wu, “Targeted HTTP Cache Control,” IETF RFC 9213, Jun. 2022.

[20] Cloudflare, “Cloudflare cache responses,” Cloudflare Cache documentation. [Online]. Available: https://developers.cloudflare.com/cache/concepts/cache-responses/ (accessed Sep. 24, 2026).

[21] LiteSpeed Technologies, “Troubleshooting guide,” LSCache for WordPress documentation. [Online]. Available: https://docs.litespeedtech.com/lscache/lscwp/troubleshoot/ (accessed Sep. 24, 2026).

[22] Cloudflare, “Default cache behavior,” Cloudflare Cache documentation. [Online]. Available: https://developers.cloudflare.com/cache/concepts/default-cache-behavior/ (accessed Sep. 24, 2026).

[23] Schema.org, “Schema.org releases,” version 30.1, Sep. 16, 2026. [Online]. Available: https://schema.org/docs/releases.html

[24] Search Engine Journal, “Google drops FAQ rich results from search,” May 10, 2026. [Online]. Available: https://www.searchenginejournal.com/google-drops-faq-rich-results-from-search/574429/

[25] P. Aggarwal, V. Murahari, T. Rajpurohit, A. Kalyan, K. Narasimhan, and A. Deshpande, “GEO: Generative Engine Optimization,” in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD ’24)*, 2024. arXiv:2311.09735.

[26] Python Software Foundation, “Lib/urllib/robotparser.py,” CPython source, 3.14 branch. [Online]. Available: https://github.com/python/cpython/blob/3.14/Lib/urllib/robotparser.py (accessed Sep. 24, 2026).

Bab 11

Privasi dan Kepatuhan sejak Rancangan (UU PDP)

Setelah menyelesaikan bab ini, Anda akan mampu menjelaskan Undang-Undang Pelindungan Data Pribadi, UU PDP No. 27/2022, dengan bahasa sederhana: apa yang termasuk data pribadi, siapa yang bertanggung jawab atasnya, hak apa saja yang dimiliki seseorang atas data tentang dirinya, dan apa yang wajib dilakukan perusahaan untuk menjaganya. Anda akan tahu posisi hukum itu pada September 2026, dengan peraturan pelaksananya, PP 33/2026, yang ditetapkan pada Juli 2026 dan mulai berlaku pada Januari 2027, serta lembaga pengawasnya yang masih dalam pembentukan, dan Anda akan mampu menyebutkan sumbernya. Anda akan memahami privasi sejak rancangan (*privacy by design*) sebagai metode rekayasa, bukan slogan, dan mampu menelusuri setiap langkah yang diterapkan Profitina dari hukum sampai ke kode: kata sandi yang disimpan dengan Argon2id, persetujuan yang eksplisit dan dicatat dengan versi serta waktu, pembatasan laju login, ekspor data, penggantian alamat email atau kata sandi, penghapusan akun secara penuh, dan halaman legal yang diterbitkan sebagai HTML biasa. Anda akan mampu menghitung mengapa *hash* kata sandi yang lambat melindungi pengguna sementara yang cepat tidak, serta menalar pembatas laju berjendela geser (*sliding window*) secara manual. Anda akan tahu mengapa hak yang tidak punya tombol belum menjadi hak yang dapat dipakai orang. Terakhir, Anda akan membangun lab G7, aplikasi kecil dalam Python dan Java yang mencatat persetujuan, meng-*hash* kata sandi dengan PBKDF2 dari pustaka standar (atau Argon2id bila paket opsionalnya terpasang), membatasi login yang gagal, mengekspor data seseorang sebagai JSON, dan menghapus akun sedemikian tuntas sehingga sebuah uji dapat membuktikan tidak ada lagi yang bersifat pribadi yang tersisa.

11.1 Titik Berangkat Kita

Bab 10 memandang Profitina dari luar: situs pemasaran, *cache*-nya, dan berkas-berkas yang memberi tahu *crawler* apa yang boleh dibacanya. Bab itu ditutup dengan prinsip yang juga berlaku di bab ini: yang penting adalah apa yang benar-benar terjadi, bukan apa yang dikatakan sebuah berkas atau dokumen tentang apa yang seharusnya terjadi.

Bab ini beralih ke orang-orang yang memakai platform dan data yang mereka tinggalkan di dalamnya. Setiap layanan yang memungkinkan orang mendaftar mengumpulkan data pribadi, sekalipun hanya alamat email. Di Indonesia, sejak Oktober 2022, pengumpulan itu diatur oleh sebuah undang-undang yang menyeluruh, UU PDP, dan sejak Oktober 2024 setiap organisasi diharapkan mematuhinya sepenuhnya. Mahasiswa yang membangun aplikasi laundry untuk tugas kelas dan perusahaan yang menjalankan platform dengan ribuan pengguna tunduk pada prinsip yang sama; yang berubah seiring ukuran adalah skala akibatnya.

Profitina dibangun dengan undang-undang itu sebagai masukan rancangan, bukan sebagai dokumen yang ditulis belakangan. Peta jalannya memuat satu fase khusus untuk itu, yang diselesaikan sebelum dan di sekitar peluncuran Juli 2026: *hashing* kata sandi dengan Argon2id, persetujuan eksplisit, pembatasan laju login, hak-hak data swalayan, dan halaman legal [1]. Bab 3 sudah menelusuri fitur-fitur itu dari sisi pengguna. Di sini kita melihatnya dari sisi insinyur: apa

yang diwajibkan hukum, bagaimana setiap kewajiban menjadi satu potong rancangan, dan bagaimana menguji bahwa rancangan itu melakukan apa yang dijanjikannya.

Bab ini terdiri atas tiga bagian. Subbab 11.2 dan 11.3 menjelaskan hukumnya dan posisinya pada 2026. Subbab 11.4 sampai 11.9 menguraikan privasi sejak rancangan menjadi langkah-langkah konkret: data minimal, *hashing* kata sandi, persetujuan, pembatasan laju, hak subjek data, dan halaman legal. Subbab 11.10 membangun lab G7, dan Subbab 11.11 sampai 11.15 menutup bab dengan Profitina dalam praktik, kesalahpahaman yang sering terjadi, ringkasan, soal latihan, dan penyelesaian lengkapnya.

Bab ini menjelaskan hukum untuk insinyur. Isinya bukan nasihat hukum, dan perusahaan yang memproses data pribadi dalam skala besar sebaiknya berkonsultasi dengan ahli hukum. Namun insinyur juga tidak dapat menyerahkan hukum sepenuhnya kepada ahli hukum: hampir setiap kewajiban dalam UU PDP pada akhirnya dijalankan oleh kode.

11.2 UU PDP No. 27/2022 dalam Bahasa Sederhana

11.2.1 Satu undang-undang, enam belas bab

Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi, yang biasa disingkat UU PDP, disahkan dan diundangkan pada 17 Oktober 2022 [2]. Undang-undang ini memiliki 76 pasal dalam 16 bab. Inilah undang-undang pelindungan data pertama di Indonesia yang bersifat menyeluruh: sebelumnya, aturan tentang data pribadi tersebar di berbagai peraturan sektoral. Pembaca yang mengenal GDPR Eropa akan mengenali bentuknya, tetapi UU PDP adalah undang-undang tersendiri, dengan istilahnya sendiri, pasal-pasalnya sendiri, dan penegakannya sendiri.

Undang-undang ini memberi organisasi waktu dua tahun untuk menyesuaikan diri: Pasal 74 mewajibkan setiap pihak yang memproses data pribadi untuk menyesuaikan diri “paling lama 2 (dua) tahun” sejak undang-undang berlaku, yaitu sampai 17 Oktober 2024 [2]. Sejak saat itu, kepatuhan bukan lagi pilihan.

11.2.2 Apa itu data pribadi?

Data pribadi adalah data tentang orang perseorangan yang teridentifikasi atau dapat diidentifikasi, secara tersendiri atau dikombinasikan dengan informasi lain. Undang-undang membaginya menjadi dua jenis (Pasal 4) [2]:

- **Data pribadi yang bersifat umum**: nama lengkap, jenis kelamin, kewarganegaraan, agama, status perkawinan, dan data pribadi yang dikombinasikan untuk mengidentifikasi seseorang. Alamat email seperti `ayu@example.com` mengidentifikasi pemiliknya dan merupakan data pribadi.
- **Data pribadi yang bersifat spesifik**: data dan informasi kesehatan, data biometrik, data genetika, catatan kejahatan, data anak, data keuangan pribadi, dan data lain sesuai peraturan. Pemrosesannya membawa risiko lebih tinggi dan kewajiban yang lebih ketat.

> **DEFINISI.** *Subjek data pribadi* adalah orang yang menjadi pokok data itu. *Pengendali data pribadi* menentukan tujuan dan cara pemrosesan. *Prosesor data pribadi* memproses data atas nama pengendali. *Pemrosesan* mencakup seluruh riwayat hidup data: pemerolehan dan pengumpulan, pengolahan dan penganalisisan, penyimpanan, perbaikan dan pembaruan, penampilan, pengiriman, serta penghapusan atau pemusnahan.

Bagi platform seperti Profitina, platform itu adalah pengendali atas akun para penggunanya; perusahaan *hosting* atau layanan pengiriman email yang menangani data atas namanya adalah

prosesor. Pembedaan ini penting karena pengendalilah yang memikul kewajiban, dan pengendali pula yang harus memastikan prosesornya memenuhi kewajiban itu.

11.2.3 Asas-asasnya

Pasal 16 mendaftar asas yang wajib diikuti setiap pemrosesan [2]. Diringkas, asas-asas itu menyatakan bahwa pengumpulan dilakukan secara terbatas dan spesifik, sah secara hukum, dan transparan; bahwa pemrosesan sesuai dengan tujuannya; bahwa hak subjek dijamin; bahwa data akurat, lengkap, tidak menyesatkan, mutakhir; bahwa data dilindungi dari pengaksesan, pengungkapan, pengubahan, penyalahgunaan, perusakan, dan penghilangan yang tidak sah; bahwa tujuan, kegiatan pemrosesan, dan kegagalan pelindungan diberitahukan; bahwa data dimusnahkan atau dihapus setelah masa retensinya berakhir atau atas permintaan subjek; dan bahwa pemrosesan dilakukan secara bertanggung jawab dan dapat dibuktikan secara jelas.

Klausa terakhir, *dapat dibuktikan*, adalah klausa milik insinyur. Sistem yang melindungi data tetapi tidak dapat menunjukkan bahwa ia melindunginya, dengan catatan siapa menyetujui apa dan kapan, atau dengan uji yang membuktikan bahwa sebuah penghapusan benar-benar menghapus, membuat pemiliknya tak mampu menjawab pertanyaan paling dasar yang dapat diajukan regulator maupun pengguna.

11.2.4 Dasar pemrosesan

Pemrosesan memerlukan dasar. Pasal 20 mendaftar enam: persetujuan yang sah secara eksplisit dari subjek untuk satu atau beberapa tujuan tertentu; pemenuhan kewajiban perjanjian yang salah satu pihaknya adalah subjek; pemenuhan kewajiban hukum pengendali; pelindungan kepentingan vital subjek; pelaksanaan tugas dalam rangka kepentingan umum, pelayanan publik, atau pelaksanaan kewenangan pengendali berdasarkan peraturan; dan pemenuhan kepentingan lain yang sah, dengan memperhatikan keseimbangannya dengan hak subjek [2].

Ketika dasarnya adalah persetujuan, Pasal 21 mendaftar apa yang harus disampaikan kepada orang itu terlebih dahulu: legalitas pemrosesan, tujuannya, jenis dan relevansi data, jangka waktu retensi dokumen yang memuatnya, rincian informasi yang dikumpulkan, jangka waktu pemrosesan, dan hak subjek. Pasal 22 menyatakan bahwa persetujuan diberikan secara tertulis atau terekam, secara elektronik maupun nonelektronik, dan bahwa permintaan persetujuan harus dapat dibedakan secara jelas dari hal lain, dibuat dalam format yang dapat dipahami dan mudah diakses, serta menggunakan bahasa yang sederhana dan jelas; persetujuan yang tidak memenuhi ketentuan itu batal demi hukum [2]. Subbab 11.6 mengubah kalimat-kalimat ini menjadi formulir pendaftaran dan sebuah catatan di basis data.

11.2.5 Hak subjek data pribadi

Bab IV undang-undang, Pasal 5 sampai 15, memberi orang serangkaian hak [2]. Yang biasanya dibutuhkan pengguna sehari-hari adalah:

- **Informasi** (Pasal 5): mengetahui kejelasan identitas pengendali, dasar kepentingan hukum dan tujuan permintaan serta penggunaan data, dan akuntabilitas pengendali.
- **Pembetulan** (Pasal 6): melengkapi, memperbarui, dan memperbaiki kesalahan atau ketidakakuratan datanya.
- **Akses** (Pasal 7): mengakses dan memperoleh salinan datanya.
- **Penghapusan** (Pasal 8): mengakhiri pemrosesan, serta menghapus atau memusnahkan datanya.

- **Penarikan persetujuan** (Pasal 9).
- **Keberatan atas keputusan otomatis** (Pasal 10): mengajukan keberatan atas keputusan yang hanya didasarkan pada pemrosesan otomatis, termasuk pemrofilan, yang menimbulkan akibat hukum atau berdampak signifikan.
- **Portabilitas** (Pasal 13): memperoleh dan menggunakan datanya dalam bentuk yang sesuai dengan struktur atau format yang lazim digunakan dan dapat dibaca oleh sistem elektronik, serta mengirimkannya ke pengendali lain.

Undang-undang juga memberi pengendali tenggat. Misalnya, pengendali wajib memberikan akses kepada subjek atas datanya paling lambat 3×24 jam sejak permintaan diterima (Pasal 32) [2]. Hak dengan tenggat 72 jam adalah hak yang harus dapat dipenuhi tim kecil tanpa proyek khusus, dan itu salah satu alasan terkuat untuk menjadikannya swalayan.

11.2.6 Kewajiban pengendali

Kewajiban pengendali mengisi sebagian besar undang-undang. Yang membentuk rancangan sebuah sistem adalah [2]:

- **Keamanan** (Pasal 35): melindungi dan memastikan keamanan data pribadi dengan langkah teknis operasional yang sepadan dengan risikonya.
- **Mencegah akses tidak sah** (Pasal 39): "Pengendali Data Pribadi wajib mencegah Data Pribadi diakses secara tidak sah", dengan sistem keamanan yang andal.
- **Penilaian dampak** (Pasal 34): menilai dampak sebelum pemrosesan yang berisiko tinggi, seperti pemrosesan data spesifik berskala besar atau pemantauan sistematis.
- **Penghapusan dan pemusnahan** (Pasal 43 sampai 45): menghapus data yang tidak lagi diperlukan untuk tujuannya, ketika persetujuan ditarik, ketika subjek memintanya, atau ketika data diperoleh secara melawan hukum; lalu memberi tahu subjek.
- **Pemberitahuan kegagalan pelindungan** (Pasal 46): bila terjadi kegagalan pelindungan data pribadi, menyampaikan pemberitahuan tertulis kepada subjek dan lembaga paling lambat 3×24 jam, memuat data yang terungkap, kapan dan bagaimana data terungkap, serta upaya penanganan dan pemulihannya.
- **Pejabat pelindungan data** (Pasal 53): wajib ditunjuk untuk pemrosesan bagi pelayanan publik, untuk pemantauan sistematis berskala besar, dan untuk pemrosesan berskala besar atas data spesifik atau data terkait tindak pidana.
- **Transfer lintas negara** (Pasal 56): hanya boleh bila negara penerima memiliki tingkat pelindungan yang setara atau lebih tinggi, atau dengan pelindungan yang memadai, atau dengan persetujuan subjek.

11.2.7 Sanksi

Pelanggaran dikenai sanksi administratif menurut Pasal 57: peringatan tertulis, penghentian sementara kegiatan pemrosesan, penghapusan atau pemusnahan data, dan sanksi uang administratif paling tinggi 2% dari pendapatan tahunan [2]. Ketentuan pidana tersendiri, Pasal 67 sampai 73, mengancam perbuatan memperoleh, mengungkapkan, atau menggunakan data pribadi yang bukan miliknya secara melawan hukum, serta membuat data pribadi palsu, dengan pidana penjara empat sampai enam tahun dan pidana uang hingga enam miliar rupiah, bergantung pada perbuatannya [2]. Angka-angka ini layak diketahui bukan sebagai ancaman, melainkan sebagai ukuran seberapa serius undang-undang memperlakukan subjek data: sanksi uangnya diskalakan terhadap pendapatan, sehingga tidak menjadi remeh bagi perusahaan besar.

11.3 Posisi Hukum pada 2026

11.3.1 PP 33/2026, peraturan pelaksananya

Undang-undang seperti UU PDP menyatakan asas dan kewajiban; tata caranya diatur dalam peraturan pelaksana. Selama hampir empat tahun setelah pengesahan, peraturan pelaksana utamanya masih berupa rancangan. Hal itu berubah pada Juli 2026. *Peraturan Pemerintah Nomor 33 Tahun 2026 tentang Peraturan Pelaksanaan Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi* ditetapkan dan diundangkan pada 16 Juli 2026, dalam Lembaran Negara Tahun 2026 Nomor 88, dengan 225 pasal dalam 12 bab, dan mulai berlaku pada 16 Januari 2027 [3].

Bab-babnya mencakup klasifikasi data pribadi, dasar pemrosesan, hak subjek data, kewajiban pengendali dan prosesor, transfer lintas negara, pejabat pelindungan data, penilaian dampak pelindungan data, pengawasan kepatuhan, kerja sama internasional, serta sanksi administratif dan penyelesaian sengketa [3]. Satu kalimatnya merangkum seluruh disiplin rekayasa: "pengendali data pribadi wajib memiliki dasar pemrosesan data pribadi sebelum melakukan pemrosesan" [3]. Dengan kata lain: putuskan dan catat *mengapa* sebelum baris pertama ditulis.

Enam bulan antara pengundangan dan keberlakuan ada agar organisasi dapat menyesuaikan diri. Bagi sistem yang sejak awal dirancang menurut undang-undangnya, penyesuaian itu sebagian besar berupa pencocokan prosedur dengan rincian peraturan; bagi sistem yang tidak, penyesuaian itu dapat berarti membangun ulang sebagian darinya.

11.3.2 Lembaga pengawas

UU PDP (Pasal 58 sampai 61) membentuk sebuah lembaga, yang ditetapkan oleh dan bertanggung jawab kepada Presiden, untuk mengawasi kepatuhan, menjatuhkan sanksi administratif, dan menangani pengaduan. Pada September 2026 lembaga itu belum ada. Menurut Kementerian Komunikasi dan Digital (Komdigi), rancangan Peraturan Presiden (Perpres) pembentukannya telah melewati tahap perencanaan, penyusunan, dan harmonisasi, lalu diajukan untuk ditetapkan Presiden melalui surat tertanggal 20 Mei 2026 [4]. Sementara itu Komdigi menjalankan fungsi pengawasan: penanganan insiden, penerimaan pengaduan atas dugaan pelanggaran dan penyalahgunaan data, serta layanan konsultasi [4]. Para pengamat berhati-hati soal waktunya. Seorang pakar kebijakan yang dikutip pada September 2026 memperkirakan lembaga itu baru dapat menegakkan standar kepatuhan secara efektif "pada 2028, itu yang paling cepat", karena Perpres yang sudah ditandatangani pun harus disusul kerja pembentukan kelembagaan [4], [5].

Gambar 11.1 menyatukan tanggal-tanggal itu dengan tonggak Profitina sendiri.

Gambar 11.1 UU PDP dari pengesahan sampai peraturan pelaksananya, bersama tonggak kerja kepatuhan Profitina sendiri. Tanggal lembaga adalah perkiraan seorang pakar, bukan jadwal.

11.3.3 Artinya bagi tim kecil

Mudah sekali membaca "lembaganya belum beroperasi" sebagai "undang-undangnya belum ditegakkan". Cara baca itu keliru dalam dua hal. Pertama, kewajiban undang-undang telah berlaku sejak Oktober 2024 apa pun keadaan lembaganya, dan Komdigi mengawasi pada masa peralihan. Kedua, dan lebih penting bagi insinyur, kewajiban yang dirancang sejak awal itu murah, sedangkan kewajiban yang ditambahkan belakangan itu mahal. Log persetujuan yang mencatat setiap pendaftaran sejak peluncuran tidak dapat direkonstruksi setelah kejadian; tabel kata sandi yang penuh *hash* cepat tidak dapat ditingkatkan tanpa menunggu setiap pengguna login lagi. Saat membangun untuk hukum adalah sebelum pengguna pertama datang, dan Profitina melakukannya.

11.4 Privasi sejak Rancangan sebagai Metode Rekayasa

11.4.1 Gagasannya

"Privasi sejak rancangan" (*privacy by design*) adalah istilah yang dicetuskan dan dikembangkan oleh Ann Cavoukian, saat itu Komisioner Informasi dan Privasi Ontario, yang merangkumnya dalam tujuh prinsip dasar: proaktif, bukan reaktif; privasi sebagai pengaturan bawaan; privasi tertanam dalam rancangan; fungsionalitas penuh, jumlah-positif alih-alih jumlah-nol; keamanan ujung ke ujung sepanjang siklus hidup data; keterlihatan dan transparansi; dan penghormatan kepada pengguna [6]. Prinsip-prinsip itu sudah lama dan umum. Yang membuatnya berguna adalah mengubah setiap prinsip menjadi keputusan yang dapat diambil, diuji, dan dipertanggungjawabkan oleh pengembang.

11.4.2 Minimalkan dulu

Data yang paling murah untuk dilindungi adalah data yang tidak pernah dikumpulkan. Pendaftaran Profitina meminta dua hal: alamat email dan kata sandi [1], [7]. Tidak ada kolom nomor telepon di bagian mana pun dari produk, dan alamat email tidak pernah dapat dipublikasikan di profil. *Handle* komunitas bersifat netral, secara bawaan satu huruf dan delapan digit heksadesimal, yang dapat diubah pengguna, dan diperiksa agar tidak berupa nomor pokok mahasiswa; nama disamarkan bagi pengunjung anonim. Tambahan profil, sebuah foto dan hingga empat tautan media sosial, masing-masing memerlukan persetujuannya sendiri, bersifat privat secara bawaan, dan dapat ditarik kapan saja.

Setiap pilihan itu menyingkirkan satu kelas risiko. Tabel alamat email yang bocor itu buruk; tabel alamat email, nomor telepon, nama lengkap, dan nomor pokok mahasiswa yang bocor jauh lebih buruk. Setiap kolom yang tidak dikumpulkan adalah kolom yang tidak dapat bocor, tidak dapat diminta paksa, dan tidak memerlukan aturan retensi.

11.4.3 Tuliskan: catatan pemrosesan

"Dapat dibuktikan" dalam Pasal 16 menuntut catatan tentang apa yang diproses, mengapa, atas dasar apa, berapa lama, dan oleh siapa. Nama lazim catatan itu adalah *record of processing activities*, ROPA. Profitina memiliki aturan internal yang mengikatnya pada pengembangan: setiap fitur baru yang menyentuh data pribadi wajib memperbarui ROPA dan, bila perlu, Kebijakan Privasi, dalam paket kerja yang sama [1]. Sebuah fitur belum selesai selama dokumen privasinya belum ada, persis seperti ia belum selesai selama ujinya belum ada.

11.4.4 Privat secara bawaan, di mana-mana

Minimalisasi berlanjut setelah pendaftaran. Riwayat pemindaian dan konten bersifat privat bagi pemiliknya. Sesi admin dan sesi pengguna saling meniadakan, sehingga seorang administrator tidak dapat menjelajah sebagai dirinya dan sebagai pengguna pada saat yang sama [7]. Pesan langsung bersifat satu-lawan-satu dan administrator tidak dapat membacanya, dengan satu pengecualian sempit: salinan beku dari pesan yang *dilaporkan* oleh salah satu pesertanya. Galat server dicatat tanpa isi mentah permintaan maupun *header*-nya, dengan alamat IP di-*hash*, dan log galat dihapus setelah 90 hari oleh tugas pembersihan yang benar-benar berjalan [1]. Frasa "yang benar-benar berjalan" bukan hiasan. Aturan retensi yang hanya ada di dokumen kebijakan bukanlah aturan retensi; tugas terjadwal yang disertai uji, itulah aturan retensi.

11.4.5 Satu rumah untuk teks hukum

Bab 10 menceritakan bagaimana Kebijakan Privasi dipindahkan ke satu rumah di aplikasi, sementara situs pemasaran meringkas dan menautkannya [1]. Prinsip rekayasa di baliknya, *hapus duplikasinya, jangan menyinkronkannya*, juga merupakan prinsip privasi: pengguna yang membaca dua versi kebijakan tidak dapat tahu versi mana yang mengikat perusahaan, dan perusahaan yang memelihara dua versi suatu hari akan membantah dirinya sendiri. Subbab 11.9 kembali ke halaman-halaman legal.

11.5 Kata Sandi: Mengapa Argon2id

11.5.1 Jangan pernah menyimpan kata sandinya

Sistem yang memeriksa kata sandi harus mengingat sesuatu tentang setiap kata sandi, tetapi tidak boleh mengingat kata sandi itu sendiri. Bila ia menyimpannya, siapa pun yang memperoleh basis datanya, entah penyerang, cadangan yang tercecer, atau orang dalam yang penasaran, akan memegang kata sandi setiap pengguna, dan karena orang memakai ulang kata sandi, kerusakannya akan menjalar ke email, bank, dan media sosial mereka.

Jawaban bakunya adalah **fungsi satu arah**: fungsi h yang mudah dihitung dan praktis mustahil dibalik. Saat pendaftaran sistem menyimpan $h(\text{password})$; saat login ia menghitung h dari apa yang diketik lalu membandingkannya. Enkripsi adalah alat yang salah, karena enkripsi dirancang untuk dibalik oleh siapa pun yang memegang kuncinya, dan kunci yang berada di server yang sama dengan basis data ikut tercuri bersamanya. Kebijakan Privasi Profitina menyatakan prinsip

itu di bagian keamanannya: kata sandi disimpan sebagai *hash* satu arah dengan algoritma modern (Argon2id atau bcrypt), tidak pernah sebagai teks biasa [7].

11.5.2 Mengapa *hash* yang *cepat* adalah fungsi satu arah yang salah

Hash serbaguna seperti SHA-256 memang satu arah, tetapi dirancang agar cepat, dan kecepatan persis yang diinginkan penyerang. Dengan tabel *hash* curian, penyerang tidak perlu membalik apa pun: ia cukup menebak, meng-*hash* setiap tebakan, dan mencari yang cocok. Bila sebuah kata sandi diambil dari himpunan N kemungkinan yang sama peluangnya dan penyerang dapat menghitung R *hash* per detik, waktu untuk mencoba semuanya adalah

$$T = \frac{N}{R} \tag{11.1}$$

dan rata-rata separuh dari waktu itu cukup untuk menemukan satu kata sandi tertentu. Pembela tidak dapat mengubah N, yang dipilih pengguna, tetapi dapat mengubah R dengan memilih fungsinya. Itulah seluruh gagasan *fungsi hash kata sandi*: fungsi satu arah yang sengaja lambat dengan biaya yang dapat diatur.

Tabel 11.1 memperlihatkan dampak pilihan itu pada satu mesin biasa, yaitu server virtual berprosesor Xeon 2,1 GHz yang dipakai menjalankan kode buku ini, diukur pada satu inti dengan Python 3.11.

Tabel 11.1 Kecepatan terukur empat cara meng-*hash* kata sandi pada satu inti CPU mesin yang dipakai untuk buku ini (September 2026). Penyerang sungguhan jauh lebih cepat dengan perangkat keras khusus; yang penting adalah perbandingannya.

Fungsi (pengaturan biaya)	Waktu per *hash*	*Hash* per detik, R
SHA-256, dengan *salt*, satu kali	0,6 µs	sekitar 1.670.000
PBKDF2-HMAC-SHA256, 100.000 iterasi	31 ms	sekitar 32
PBKDF2-HMAC-SHA256, 600.000 iterasi	230 ms	sekitar 4,3
Argon2id, 19 MiB, 2 putaran, 1 jalur	39 ms	sekitar 26

CONTOH: Kata sandi berupa 8 huruf kecil memiliki $N = 26^8 = 208\,827\,064\,576$ kemungkinan. Dengan SHA-256 ber-*salt* pada $R \approx 1.67 \times 10^6$ per detik, Persamaan (11.1) memberikan $T \approx 1.25 \times 10^5$ detik, sekitar 35 jam pada satu inti. Dengan PBKDF2 600.000 iterasi, $R \approx 4.35$, sehingga $T \approx 4.8 \times 10^{10}$ detik, sekitar 1.500 tahun pada inti yang sama. Kata sandinya tidak berubah; yang berubah hanya biaya setiap tebakan.

Angka-angka dalam contoh itu terlalu menyanjung pembela, karena penyerang tidak memakai satu proses Python di satu inti. Prosesor grafis dan perangkat keras khusus menghitung SHA-256 dan PBKDF2 ribuan kali lebih cepat. Yang jauh kurang dikuasai perangkat seperti itu adalah fungsi yang juga memerlukan banyak *memori* untuk setiap tebakan, dan di situlah Argon2 berperan.

11.5.3 *Salt*

Sebelum semua itu, setiap kata sandi diberi **salt** (garam): nilai acak, berbeda untuk setiap kata sandi yang disimpan, yang di-*hash* bersama kata sandi dan disimpan di sebelah hasilnya. Tanpa *salt*, dua pengguna dengan kata sandi yang sama memiliki *hash* yang sama, dan penyerang dapat menghitung sekali saja tabel *hash* kata sandi yang umum lalu mencocokkan semua pengguna sekaligus. Dengan *salt*, setiap tebakan harus dihitung terpisah untuk setiap pengguna. Pedoman NIST terkini mensyaratkan bahwa kata sandi "SHALL be salted and hashed using a suitable

password hashing scheme", dengan *salt* sekurang-kurangnya 32 bit [8]; lab ini memakai 128 bit (16 bait).

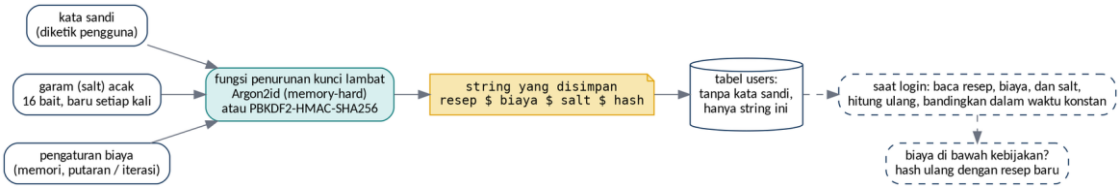
11.5.4 Argon2id

Argon2 memenangi Password Hashing Competition pada 2015 [9] dan dibakukan dalam RFC 9106 pada September 2021 [10]. Argon2 bersifat *memory-hard*: menghitung satu *hash* memerlukan pengisian dan pembacaan berulang atas satu blok memori yang ukurannya dipilih pembela, sehingga penyerang yang ingin menjalankan banyak tebakan secara paralel memerlukan memori sebesar itu untuk setiap tebakan. Argon2 memiliki tiga varian; **Argon2id**, hibrida yang tahan terhadap serangan kanal samping maupun serangan dengan perangkat keras khusus, adalah varian yang dianjurkan RFC 9106 sebagai bawaan [10].

Argon2id memiliki tiga parameter biaya: memori m, jumlah putaran t, dan paralelisme p. *Password Storage Cheat Sheet* dari OWASP mendaftar lima konfigurasi yang dianggapnya sama kuat, dari $m = 47104$ KiB (46 MiB) dengan $t = 1$ sampai $m = 7168$ KiB (7 MiB) dengan $t = 5$, semuanya dengan $p = 1$; pilihan tengah yang lazim adalah $m = 19456$ KiB (19 MiB), $t = 2$, $p = 1$ [11]. Panduan yang sama menganjurkan 600.000 iterasi untuk PBKDF2-HMAC-SHA256 bila Argon2id tidak tersedia [11]. PBKDF2 sendiri didefinisikan dalam RFC 8018 [12]; fungsi itu tidak *memory-hard*, tetapi tersedia di pustaka standar Python (`hashlib.pbkdf2_hmac`) maupun Java (`PBKDF2WithHmacSHA256`), dan karena itulah lab G7 memakainya secara bawaan serta menawarkan Argon2id sebagai pilihan [13], [14].

11.5.5 *Hash* yang tersimpan menyebut resepnya sendiri

Gambar 11.2 memperlihatkan seluruh jalur dari kata sandi yang diketik sampai string yang disimpan, dan kembali lagi saat login.



Gambar 11.2 Menyimpan kata sandi. *Salt* acak dan pengaturan biaya masuk ke fungsi penurunan kunci yang lambat; string yang disimpan mencatat resep, biaya, *salt*, dan hasilnya, sehingga login dapat menghitung ulang dan resep yang usang dapat ditingkatkan.

String yang disimpan bukan sekadar *hash*. Ia adalah catatan kecil: nama algoritma, pengaturan biayanya, *salt*, dan hasilnya, misalnya

```
pbkdf2_sha256$600000$<salt, base64>$<hash, base64>
$argon2id$v=19$m=19456,t=2,p=1$<salt, base64>$<hash, base64>
```

Karena setiap *hash* yang tersimpan menyebut resepnya sendiri, resep itu dapat berganti tanpa hari perpindahan besar-besaran. Ketika kebijakan beralih ke pengaturan yang lebih kuat, *hash* lama tetap dapat diverifikasi dengan pengaturan lamanya, dan pada login berhasil berikutnya, satu-satunya saat sistem melihat kata sandi asli lagi, sistem menghitung ulang *hash* dengan resep baru dan mengganti string lama. Inilah pola *upgrade on login* (peningkatan saat login), dan Profitina memakainya: *hash* lama ditingkatkan menjadi Argon2id ketika pemiliknya login [1]. Lab G7 memperagakannya dengan beralih dari 100.000 ke 600.000 iterasi PBKDF2 (atau ke Argon2id) di tengah demonya.

Dua rincian kecil melengkapi gambarannya. Perbandingan *hash* yang dihitung ulang dengan yang tersimpan harus memakan waktu yang sama apa pun masukannya, agar lamanya login yang gagal

tidak mengungkap berapa bait yang cocok; `hmac.compare_digest` di Python dan `MessageDigest.isEqual` di Java melakukan hal itu. Dan batas panjang itu penting: SP 800-63B-4 dari NIST, yang terbit pada Agustus 2025, mensyaratkan kata sandi sekurang-kurangnya 15 karakter bila kata sandi adalah satu-satunya faktor, dan sekurang-kurangnya 8 bila dipakai bersama faktor lain, serta membatasi kegagalan berturut-turut paling banyak 100 per akun [8]. Lab G7 memakai batas minimum 15 karakter.

11.6 Persetujuan yang Bermakna

11.6.1 Dari Pasal 22 ke kotak centang

Subbab 11.2.4 mengutip undang-undang: persetujuan diberikan secara tertulis atau terekam; permintaannya harus dapat dibedakan, dapat dipahami, dan memakai bahasa yang sederhana; persetujuan yang tidak memenuhi syarat itu batal demi hukum [2]. Diterjemahkan ke dalam formulir pendaftaran, artinya empat hal:

1. **Tindakan tersendiri yang eksplisit.** Kotak centang yang dicentang sendiri oleh pengguna, bukan kalimat yang tersembunyi di syarat layanan bahwa “dengan mendaftar Anda setuju”.
2. **Tidak pernah tercentang lebih dulu.** Kotak yang datang dalam keadaan tercentang mencatat pilihan perancang, bukan pilihan pengguna. Kebijakan Privasi Profitina menyatakannya langsung: kotak persetujuan tidak dicentang otomatis [7].
3. **Berdasarkan informasi.** Pemberitahuan singkat dalam dua bahasa di samping kotak, dengan tautan ke kebijakan lengkap, yang mencakup apa yang didaftar Pasal 21.
4. **Tercatat.** Sebuah baris yang menyatakan siapa yang menyetujui, versi kebijakan mana, dan kapan.

Butir keempat itulah yang mengubah persetujuan dari rincian antarmuka menjadi bukti. Profitina mencatat setiap persetujuan dengan versi kebijakan dan waktunya, sebagai jejak audit [1]. Ketika Kebijakan Privasi berubah, seperti ketika versi 1.15 mulai berlaku pada 9 September 2026 [7], log itu menunjukkan dengan tepat versi mana yang diterima setiap orang.

11.6.2 Merancang log persetujuan

Log persetujuan adalah daftar peristiwa, bukan satu kolom pada catatan pengguna. Perbedaannya penting. Nilai boolean `consented = true` tidak mengatakan apa pun tentang *apa* yang disetujui, *kapan*, atau apakah kemudian ditarik; daftar peristiwa menjawab ketiganya dan tidak pernah kehilangan riwayat. Log di lab G7 memiliki satu baris per peristiwa dengan lima kolom: subjek, peristiwa (`given` atau `withdrawn`), versi kebijakan, waktu, dan *hash* alamat IP asal persetujuan itu.

Mengapa *hash* alamat IP, dan mengapa *hash* berkunci? Alamat IP itu sendiri adalah data pribadi. Menyimpannya sebagai teks biasa dalam log yang tujuannya membuktikan persetujuan justru menciptakan risiko privasi baru di dalam upaya mendokumentasikan privasi. SHA-256 biasa atas alamat IPv4 tidak banyak menolong, karena hanya ada sekitar empat miliar alamat dan semuanya dapat di-*hash* dalam hitungan menit. *Hash* berkunci (HMAC) dengan rahasia yang disimpan di luar basis data tidak dapat dibalik dengan cara itu oleh orang yang hanya memegang basis datanya. Lab ini menyimpan 16 digit heksadesimal pertama, cukup untuk membedakan peristiwa, dan menghapus bahkan itu ketika akun dihapus.

11.6.3 Penarikan

Pasal 9 memberi hak untuk menarik persetujuan. Menarik harus semudah memberi, dan penarikan pun harus dicatat. Di lab G7, seperti di Profitina, penarikan yang definitif adalah penghapusan akun: log bertambah satu peristiwa `withdrawn`, dan peristiwa `given` yang lebih awal tetap ada sebagai bukti bahwa pemrosesan sampai saat itu sah, tetapi keduanya diberi label ulang dengan pengenal *tombstone* (nisan) yang anonim dan dilucuti *hash* IP-nya, sehingga bukti itu tidak lagi mengidentifikasi orangnya (Subbab 11.8.4).

Persetujuan juga bersifat granular sejauh produknya granular. Halaman profil Profitina meminta persetujuan per butir, satu untuk foto dan satu untuk setiap tautan media sosial, dan masing-masing dapat ditarik sendiri-sendiri [1]. Aturan umumnya: satu persetujuan mencakup satu tujuan, dan seseorang tidak boleh dipaksa menerima satu hal untuk mendapatkan hal lain yang tidak berkaitan.

11.6.4 Anak

Pasal 25 mensyaratkan persetujuan orang tua atau wali untuk pemrosesan data anak [2]. Profitina tidak ditujukan untuk anak, dan kebijakannya menyatakan demikian, dengan persetujuan orang tua bila diwajibkan hukum [7]. Pelajaran rekayasanya adalah bahwa pertanyaan tentang usia itu sendiri merupakan data pribadi; layanan yang tidak ditujukan untuk anak sebaiknya menyatakannya terus terang alih-alih mengumpulkan tanggal lahir yang tidak diperlukannya.

11.7 Pembatasan Laju Login

11.7.1 Serangannya

Dua serangan mengincar formulir login. ***Brute force*** mencoba banyak kata sandi terhadap satu akun. ***Credential stuffing*** mencoba pasangan nama pengguna dan kata sandi yang bocor dari situs *lain*, dengan bertaruh bahwa orang memakai ulang kata sandi; setiap akun hanya dicoba beberapa kali, tetapi jutaan akun dicoba [15]. *Hash* yang lambat (Subbab 11.5) membuat setiap percobaan juga memakan waktu CPU pembela, dan itu alasan ketiga untuk membatasi percobaan: tanpa batas, banjir permintaan login berubah menjadi serangan penolakan layanan terhadap prosesor server.

Pasal 39 mewajibkan pengendali mencegah akses tidak sah [2]; pembatasan laju adalah salah satu langkah teknis baku untuk memenuhinya. Kebijakan Privasi Profitina menggambarkan langkahnya sendiri dengan istilah ini: pembatasan laju login terhadap *brute force* dan *credential stuffing*, dengan batas per identitas akun dan, bagi pengunjung anonim, per alamat IP, yang disimpan hanya di memori server dan tidak pernah ditulis ke basis data [7]. Metodenya sama apa pun angkanya, dan bab ini mengajarkannya dengan angka lab sendiri yang sengaja kecil.

11.7.2 Jendela tetap dan jendela geser

Pembatas paling sederhana menghitung kegagalan per menit jam dan mengatur ulang hitungan di awal setiap menit: *jendela tetap* (*fixed window*). Kelemahannya ada di perbatasan. Dengan batas 3 per menit, penyerang dapat gagal 3 kali pada 09:00:59 dan 3 kali lagi pada 09:01:00: enam percobaan dalam dua detik, dua kali laju yang dimaksud.

Jendela geser (*sliding window*) tidak memiliki perbatasan. Ia mengingat waktu setiap kegagalan terbaru untuk sebuah kunci, dan sebelum setiap percobaan ia melupakan kegagalan yang lebih

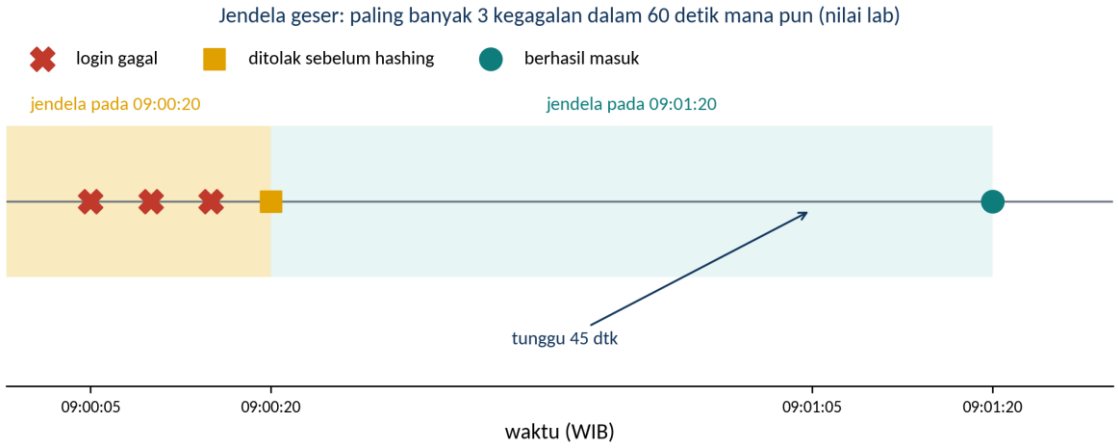
tua dari jendela. Misalkan $F_k(t)$ adalah himpunan waktu kegagalan untuk kunci k di dalam jendela w sebelum waktu kini t, dan n adalah jumlah kegagalan maksimum yang diizinkan:

$$F_k(t) = \{\, s \in \text{failures}(k) : t - w < s \le t \,\}, \qquad \text{blocked}(k,t) \Leftrightarrow |F_k(t)| \ge n \tag{11.2}$$

Ketika sebuah kunci terblokir, kunci itu boleh mencoba lagi begitu kegagalan tertua yang diingatnya keluar dari jendela:

$$\text{retry_after}(k,t) = \min F_k(t) + w - t \tag{11.3}$$

Gambar 11.3 mengikuti demo lab dengan $n = 3$ dan $w = 60$ detik.



Gambar 11.3 Pembatas berjendela geser dengan nilai-nilai lab. Tiga kegagalan pada 09:00:05, 09:00:10, dan 09:00:15 memblokir percobaan keempat pada 09:00:20, bahkan dengan kata sandi yang benar; blokir terangkat ketika kegagalan pertama keluar dari jendela pada 09:01:05.

> **CONTOH:** Kegagalan tercatat pada $s = 5{,}10{,}15$ detik setelah 09:00:00. Pada $t = 20$, $F_k(20) = \{5{,}10{,}15\}$ memiliki 3 unsur, sehingga menurut Persamaan (11.2) kunci itu terblokir, dan menurut Persamaan (11.3) ia harus menunggu $5 + 60 - 20 = 45$ detik, persis seperti yang dicetak demo. Pada $t = 80$, jendelanya adalah $(20{,}80]$, $F_k(80) = \emptyset$, dan login dilanjutkan.

Menyimpan waktu-waktu kegagalan dalam antrean dua ujung (*deque*) membuat setiap pemeriksaan murah: kegagalan baru masuk dari kanan, yang kedaluwarsa keluar dari kiri, dan setiap waktu masuk dan keluar antrean satu kali saja, sehingga kerja per percobaan secara amortisasi adalah $O(1)$, dengan paling banyak n waktu disimpan per kunci.

11.7.3 Empat rincian yang penting

Dua kunci, bukan satu. Membatasi hanya per akun membiarkan serangan *credential stuffing* menyebar tebakannya ke banyak akun, masing-masing beberapa kali; membatasi hanya per alamat IP membiarkan serangan terdistribusi memakai banyak alamat terhadap satu akun. Lab ini memeriksa keduanya, kunci `acct:` untuk alamat email dan kunci `ip:` untuk alamat IP (yang di-*hash* berkunci), dan menolak bila salah satunya terblokir.

Tolak sebelum meng-*hash*. Percobaan yang terblokir ditolak *sebelum* kata sandi diverifikasi. Bila tidak, pembatas itu tetap membiarkan penyerang menghabiskan CPU server dengan *hash* yang lambat, dan kata sandi yang benar tetap menghasilkan waktu respons yang berbeda dari kata sandi yang salah.

Katakan hal yang sama. “Wrong email or password” adalah satu-satunya pesan kegagalan, baik untuk email yang tidak dikenal maupun kata sandi yang salah, agar formulir login tidak memberi

tahu penyerang alamat email mana yang memiliki akun. Dengan alasan yang sama lab ini memverifikasi *hash* tiruan bila emailnya tidak dikenal, sehingga kedua kasus memakan waktu yang kurang lebih sama.

Simpan di memori. Catatan pembatas pada dasarnya berumur pendek. Menyimpannya di memori, seperti yang dilakukan Profitina [7], berarti catatan itu lenyap ketika tidak lagi berguna, dan tidak pernah menjadi satu lagi tabel data pribadi di dalam cadangan.

11.8 Hak yang Benar-Benar Dapat Dijangkau

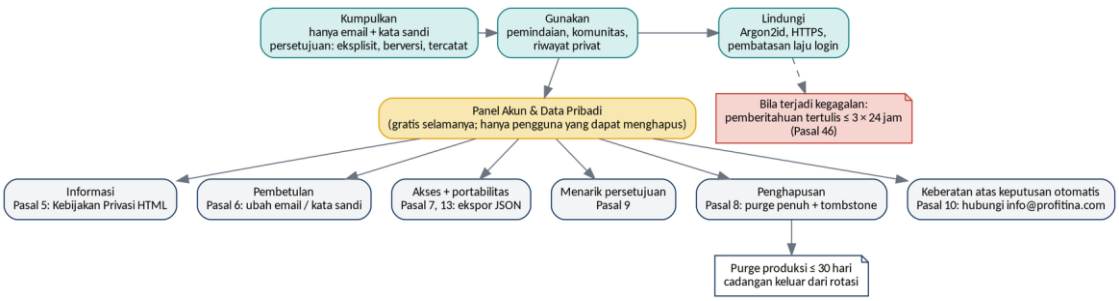
11.8.1 Dari pasal ke tombol

Sebuah hak dapat dijangkau bila orang biasa dapat memakainya, tanpa perlu mengetahui hukumnya, tanpa perlu menulis surat kepada siapa pun, dan tanpa menunggu. **Tabel 11.2** memetakan hak-hak di Subbab 11.2.5 ke fitur yang disediakan Profitina [1], [7].

Tabel 11.2 Hak subjek data menurut UU PDP dan fitur swalayan, atau saluran, yang dipakai pengguna Profitina untuk menjalankan masing-masing hak.

Hak (UU PDP)	Cara pengguna Profitina memakainya
Informasi (Pasal 5)	Kebijakan Privasi serta Syarat dan Ketentuan sebagai halaman HTML dwibahasa di `app.profitina.com/legal/`, ditautkan dari setiap halaman
Pembetulan (Pasal 6)	Ubah email atau kata sandi di panel *Akun & Data Pribadi*, dikonfirmasi dengan kata sandi
Akses (Pasal 7)	Unduh seluruh data akun sebagai JSON dari panel yang sama
Penghapusan (Pasal 8)	Hapus akun secara permanen dari panel, dikonfirmasi dengan kata sandi
Penarikan persetujuan (Pasal 9)	Persetujuan per butir profil dapat dimatikan; menghapus akun menarik sisanya
Keberatan atas keputusan otomatis (Pasal 10)	Hubungi info@profitina.com; keputusan produk yang menyangkut pengguna diambil oleh manusia
Portabilitas (Pasal 13)	Ekspor JSON terstruktur, berformat lazim, dan dapat dibaca mesin

Gambar 11.4 memperlihatkan hak-hak yang sama dalam riwayat hidup sebuah akun, dari pengumpulan sampai penghapusan, beserta kewajiban yang mengelilinginya.



Gambar 11.4 Riwayat hidup data pribadi di Profitina: pengumpulan minimal, pemakaian yang terlindungi, hak yang dijalankan dari satu panel, penghapusan yang disusul *purge* produksi, dan kewajiban pemberitahuan kegagalan pelindungan yang melatari semuanya.

Tiga komitmen membingkai panel itu. Hak-hak itu, bersama keamanan dan pemulihan akun, **gratis selamanya**; tidak pernah menjadi fitur Pro [1]. **Hanya pengguna yang dapat memulai penghapusan**: administrator tidak pernah menghapus akun "atas nama pengguna", karena penghapusan yang dapat dimulai orang lain juga dapat dimulai orang lain karena kekeliruan atau itikad buruk [7]. Dan setiap tindakan yang merusak **dikonfirmasi dengan kata sandi**, sehingga

sesi yang dibiarkan terbuka di komputer bersama tidak dapat dipakai untuk menghapus atau mengalihkan akun.

11.8.2 Sebuah koreksi yang jujur

Riwayat Profitina sendiri menyimpan pelajaran yang lebih berharga daripada tabel mana pun. Mekanisme ekspor, penggantian, dan penghapusan sudah ada di kode aplikasi sebelum 19 Agustus 2026. Yang belum ada adalah tombolnya: fungsi-fungsi itu dapat dijangkau lewat antarmuka pemrograman, tetapi pengguna antarmuka web tidak punya cara untuk menemukannya. Pada tanggal itu sebuah panel khusus, *Akun & Data Pribadi*, dirilis di tab *Pantau*, dan baru sejak itu benar untuk menyebut hak-hak itu swalayan [1].

Catatan tim sendiri menyatakannya terus terang, dan pelajarannya berlaku umum. Insinyur yang melihat kode melihat bahwa hak itu sudah diterapkan. Pengguna yang melihat layar melihat bahwa hak itu tidak ada. Undang-undang ditulis untuk pengguna. Fitur yang tidak dapat ditemukan orang yang dituju tidak ada bagi orang itu, dan rangkaian uji yang memanggil fungsinya secara langsung tidak akan pernah menyadarinya. Obatnya adalah uji ujung ke ujung (*end-to-end*) yang dimulai dari tempat pengguna memulai, yaitu layar, jenis pemeriksaan tingkat peramban, dengan Playwright yang mengendalikan Chromium, yang dipakai rangkaian uji Profitina.

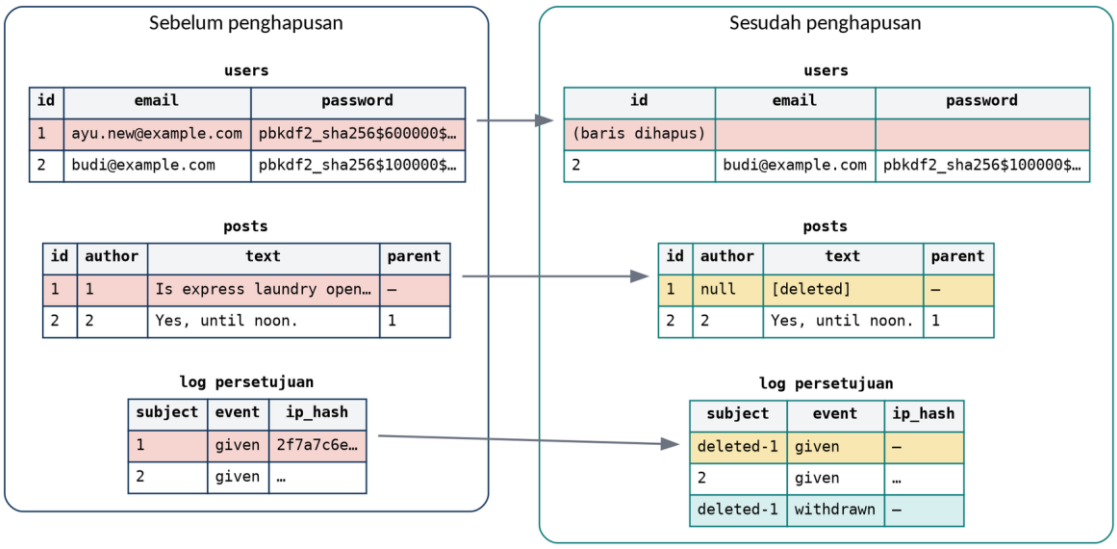
11.8.3 Ekspor: semuanya, dan hanya milik saya

Ekspor untuk hak akses dan portabilitas memiliki dua cara gagal, dan uji yang baik memeriksa keduanya. Ekspor dapat melewatkan sesuatu: tabel yang ditambahkan belakangan tanpa pernah diberitahukan kepada fungsi ekspor. Dan ekspor dapat memuat terlalu banyak: balasan milik pengguna lain, atau *hash* kata sandi internal, yang bukan data pengguna dalam arti yang berguna dan merupakan risiko keamanan dalam berkas yang diunduh. Uji di lab G7 memeriksa bahwa ekspor Ayu memuat alamat emailnya, peristiwa persetujuannya, dan tulisannya, serta tidak memuat tulisan Budi maupun *hash* kata sandi apa pun.

11.8.4 Penghapusan: *purge*, dan tinggalkan *tombstone* hanya di tempat orang lain membutuhkannya

Penghapusan adalah hak yang paling sulit diterapkan dengan benar, karena data pribadi menyebar. Alamat email sebuah akun ada di tabel pengguna; tulisannya ada di tabel tulisan; peristiwa persetujuannya ada di log persetujuan; sesinya ada di penyimpanan sesi; catatan login gagalnya mungkin ada di pembatas; pengenalnya mungkin dirujuk balasan orang lain. Penghapusan yang hanya menghapus baris pengguna meninggalkan sebagian besar data pribadinya.

Penghapusan di Profitina membersihkan data pribadi di semua modul dan meninggalkan *tombstone* yang anonim [1]. *Tombstone* memecahkan dilema nyata. Misalkan Ayu bertanya dan Budi menjawab. Bila pertanyaan Ayu lenyap, jawaban Budi menunjuk ke kekosongan dan utasnya putus; bila pertanyaan itu tetap ada dengan namanya, Ayu belum terhapus. *Tombstone* menyimpan cangkang kosong, tanpa pembuatnya dan dengan teks diganti `[deleted]`, *hanya* di tempat konten orang lain menunjuk kepadanya, dan menghapus semua yang lain. **Gambar 11.5** memperlihatkan aturan itu diterapkan pada tiga tabel di lab.



Gambar 11.5 Penghapusan di lab G7. Baris pengguna hilang; tulisan yang dibalas pengguna lain menjadi cangkang anonim; peristiwa persetujuan hanya menyimpan pengenal anonim, versi, dan waktu, dan satu peristiwa `withdrawn` ditambahkan.

Di Profitina, data produksi dihapus atau dianonimkan paling lambat 30 hari setelah akun dihapus, dan cadangan keluar dari siklus rotasi 90 harinya; log galat disimpan paling lama 90 hari [7]. Batas-batas itu bagian dari janji: penghapusan dari basis data yang sedang berjalan berlangsung seketika, tetapi sistem yang memiliki cadangan juga harus menyatakan kapan cadangannya lupa.

11.8.5 Ketika terjadi kegagalan

Pasal 46 mewajibkan pemberitahuan tertulis kepada pengguna yang terdampak dan kepada lembaga pengawas dalam 3 × 24 jam setelah terjadi kegagalan pelindungan data pribadi [2], dan kebijakan Profitina mengulangi komitmen itu [7]. Tujuh puluh dua jam itu singkat. *Runbook* Profitina membaginya sejak awal: dua jam pertama untuk penahanan, jam ke-2 sampai ke-12 untuk menilai apa yang terdampak, dan jam ke-12 sampai ke-48 untuk pemberitahuan, dengan templat kedua surat disiapkan jauh sebelum diperlukan [1]. Pokok rekayasanya sama seperti pada log persetujuan: pekerjaan yang membuat sebuah tenggat dapat dipenuhi dilakukan jauh sebelum tenggat itu mulai berjalan.

11.9 Halaman Legal HTML dan Satu Sumber Kebenaran

Kebijakan Privasi adalah wujud konkret hak atas informasi (Pasal 5), dan pemberitahuan yang disyaratkan Pasal 21 sebelum persetujuan. Kebijakan itu juga dokumen yang harus dapat dibaca pengguna di ponsel, dalam bahasanya sendiri, dan dapat ditautkan. Karena itulah Profitina menerbitkan teks-teks legalnya sebagai halaman HTML biasa, `/legal/privacy` dan `/legal/terms`, bukan sebagai berkas PDF untuk diunduh [1], [7].

HTML melayani setiap kebutuhan itu dengan lebih baik:

- **Terbaca di mana saja.** Halaman HTML menyesuaikan diri dengan layar kecil dan dapat dipakai pembaca layar serta fitur terjemahan peramban; PDF berukuran halaman A4 tidak melakukan keduanya dengan baik.
- **Dapat ditautkan sampai ke paragraf.** Setiap bagian dapat memiliki jangkar, sehingga pemberitahuan persetujuan dapat menautkan tepat ke paragraf yang diringkasnya.

- **Berversi.** Halaman itu menyatakan versinya dan tanggal mulai berlakunya. Saat buku ini disusun, Kebijakan Privasi berada pada versi 1.15, berlaku sejak 9 September 2026, dan Syarat dan Ketentuan pada versi 1.1, berlaku sejak 6 Agustus 2026 [7], [16]. Aplikasi mengetahui versi kebijakan yang berlaku, sehingga log persetujuan dapat mencatatnya pada setiap persetujuan.
- **Dwibahasa di satu tempat.** Teks bahasa Indonesia dan Inggris hidup bersama, di aplikasi yang sama, dan berubah bersama.
- **Satu rumah.** Situs pemasaran meringkas dan menautkan; ia tidak menyimpan salinan (Subbab 10.2.3).

Kebijakan itu sendiri mengikuti susunan yang diminta undang-undang: apa yang dikumpulkan (data akun, konten, data teknis, butir profil opsional, pesan, riwayat pemindaian, transaksi), mengapa dan atas dasar apa, berapa lama disimpan, siapa yang memprosesnya atas nama perusahaan dan dengan pelindungan apa bila lintas negara, *cookie* apa yang dipakai (hanya yang fungsional: sesi, bahasa, dan tanda bahwa pemberitahuan privasi sudah dibaca, tanpa *cookie* iklan), hak apa yang dimiliki pengguna dan cara menjalankannya, serta cara menghubungi perusahaan: info@profitina.com [7].

11.10 Lab G7: Log Persetujuan, Ekspor, dan Penghapusan

11.10.1 Apa yang akan kita bangun

Lab G7 adalah aplikasi kecil yang menyimpan akun pengguna, tulisan, dan log persetujuan di memori, serta menerapkan setiap langkah dari Subbab 11.5 sampai 11.8: *hash* kata sandi yang lambat dan ber-*salt* dengan peningkatan saat login, persetujuan yang eksplisit dan berversi, pembatas login berjendela geser pada dua kunci, ekspor sebagai JSON, penggantian email dan kata sandi dengan konfirmasi kata sandi, serta penghapusan dengan *tombstone*. Sebuah program demo menjalankan skenario dengan dua pengguna, Ayu dan Budi, dan sebuah rangkaian uji membuktikan sepuluh sifat, yang terpenting adalah bahwa setelah penghapusan tidak ada lagi yang bersifat pribadi tentang pengguna yang dihapus di tabel mana pun.

Aplikasi ini menyimpan tabelnya di memori dan memakai jam tiruan, sehingga setiap kali dijalankan ia mencetak hal yang persis sama dan pembaca dapat mengikuti waktunya secara manual. Tidak ada bagian yang bergantung pada basis data; Soal 11.12 mengusulkan pemindahan tabel-tabel itu ke SQLite, yang tersedia di pustaka standar Python. Alamat IP dalam demo, `203.0.113.7` dan `198.51.100.23`, berasal dari blok yang dicadangkan untuk dokumentasi, yang tidak pernah dimiliki mesin sungguhan [17].

Semuanya hanya memakai pustaka standar. Berkas Python memerlukan Python 3.11 atau yang lebih baru (ditulis untuk 3.14); `passwords.py` juga dapat memakai Argon2id bila paket opsional `argon2-cffi` terpasang [18]. Berkas Java dikompilasi dengan JDK 21 dan berjalan di JDK 25:

```
Chapter11/
  python/  passwords.py  rate_limit.py  privacy_app.py  demo.py
           test_privacy_app.py
  java/    Pbkdf2Hasher.java  SlidingWindowLimiter.java  PrivacyApp.java
           Json.java  Demo.java  PrivacyAppTest.java
```

`README.md` bab ini memuat perintah untuk Windows, macOS, dan Linux, baik di terminal maupun di VS Code.

11.10.2 *Hashing* kata sandi

Kode 11.1 memuat dua *hasher* dengan empat metode yang sama: `recognizes` (apakah string tersimpan ini milik saya?), `hash`, `verify`, dan `needs_rehash` (apakah string tersimpan ini lebih lemah daripada kebijakan saya saat ini?).

Kode 11.1 `passwords.py`: PBKDF2-HMAC-SHA256 dari pustaka standar, dan Argon2id melalui paket opsional `argon2-cffi`; setiap *hash* yang tersimpan menyebut resepnya.

```
"""Password hashing for the privacy lab: PBKDF2 from the standard library,
and Argon2id when the optional argon2-cffi package is installed.

A stored hash names its own recipe, so the recipe can change later:
    pbkdf2_sha256$600000$<salt, base64>$<hash, base64>
"""
import base64
import hashlib
import hmac
import secrets

OWASP_PBKDF2_SHA256 = 600_000       # OWASP Password Storage Cheat Sheet


class Pbkdf2Hasher:
    def __init__(self, iterations=OWASP_PBKDF2_SHA256):
        self.iterations = iterations

    def recognizes(self, stored):
        return stored.startswith("pbkdf2_sha256$")

    def hash(self, password, salt=None):
        salt = salt if salt is not None else secrets.token_bytes(16)
        key = hashlib.pbkdf2_hmac("sha256", password.encode("utf-8"), salt,
                                  self.iterations, dklen=32)
        return "pbkdf2_sha256$%d$%s$%s" % (
            self.iterations, base64.b64encode(salt).decode(),
            base64.b64encode(key).decode())

    def verify(self, stored, password):
        try:
            scheme, iters, salt, key = stored.split("$")
        except ValueError:
            return False
        if scheme != "pbkdf2_sha256":
            return False
        again = hashlib.pbkdf2_hmac("sha256", password.encode("utf-8"),
                                    base64.b64decode(salt), int(iters),
                                    dklen=32)
        # compare_digest takes the same time wherever the first difference is
        return hmac.compare_digest(again, base64.b64decode(key))

    def needs_rehash(self, stored):
        parts = stored.split("$")
        return parts[0] != "pbkdf2_sha256" or int(parts[1]) < self.iterations
```

Kode 11.1 (Lanjutan) *passwords.py: PBKDF2-HMAC-SHA256 dari pustaka standar, dan Argon2id melalui paket opsional argon2-cffi; setiap hash yang tersimpan menyebut resepnya.*

```
class Argon2Hasher:
    """Argon2id with an OWASP configuration (19 MiB, 2 passes, 1 lane)."""

    def __init__(self):
        from argon2 import PasswordHasher          # pip install argon2-cffi
        self.ph = PasswordHasher(time_cost=2, memory_cost=19456,
                                 parallelism=1)

    def recognizes(self, stored):
        return stored.startswith("$argon2")

    def hash(self, password):
        return self.ph.hash(password)

    def verify(self, stored, password):
        from argon2.exceptions import VerificationError, InvalidHashError
        try:
            return self.ph.verify(stored, password)
        except (VerificationError, InvalidHashError):
            return False

    def needs_rehash(self, stored):
        return not stored.startswith("$argon2id$") or \
            self.ph.check_needs_rehash(stored)
```

`Pbkdf2Hasher.hash` mengambil 16 bait acak dari `secrets`, modul yang ditujukan bagi keacakan yang peka keamanan, menurunkan 32 bait dengan `hashlib.pbkdf2_hmac`, lalu menulis string resep dari Subbab 11.5.5. `verify` memecah string itu, menghitung ulang dengan jumlah iterasi dan *salt* yang tersimpan, lalu membandingkan dengan `hmac.compare_digest`. `needs_rehash` menjawab ya bila string yang tersimpan sama sekali bukan PBKDF2 atau memiliki iterasi lebih sedikit daripada kebijakan saat ini. `Argon2Hasher` membungkus `PasswordHasher` milik `argon2-cffi` dengan konfigurasi OWASP 19 MiB, 2 putaran, dan 1 jalur, serta baru mengimpor paket itu ketika objeknya dibuat, sehingga sisa lab tetap berjalan tanpanya.

Kode 11.2 adalah *hasher* versi Java. `SecretKeyFactory` milik JDK menyediakan PBKDF2 dengan nama baku `PBKDF2WithHmacSHA256` [14]; `SecureRandom` menyediakan *salt* dan `MessageDigest.isEqual` perbandingan berwaktu konstan. Untuk kata sandi, *salt*, dan jumlah iterasi yang sama, kelas ini menghasilkan string yang persis sama dengan versi Python, dan salah satu uji memeriksanya.

Kode 11.2 `Pbkdf2Hasher.java`: resep PBKDF2 yang sama dengan `SecretKeyFactory` milik JDK.

```
import java.security.MessageDigest;
import java.security.SecureRandom;
import java.util.Base64;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;

/** PBKDF2-HMAC-SHA256 from the JDK. A stored hash names its own recipe:
 *  pbkdf2_sha256$600000$<salt, base64>$<hash, base64>
 *  (Argon2id needs a library such as Bouncy Castle; see the README.) */
public class Pbkdf2Hasher {
    static final int OWASP_PBKDF2_SHA256 = 600_000;   // OWASP cheat sheet
    private static final SecureRandom RANDOM = new SecureRandom();
    final int iterations;

    Pbkdf2Hasher() {
        this(OWASP_PBKDF2_SHA256);
    }

    Pbkdf2Hasher(int iterations) {
        this.iterations = iterations;
    }

    boolean recognizes(String stored) {
        return stored.startsWith("pbkdf2_sha256$");
    }

    static byte[] derive(String password, byte[] salt, int iterations) {
        try {
            PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), salt,
                                             iterations, 256);
            return SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256")
                                   .generateSecret(spec).getEncoded();
        } catch (Exception e) {
            throw new IllegalStateException(e);
        }
    }

    String hash(String password) {
        byte[] salt = new byte[16];
        RANDOM.nextBytes(salt);
        return hash(password, salt);
    }
```

Kode 11.2 (Lanjutan) *Pbkdf2Hasher.java: resep PBKDF2 yang sama dengan SecretKeyFactory milik JDK.*

```
    String hash(String password, byte[] salt) {
        Base64.Encoder b64 = Base64.getEncoder();
        return "pbkdf2_sha256$" + iterations + "$" + b64.encodeToString(salt)
                + "$" + b64.encodeToString(derive(password, salt, iterations));
    }

    boolean verify(String stored, String password) {
        String[] p = stored.split("\\$");
        if (p.length != 4 || !p[0].equals("pbkdf2_sha256")) return false;
        Base64.Decoder b64 = Base64.getDecoder();
        byte[] again = derive(password, b64.decode(p[2]), Integer.parseInt(p[1]));
        // isEqual takes the same time wherever the first difference is
        return MessageDigest.isEqual(again, b64.decode(p[3]));
    }

    boolean needsRehash(String stored) {
        String[] p = stored.split("\\$");
        return !p[0].equals("pbkdf2_sha256") || Integer.parseInt(p[1]) < iterations;
    }
}
```

11.10.3 Pembatas berjendela geser

Kode 11.3 adalah Persamaan (11.2) dan Persamaan (11.3) dalam bentuk kode. Setiap kunci dipetakan ke sebuah `deque` berisi waktu-waktu kegagalan; `_trim` membuang waktu yang sudah keluar dari jendela; `retry_after` mengembalikan 0 atau waktu tunggu; `record_failure` menambahkan; `forget` membuang sebuah kunci setelah login berhasil atau penghapusan.

Kode 11.3 `rate_limit.py`: pembatas berjendela geser untuk login yang gagal, dengan jam yang diberikan dari luar agar uji dapat mengendalikan waktu.

```
"""A sliding-window limiter for failed logins.

Each key (an account, or an IP address for anonymous traffic) keeps the
times of its recent failures. A key is blocked while it has max_failures
failures inside the last window_seconds. The numbers are the reader's
choice; the demo uses small ones so that the effect is easy to see.
"""
from collections import defaultdict, deque


class SlidingWindowLimiter:
    def __init__(self, max_failures, window_seconds, clock):
        self.max_failures = max_failures
        self.window = window_seconds
        self.clock = clock                      # a function returning seconds
        self.failures = defaultdict(deque)      # key -> times of failures

    def _trim(self, key):
        times, now = self.failures[key], self.clock()
        while times and times[0] <= now - self.window:
            times.popleft()                     # forget failures that aged out
        return times

    def retry_after(self, key):
        """0 if the key may try now, else seconds until it may."""
        times = self._trim(key)
        if len(times) < self.max_failures:
            return 0
        return int(times[0] + self.window - self.clock())

    def record_failure(self, key):
        self._trim(key).append(self.clock())

    def forget(self, key):
        self.failures.pop(key, None)
```

Jam itu berupa fungsi yang diberikan kepada konstruktor, bukan panggilan ke jam sistem di dalam kelas. Satu keputusan itulah yang membuat pembatas ini dapat diuji: demo dan uji memajukan waktu dengan memberi nilai sebuah angka, bukan dengan tidur selama satu menit. **Kode 11.4** adalah versi Java-nya, dengan `LongSupplier` sebagai jam dan satu `ArrayDeque` per kunci.

Kode 11.4 `SlidingWindowLimiter.java`: pembatas yang sama dalam Java.

```java
import java.util.ArrayDeque;
import java.util.Deque;
import java.util.HashMap;
import java.util.Map;
import java.util.function.LongSupplier;

/** A sliding-window limiter for failed logins. A key (an account, or an IP
 *  address for anonymous traffic) is blocked while it has maxFailures
 *  failures inside the last windowSeconds. The numbers are the reader's. */
public class SlidingWindowLimiter {
    final int maxFailures;
    final long window;
    final LongSupplier clock;                       // returns seconds
    final Map<String, Deque<Long>> failures = new HashMap<>();

    SlidingWindowLimiter(int maxFailures, long windowSeconds, LongSupplier clock) {
        this.maxFailures = maxFailures;
        this.window = windowSeconds;
        this.clock = clock;
    }

    private Deque<Long> trim(String key) {
        Deque<Long> times = failures.computeIfAbsent(key, k -> new ArrayDeque<>());
        long now = clock.getAsLong();
        while (!times.isEmpty() && times.peekFirst() <= now - window) {
            times.pollFirst();                      // failures that aged out
        }
        return times;
    }

    /** 0 if the key may try now, else seconds until it may. */
    long retryAfter(String key) {
        Deque<Long> times = trim(key);
        if (times.size() < maxFailures) return 0;
        return times.peekFirst() + window - clock.getAsLong();
    }

    void recordFailure(String key) {
        trim(key).addLast(clock.getAsLong());
    }

    void forget(String key) {
        failures.remove(key);
    }
}
```

11.10.4 Aplikasinya

Kode 11.5 adalah aplikasinya. Tabel-tabelnya adalah *dictionary* dan *list* biasa, dan setiap metodenya berpadanan dengan satu subbab dalam bab ini.

Kode 11.5 `privacy_app.py`: pendaftaran dengan persetujuan eksplisit, login berbatas laju dengan peningkatan *hash*, ekspor, pembetulan, dan penghapusan dengan *tombstone*.

```
"""G7: a privacy-by-design mini app (in memory, standard library only).

Explicit consent with a version and a time; passwords stored as slow salted
hashes; failed logins rate-limited; self-service export, correction and
erasure. Erasure purges every personal field and leaves an anonymous
tombstone only where other people's content still points to it.
"""
import hashlib
import hmac
import json
import secrets
from datetime import datetime, timedelta, timezone

from passwords import Pbkdf2Hasher
from rate_limit import SlidingWindowLimiter

WIB = timezone(timedelta(hours=7))
MIN_PASSWORD = 15          # NIST SP 800-63B-4, password as the only factor
GENERIC = "wrong email or password"


class PrivacyError(Exception):
    pass


class PrivacyApp:
    def __init__(self, clock, secret, hasher=None, max_failures=3,
                 window_seconds=60, policy_version="1.0"):
        self.clock = clock
        self.secret = secret           # keys the IP hashes; not in the tables
        self.hasher = hasher or Pbkdf2Hasher()
        self.hashers = [self.hasher]   # older recipes still readable
        self.limiter = SlidingWindowLimiter(max_failures, window_seconds,
                                            clock)
        self.policy_version = policy_version
        self.users = {}        # id -> email, password (hash), created
        self.consents = []     # subject, event, version, at, ip_hash
        self.posts = {}        # id -> author, text, parent, at
        self.sessions = {}     # token -> user id
        self.tombstones = []   # tombstone, at
        self.next_user = self.next_post = self.next_tomb = 1
        self.dummy = self.hasher.hash(secrets.token_hex(16))
```

Kode 11.5 (Lanjutan) *privacy_app.py: pendaftaran dengan persetujuan eksplisit, login berbatas laju dengan peningkatan hash, ekspor, pembetulan, dan penghapusan dengan tombstone.*

```
    # ---- helpers ------------------------------------------------------
    def stamp(self):
        return datetime.fromtimestamp(self.clock(), WIB).isoformat()

    def ip_hash(self, ip):
        mac = hmac.new(self.secret, ip.encode("utf-8"), hashlib.sha256)
        return mac.hexdigest()[:16]

    def find(self, email):
        for uid, user in self.users.items():
            if user["email"] == email:
                return uid
        return None

    def verify(self, stored, password):
        for h in self.hashers:
            if h.recognizes(stored):
                return h.verify(stored, password)
        return False

    def user_of(self, token):
        if token not in self.sessions:
            raise PrivacyError("not logged in")
        return self.sessions[token]

    def check_password(self, uid, password):
        if not self.verify(self.users[uid]["password"], password):
            raise PrivacyError("password confirmation failed")

    def use_hasher(self, hasher):
        """Adopt a new recipe; old hashes are upgraded at the next login."""
        self.hasher = hasher
        self.hashers.insert(0, hasher)

    # ---- registration and login ---------------------------------------
    def register(self, email, password, consent, ip):
        email = email.strip().lower()
        if "@" not in email:
            raise PrivacyError("not an email address")
        if len(password) < MIN_PASSWORD:
            raise PrivacyError("password needs at least %d characters"
                               % MIN_PASSWORD)
        if consent is not True:                 # never assumed, never default
            raise PrivacyError("consent is required")
        if self.find(email) is not None:
            raise PrivacyError("email already registered")
```

Kode 11.5 (Lanjutan) *privacy_app.py: pendaftaran dengan persetujuan eksplisit, login berbatas laju dengan peningkatan hash, ekspor, pembetulan, dan penghapusan dengan tombstone.*

```
        uid, self.next_user = self.next_user, self.next_user + 1
        self.users[uid] = {"email": email, "created": self.stamp(),
                           "password": self.hasher.hash(password)}
        self.consents.append({"subject": uid, "event": "given",
                              "version": self.policy_version,
                              "at": self.stamp(), "ip_hash": self.ip_hash(ip)})
        return uid

    def login(self, email, password, ip):
        email = email.strip().lower()
        keys = ["acct:" + email, "ip:" + self.ip_hash(ip)]
        wait = max(self.limiter.retry_after(k) for k in keys)
        if wait:                            # refuse before touching the hash
            raise PrivacyError("too many attempts, wait %d s" % wait)
        uid = self.find(email)
        stored = self.users[uid]["password"] if uid else self.dummy
        if not self.verify(stored, password) or uid is None:
            for k in keys:
                self.limiter.record_failure(k)
            raise PrivacyError(GENERIC)          # same words either way
        self.limiter.forget(keys[0])
        if self.hasher.needs_rehash(stored):
            self.users[uid]["password"] = self.hasher.hash(password)
        token = secrets.token_hex(16)
        self.sessions[token] = uid
        return token

    # ---- ordinary use ------------------------------------------------------
    def post(self, token, text, parent=None):
        pid, self.next_post = self.next_post, self.next_post + 1
        self.posts[pid] = {"author": self.user_of(token), "text": text,
                           "parent": parent, "at": self.stamp()}
        return pid

    # ---- data-subject rights -----------------------------------------------
    def export(self, token):
        """Right of access and portability: everything held, as JSON."""
        uid = self.user_of(token)
        user = self.users[uid]
        data = {
            "exported_at": self.stamp(),
            "account": {"id": uid, "email": user["email"],
                        "created": user["created"]},
            "consents": [{k: c[k] for k in ("event", "version", "at",
                                            "ip_hash") if k in c}
                         for c in self.consents if c["subject"] == uid],
```

Kode 11.5 (Lanjutan) *privacy_app.py: pendaftaran dengan persetujuan eksplisit, login berbatas laju dengan peningkatan hash, ekspor, pembetulan, dan penghapusan dengan tombstone.*

```
            "posts": [{"id": pid, "text": p["text"], "parent": p["parent"],
                       "at": p["at"]}
                      for pid, p in self.posts.items() if p["author"] == uid],
        }
        return json.dumps(data, indent=2, ensure_ascii=False)

    def change_email(self, token, password, new_email):
        uid = self.user_of(token)
        self.check_password(uid, password)
        new_email = new_email.strip().lower()
        if "@" not in new_email or self.find(new_email) is not None:
            raise PrivacyError("that email cannot be used")
        self.users[uid]["email"] = new_email

    def change_password(self, token, old, new):
        uid = self.user_of(token)
        self.check_password(uid, old)
        if len(new) < MIN_PASSWORD:
            raise PrivacyError("password needs at least %d characters"
                               % MIN_PASSWORD)
        self.users[uid]["password"] = self.hasher.hash(new)
        for t in [t for t, u in self.sessions.items() if u == uid]:
            if t != token:
                del self.sessions[t]            # sign out every other device

    def delete_account(self, token, password):
        """Right to erasure. Only the account holder can start it."""
        uid = self.user_of(token)
        self.check_password(uid, password)
        tomb = "deleted-%d" % self.next_tomb
        self.next_tomb += 1
        mine = {pid for pid, p in self.posts.items() if p["author"] == uid}
        needed = {p["parent"] for p in self.posts.values()
                  if p["parent"] in mine and p["author"] != uid}
        for pid in mine:
            if pid in needed:           # others replied: keep an empty shell
                self.posts[pid] = {"author": None, "text": "[deleted]",
                                   "parent": self.posts[pid]["parent"],
                                   "at": self.posts[pid]["at"]}
            else:
                del self.posts[pid]
        for c in self.consents:
            if c["subject"] == uid:
                c["subject"] = tomb     # proof of consent, without the person
                c.pop("ip_hash", None)
        self.consents.append({"subject": tomb, "event": "withdrawn",
```

Kode 11.5 (Lanjutan) *privacy_app.py: pendaftaran dengan persetujuan eksplisit, login berbatas laju dengan peningkatan hash, ekspor, pembetulan, dan penghapusan dengan tombstone.*

```
                                "version": self.policy_version,
                                "at": self.stamp()})
        for t in [t for t, u in self.sessions.items() if u == uid]:
            del self.sessions[t]
        self.limiter.forget("acct:" + self.users[uid]["email"])
        del self.users[uid]
        self.tombstones.append({"tombstone": tomb, "at": self.stamp()})
        return tomb

    def dump(self):
        """Every stored table as one string, for the 'nothing left' test."""
        return json.dumps({"users": self.users, "consents": self.consents,
                           "posts": self.posts, "sessions": self.sessions,
                           "tombstones": self.tombstones})
```

Membacanya metode demi metode:

- `register` menormalkan email menjadi huruf kecil, memeriksa bahwa bentuknya menyerupai alamat email dan bahwa kata sandinya sekurang-kurangnya 15 karakter, lalu bersikeras bahwa `consent is True`. Bukan sekadar bernilai "benar": string `"yes"`, angka 1, dan `None` semuanya ditolak, karena persetujuan harus merupakan tindakan eksplisit, dan program yang menerima nilai bawaan atau nilai sisa persis sama dengan kotak yang tercentang lebih dulu di Subbab 11.6.1. Baru setelah itu metode ini membuat pengguna dan menambahkan peristiwa `given` dengan versi kebijakan, waktu, dan *hash* IP berkunci.
- `login` menyusun dua kunci, bertanya lebih dulu kepada pembatas dan menolak sebelum menyentuh *hash* apa pun, memverifikasi kata sandi (terhadap *hash* tiruan bila emailnya tidak dikenal), mencatat kegagalan pada kedua kunci atau melupakan kunci akun bila berhasil, meningkatkan *hash* yang usang, lalu mengembalikan token sesi acak.
- `export` menghimpun akun, peristiwa persetujuan, dan tulisan milik pengguna yang sedang login, dan tidak ada yang lain, sebagai JSON berindentasi.
- `change_email` dan `change_password` mensyaratkan kata sandi saat ini. Penggantian kata sandi juga mengeluarkan setiap sesi lain milik pengguna itu, karena kata sandi sering diganti justru ketika ada orang lain yang mungkin mengetahui kata sandi lama.
- `delete_account` mensyaratkan kata sandi, menghitung tulisan pengguna mana yang dibalas orang lain, menyimpan tulisan-tulisan itu sebagai cangkang anonim dan menghapus sisanya, memberi label ulang peristiwa persetujuan dengan pengenal *tombstone* dan membuang *hash* IP-nya, menambahkan peristiwa `withdrawn`, menghapus setiap sesi dan kunci akun di pembatas, menghapus baris pengguna, lalu mencatat *tombstone*-nya.
- `dump` menyerialkan setiap tabel menjadi satu string. Metode ini ada untuk uji yang menelusuri seluruh penyimpanan demi mencari jejak pengguna yang telah dihapus.

Aplikasi Java di **Kode 11.6** memiliki susunan yang sama, dengan kelas-kelas kecil untuk pengguna, peristiwa persetujuan, dan tulisan, serta `PrivacyException` untuk penolakan. Aplikasi itu memerlukan satu pembantu yang di Python sudah tersedia: **Kode 11.7** menulis *map* dan *list* sebagai JSON dengan tata letak dua spasi yang sama seperti `json.dumps(indent=2)` di Python, sehingga kedua ekspor identik.

Kode 11.6 `PrivacyApp.java`: aplikasi yang sama dalam Java.

```
import java.nio.charset.StandardCharsets;
import java.security.SecureRandom;
import java.time.Instant;
import java.time.OffsetDateTime;
import java.time.ZoneOffset;
import java.time.format.DateTimeFormatter;
import java.util.*;
import java.util.function.LongSupplier;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

/** G7: a privacy-by-design mini app (in memory, JDK only). Explicit consent
 *  with a version and a time; slow salted password hashes; rate-limited
 *  logins; self-service export, correction and erasure. Erasure purges every
 *  personal field and leaves an anonymous tombstone only where other
 *  people's content still points to it. */
public class PrivacyApp {
    static final int MIN_PASSWORD = 15;   // NIST SP 800-63B-4, single factor
    static final String GENERIC = "wrong email or password";
    static final ZoneOffset WIB = ZoneOffset.ofHours(7);
    private static final SecureRandom RANDOM = new SecureRandom();

    static class PrivacyException extends RuntimeException {
        private static final long serialVersionUID = 1L;

        PrivacyException(String message) {
            super(message);
        }
    }

    static class User {
        String email, password, created;
    }

    static class Consent {
        String subject, event, version, at, ipHash;   // ipHash may be null
    }

    static class Post {
        Integer author, parent;                       // author null: deleted
        String text, at;
    }
```

Kode 11.6 (Lanjutan) *PrivacyApp.java: aplikasi yang sama dalam Java.*

```
    final LongSupplier clock;
    final byte[] secret;                // keys the IP hashes; not in the tables
    Pbkdf2Hasher hasher;
    final List<Pbkdf2Hasher> hashers = new ArrayList<>();
    final SlidingWindowLimiter limiter;
    String policyVersion = "1.0";               // the current policy
    final Map<Integer, User> users = new LinkedHashMap<>();
    final List<Consent> consents = new ArrayList<>();
    final Map<Integer, Post> posts = new LinkedHashMap<>();
    final Map<String, Integer> sessions = new LinkedHashMap<>();
    final List<String[]> tombstones = new ArrayList<>();    // {tombstone, at}
    int nextUser = 1, nextPost = 1, nextTomb = 1;
    final String dummy;

    PrivacyApp(LongSupplier clock, byte[] secret, Pbkdf2Hasher hasher,
               int maxFailures, long windowSeconds) {
        this.clock = clock;
        this.secret = secret;
        this.hasher = hasher;
        this.hashers.add(hasher);
        this.limiter = new SlidingWindowLimiter(maxFailures, windowSeconds, clock);
        this.dummy = hasher.hash(token());
    }

    // ---- helpers -----------------------------------------------------------
    String stamp() {
        return OffsetDateTime.ofInstant(Instant.ofEpochSecond(clock.getAsLong()),
                WIB).format(DateTimeFormatter.ISO_OFFSET_DATE_TIME);
    }

    String ipHash(String ip) {
        try {
            Mac mac = Mac.getInstance("HmacSHA256");
            mac.init(new SecretKeySpec(secret, "HmacSHA256"));
            byte[] d = mac.doFinal(ip.getBytes(StandardCharsets.UTF_8));
            return HexFormat.of().formatHex(d).substring(0, 16);
        } catch (Exception e) {
            throw new IllegalStateException(e);
        }
    }
```

Kode 11.6 (Lanjutan) *PrivacyApp.java: aplikasi yang sama dalam Java.*

```
    static String token() {
        byte[] b = new byte[16];
        RANDOM.nextBytes(b);
        return HexFormat.of().formatHex(b);
    }

    Integer find(String email) {
        for (var e : users.entrySet()) {
            if (e.getValue().email.equals(email)) return e.getKey();
        }
        return null;
    }

    boolean verify(String stored, String password) {
        for (Pbkdf2Hasher h : hashers) {
            if (h.recognizes(stored)) return h.verify(stored, password);
        }
        return false;
    }

    int userOf(String token) {
        Integer uid = sessions.get(token);
        if (uid == null) throw new PrivacyException("not logged in");
        return uid;
    }

    void checkPassword(int uid, String password) {
        if (!verify(users.get(uid).password, password)) {
            throw new PrivacyException("password confirmation failed");
        }
    }

    /** Adopt a new recipe; old hashes are upgraded at the next login. */
    void useHasher(Pbkdf2Hasher h) {
        hasher = h;
        hashers.add(0, h);
    }
```

Kode 11.6 (Lanjutan) *PrivacyApp.java: aplikasi yang sama dalam Java.*

```
    // ---- registration and login -------------------------------------------
    int register(String email, String password, Boolean consent, String ip) {
        email = email.strip().toLowerCase();
        if (!email.contains("@")) {
            throw new PrivacyException("not an email address");
        }
        if (password.length() < MIN_PASSWORD) {
            throw new PrivacyException("password needs at least " + MIN_PASSWORD
                    + " characters");
        }
        if (!Boolean.TRUE.equals(consent)) {       // never assumed or default
            throw new PrivacyException("consent is required");
        }
        if (find(email) != null) {
            throw new PrivacyException("email already registered");
        }
        int uid = nextUser++;
        User u = new User();
        u.email = email;
        u.created = stamp();
        u.password = hasher.hash(password);
        users.put(uid, u);
        Consent c = new Consent();
        c.subject = String.valueOf(uid);
        c.event = "given";
        c.version = policyVersion;
        c.at = stamp();
        c.ipHash = ipHash(ip);
        consents.add(c);
        return uid;
    }

    String login(String email, String password, String ip) {
        email = email.strip().toLowerCase();
        List<String> keys = List.of("acct:" + email, "ip:" + ipHash(ip));
        long wait = Math.max(limiter.retryAfter(keys.get(0)),
                             limiter.retryAfter(keys.get(1)));
        if (wait > 0) {                            // refuse before hashing
            throw new PrivacyException("too many attempts, wait " + wait + " s");
        }
        Integer uid = find(email);
        String stored = uid != null ? users.get(uid).password : dummy;
        if (!verify(stored, password) || uid == null) {
            for (String k : keys) limiter.recordFailure(k);
            throw new PrivacyException(GENERIC);   // same words either way
        }
```

Kode 11.6 (Lanjutan) *PrivacyApp.java: aplikasi yang sama dalam Java.*

```
        limiter.forget(keys.get(0));
        if (hasher.needsRehash(stored)) {
            users.get(uid).password = hasher.hash(password);
        }
        String t = token();
        sessions.put(t, uid);
        return t;
    }

    // ---- ordinary use ----------------------------------------------------
    int post(String token, String text, Integer parent) {
        Post p = new Post();
        p.author = userOf(token);
        p.text = text;
        p.parent = parent;
        p.at = stamp();
        int pid = nextPost++;
        posts.put(pid, p);
        return pid;
    }

    // ---- data-subject rights ---------------------------------------------
    /** Right of access and portability: everything held, as JSON. */
    String export(String token) {
        int uid = userOf(token);
        User u = users.get(uid);
        Map<String, Object> data = new LinkedHashMap<>();
        data.put("exported_at", stamp());
        Map<String, Object> account = new LinkedHashMap<>();
        account.put("id", uid);
        account.put("email", u.email);
        account.put("created", u.created);
        data.put("account", account);
        List<Object> cs = new ArrayList<>();
        for (Consent c : consents) {
            if (!c.subject.equals(String.valueOf(uid))) continue;
            Map<String, Object> m = new LinkedHashMap<>();
            m.put("event", c.event);
            m.put("version", c.version);
            m.put("at", c.at);
            if (c.ipHash != null) m.put("ip_hash", c.ipHash);
            cs.add(m);
        }
        data.put("consents", cs);
        List<Object> ps = new ArrayList<>();
        for (var e : posts.entrySet()) {
```

Kode 11.6 (Lanjutan) *PrivacyApp.java: aplikasi yang sama dalam Java.*

```
            Post p = e.getValue();
            if (!Integer.valueOf(uid).equals(p.author)) continue;
            Map<String, Object> m = new LinkedHashMap<>();
            m.put("id", e.getKey());
            m.put("text", p.text);
            m.put("parent", p.parent);
            m.put("at", p.at);
            ps.add(m);
        }
        data.put("posts", ps);
        return Json.write(data, "");
    }

    void changeEmail(String token, String password, String newEmail) {
        int uid = userOf(token);
        checkPassword(uid, password);
        newEmail = newEmail.strip().toLowerCase();
        if (!newEmail.contains("@") || find(newEmail) != null) {
            throw new PrivacyException("that email cannot be used");
        }
        users.get(uid).email = newEmail;
    }

    void changePassword(String token, String old, String fresh) {
        int uid = userOf(token);
        checkPassword(uid, old);
        if (fresh.length() < MIN_PASSWORD) {
            throw new PrivacyException("password needs at least " + MIN_PASSWORD
                    + " characters");
        }
        users.get(uid).password = hasher.hash(fresh);
        // sign out every other device
        sessions.entrySet().removeIf(e -> e.getValue() == uid
                                        && !e.getKey().equals(token));
    }
```

Kode 11.6 (Lanjutan) *PrivacyApp.java: aplikasi yang sama dalam Java.*

```
    /** Right to erasure. Only the account holder can start it. */
    String deleteAccount(String token, String password) {
        int uid = userOf(token);
        checkPassword(uid, password);
        String tomb = "deleted-" + nextTomb++;
        Set<Integer> mine = new HashSet<>();
        posts.forEach((pid, p) -> {
            if (Integer.valueOf(uid).equals(p.author)) mine.add(pid);
        });
        Set<Integer> needed = new HashSet<>();
        for (Post p : posts.values()) {
            if (p.parent != null && mine.contains(p.parent)
                    && !Integer.valueOf(uid).equals(p.author)) needed.add(p.parent);
        }
        for (int pid : mine) {
            if (needed.contains(pid)) {           // others replied: empty shell
                Post p = posts.get(pid);
                p.author = null;
                p.text = "[deleted]";
            } else {
                posts.remove(pid);
            }
        }
        for (Consent c : consents) {
            if (c.subject.equals(String.valueOf(uid))) {
                c.subject = tomb;                 // proof, without the person
                c.ipHash = null;
            }
        }
        Consent w = new Consent();
        w.subject = tomb;
        w.event = "withdrawn";
        w.version = policyVersion;
        w.at = stamp();
        consents.add(w);
        sessions.values().removeIf(v -> v == uid);
        limiter.forget("acct:" + users.get(uid).email);
        users.remove(uid);
        tombstones.add(new String[] {tomb, stamp()});
        return tomb;
    }
```

Kode 11.6 (Lanjutan) *PrivacyApp.java: aplikasi yang sama dalam Java.*

```
    /** Every stored table as one string, for the "nothing left" test. */
    String dump() {
        StringBuilder b = new StringBuilder();
        users.forEach((id, u) -> b.append(id).append(' ').append(u.email)
                .append(' ').append(u.password).append(' ').append(u.created)
                .append('\n'));
        for (Consent c : consents) {
            b.append(c.subject).append(' ').append(c.event).append(' ')
             .append(c.version).append(' ').append(c.at).append(' ')
             .append(c.ipHash).append('\n');
        }
        posts.forEach((id, p) -> b.append(id).append(' ').append(p.author)
                .append(' ').append(p.parent).append(' ').append(p.text)
                .append('\n'));
        sessions.forEach((t, id) ->
                b.append(t).append(' ').append(id).append('\n'));
        for (String[] t : tombstones) b.append(String.join(" ", t)).append('\n');
        return b.toString();
    }
}
```

Kode 11.7 `Json.java`: penghasil JSON kecil dengan tata letak yang sama seperti `json.dumps(indent=2)` di Python.

```
import java.util.List;
import java.util.Map;

/** Writes maps, lists, strings, numbers, booleans and null as JSON with a
 *  two-space indent, the same layout as Python's json.dumps(indent=2). */
public class Json {
    static String write(Object v, String indent) {
        String in = indent + "  ";
        if (v instanceof Map<?, ?> map) {
            if (map.isEmpty()) return "{}";
            StringBuilder b = new StringBuilder("{\n");
            int i = 0;
            for (var e : map.entrySet()) {
                b.append(in).append(quote(String.valueOf(e.getKey()))).append(": ")
                 .append(write(e.getValue(), in));
                b.append(++i < map.size() ? ",\n" : "\n");
            }
            return b.append(indent).append('}').toString();
        }
        if (v instanceof List<?> list) {
            if (list.isEmpty()) return "[]";
            StringBuilder b = new StringBuilder("[\n");
            for (int i = 0; i < list.size(); i++) {
                b.append(in).append(write(list.get(i), in));
                b.append(i + 1 < list.size() ? ",\n" : "\n");
            }
            return b.append(indent).append(']').toString();
        }
        if (v instanceof String s) return quote(s);
        return String.valueOf(v);          // numbers, true, false, null
    }

    static String quote(String s) {
        StringBuilder b = new StringBuilder("\"");
        for (char c : s.toCharArray()) {
            switch (c) {
                case '"' -> b.append("\\\"");
                case '\\' -> b.append("\\\\");
                case '\n' -> b.append("\\n");
                case '\r' -> b.append("\\r");
                case '\t' -> b.append("\\t");
                default -> {
                    if (c < 0x20) b.append(String.format("\\u%04x", (int) c));
                    else b.append(c);
                }
            }
```

Kode 11.7 (Lanjutan) *Json.java: penghasil JSON kecil dengan tata letak yang sama seperti json.dumps(indent=2) di Python.*

```
        }
        return b.append('"').toString();
    }
}
```

11.10.5 Demonya

Kode 11.8 menjalankan skenario **Gambar 11.3** dan **Gambar 11.5**. Demo ini dimulai dengan resep *lama* berupa 100.000 iterasi PBKDF2, mendaftarkan dua pengguna, lalu mengalihkan kebijakan ke 600.000 iterasi (atau ke Argon2id dengan `--argon2`), sehingga peningkatan saat login menjadi terlihat. Fungsi pembantu `recipe` hanya mencetak algoritma dan pengaturan biayanya, tidak pernah *salt* maupun *hash*-nya.

Kode 11.8 `demo.py`: persetujuan, *hashing*, pembatasan laju, ekspor, pembetulan, dan penghapusan dalam satu skenario.

```
"""Walk through the G7 lab: consent, hashing, rate limit, export, erasure.

usage: python demo.py            (PBKDF2 from the standard library)
       python demo.py --argon2   (Argon2id; needs: pip install argon2-cffi)
"""
import sys

from passwords import Argon2Hasher, Pbkdf2Hasher
from privacy_app import PrivacyApp, PrivacyError


class Clock:
    def __init__(self, start):
        self.now = start

    def __call__(self):
        return self.now


def attempt(label, action, shown=None):
    try:
        result = action()
        print("%-38s -> %s" % (label, shown or result))
        return result
    except PrivacyError as e:
        print("%-38s -> refused: %s" % (label, e))
        return None


def recipe(stored):
    """The algorithm and its cost settings; never the salt or the hash."""
    parts = stored.split("$")
    return "$".join(parts[:2] if parts[0] else parts[:4])


def main():
    clock = Clock(1790215200)                # 2026-09-24 09:00:00 WIB
    legacy = Pbkdf2Hasher(iterations=100_000)
    app = PrivacyApp(clock, secret=b"lab-secret-kept-outside-the-db",
                     hasher=legacy, max_failures=3, window_seconds=60)
    ayu_pw, budi_pw = "kopi-susu-tanpa-gula-2026", "nasi-goreng-pedas-sekali"
    ayu_ip, budi_ip = "203.0.113.7", "198.51.100.23"
```

Kode 11.8 (Lanjutan) *demo.py: persetujuan, hashing, pembatasan laju, ekspor, pembetulan, dan penghapusan dalam satu skenario.*

```
print("== 1. Registration needs explicit consent")
attempt("register ayu, consent box left empty",
        lambda: app.register("ayu@example.com", ayu_pw, False, ayu_ip))
attempt("register ayu, consent box ticked",
        lambda: "user %d" % app.register("Ayu@Example.com", ayu_pw,
                                         True, ayu_ip))
attempt("register budi, consent box ticked",
        lambda: "user %d" % app.register("budi@example.com", budi_pw,
                                         True, budi_ip))

print("\n== 2. What is stored instead of the password")
print("ayu's recipe: " + recipe(app.users[1]["password"]))
new = Argon2Hasher() if "--argon2" in sys.argv else Pbkdf2Hasher()
app.use_hasher(new)
print("policy moves to: " + recipe(new.hash("x" * 16)))

print("\n== 3. Login rate limiting (lab values: 3 failures in 60 s)")
for guess in ("kopi-susu-2025", "kopi-susu-2024", "kopi-susu-2023"):
    clock.now += 5
    attempt("09:00:%02d ayu, wrong password" % (clock.now % 60),
            lambda: app.login("ayu@example.com", guess, ayu_ip))
clock.now += 5
attempt("09:00:20 ayu, right password",
        lambda: app.login("ayu@example.com", ayu_pw, ayu_ip), "signed in")
clock.now += 60
token = attempt("09:01:20 ayu, right password",
                lambda: app.login("ayu@example.com", ayu_pw, ayu_ip),
                "signed in")
print("ayu's recipe after login: " + recipe(app.users[1]["password"]))
budi = app.login("budi@example.com", budi_pw, budi_ip)

print("\n== 4. Ordinary use")
first = app.post(token, "Is express laundry open on Sunday?")
app.post(budi, "Yes, until noon.", parent=first)
print("ayu posted #%d; budi replied" % first)

print("\n== 5. Right of access and portability: export")
print(app.export(token))

print("\n== 6. Right to correction: change email")
attempt("change email, wrong password",
        lambda: app.change_email(token, "guess-guess-guess", "a@x.id"))
attempt("change email, password confirmed",
        lambda: app.change_email(token, ayu_pw, "ayu.new@example.com"),
        "email is now ayu.new@example.com")
```

Kode 11.8 (Lanjutan) *demo.py: persetujuan, hashing, pembatasan laju, ekspor, pembetulan, dan penghapusan dalam satu skenario.*

```
    print("\n== 7. Right to erasure")
    clock.now += 3600
    attempt("delete ayu's account",
            lambda: app.delete_account(token, ayu_pw))
    reply = app.posts[2]
    print("budi's reply still answers post #%d: %r" % (
        reply["parent"], app.posts[reply["parent"]]["text"]))

    print("\n== 8. What is left of ayu")
    dump = app.dump()
    for trace in ("ayu@example.com", "ayu.new@example.com", ayu_ip,
                  app.ip_hash(ayu_ip), "Is express laundry open"):
        print("%-30s %s" % (trace, "FOUND" if trace in dump else "not found"))
    for c in app.consents:
        print("consent log: %-10s %-9s v%s %s" % (c["subject"], c["event"],
                                                  c["version"], c["at"]))


if __name__ == "__main__":
    main()
```

Kode 11.9 adalah demo versi Java.

Kode 11.9 `Demo.java`: skenario yang sama dalam Java.

```
import java.util.function.Supplier;

/** Walks through the G7 lab: consent, hashing, rate limit, export, erasure.
 *  usage: java Demo      (PBKDF2 from the JDK) */
public class Demo {
    static long now = 1790215200L;                  // 2026-09-24 09:00:00 WIB

    static <T> T attempt(String label, Supplier<T> action, String shown) {
        try {
            T result = action.get();
            System.out.printf("%-38s -> %s%n", label,
                              shown != null ? shown : result);
            return result;
        } catch (PrivacyApp.PrivacyException e) {
            System.out.printf("%-38s -> refused: %s%n", label, e.getMessage());
            return null;
        }
    }

    /** The algorithm and its cost settings; never the salt or the hash. */
    static String recipe(String stored) {
        String[] p = stored.split("\\$");
        return p[0] + "$" + p[1];
    }

    public static void main(String[] args) {
        Pbkdf2Hasher legacy = new Pbkdf2Hasher(100_000);
        PrivacyApp app = new PrivacyApp(() -> now,
                "lab-secret-kept-outside-the-db".getBytes(), legacy, 3, 60);
        String ayuPw = "kopi-susu-tanpa-gula-2026";
        String budiPw = "nasi-goreng-pedas-sekali";
        String ayuIp = "203.0.113.7", budiIp = "198.51.100.23";

        System.out.println("== 1. Registration needs explicit consent");
        attempt("register ayu, consent box left empty",
                () -> app.register("ayu@example.com", ayuPw, false, ayuIp), null);
        attempt("register ayu, consent box ticked",
                () -> "user " + app.register("Ayu@Example.com", ayuPw, true, ayuIp),
                null);
        attempt("register budi, consent box ticked",
                () -> "user " + app.register("budi@example.com", budiPw, true,
                                             budiIp),
                null);
```

Kode 11.9 (Lanjutan) *Demo.java: skenario yang sama dalam Java.*

```
        System.out.println("\n== 2. What is stored instead of the password");
        System.out.println("ayu's recipe: " + recipe(app.users.get(1).password));
        Pbkdf2Hasher fresh = new Pbkdf2Hasher();
        app.useHasher(fresh);
        System.out.println("policy moves to: "
                + recipe(fresh.hash("x".repeat(16))));

        System.out.println(
                "\n== 3. Login rate limiting (lab values: 3 failures in 60 s)");
        for (String guess : new String[] {"kopi-susu-2025", "kopi-susu-2024",
                                          "kopi-susu-2023"}) {
            now += 5;
            attempt(String.format("09:00:%02d ayu, wrong password", now % 60),
                    () -> app.login("ayu@example.com", guess, ayuIp), null);
        }
        now += 5;
        attempt("09:00:20 ayu, right password",
                () -> app.login("ayu@example.com", ayuPw, ayuIp), "signed in");
        now += 60;
        String token = attempt("09:01:20 ayu, right password",
                () -> app.login("ayu@example.com", ayuPw, ayuIp), "signed in");
        System.out.println("ayu's recipe after login: "
                + recipe(app.users.get(1).password));
        String budi = app.login("budi@example.com", budiPw, budiIp);

        System.out.println("\n== 4. Ordinary use");
        int first = app.post(token, "Is express laundry open on Sunday?", null);
        app.post(budi, "Yes, until noon.", first);
        System.out.println("ayu posted #" + first + "; budi replied");

        System.out.println("\n== 5. Right of access and portability: export");
        System.out.println(app.export(token));

        System.out.println("\n== 6. Right to correction: change email");
        attempt("change email, wrong password", () -> {
            app.changeEmail(token, "guess-guess-guess", "a@x.id");
            return "";
        }, null);
        attempt("change email, password confirmed", () -> {
            app.changeEmail(token, ayuPw, "ayu.new@example.com");
            return "";
        }, "email is now ayu.new@example.com");
```

Kode 11.9 (Lanjutan) *Demo.java: skenario yang sama dalam Java.*

```
        System.out.println("\n== 7. Right to erasure");
        now += 3600;
        attempt("delete ayu's account", () -> app.deleteAccount(token, ayuPw),
                null);
        PrivacyApp.Post reply = app.posts.get(2);
        System.out.println("budi's reply still answers post #" + reply.parent
                + ": '" + app.posts.get(reply.parent).text + "'");

        System.out.println("\n== 8. What is left of ayu");
        String dump = app.dump();
        for (String trace : new String[] {"ayu@example.com", "ayu.new@example.com",
                ayuIp, app.ipHash(ayuIp), "Is express laundry open"}) {
            System.out.printf("%-30s %s%n", trace,
                    dump.contains(trace) ? "FOUND" : "not found");
        }
        for (PrivacyApp.Consent c : app.consents) {
            System.out.printf("consent log: %-10s %-9s v%s %s%n", c.subject,
                    c.event, c.version, c.at);
        }
    }
}
```

Jalankan `python demo.py` di folder `python`, atau `javac -d out *.java` lalu `java -cp out Demo` di folder `java`. Keduanya mencetak keluaran berikut, identik:

```
== 1. Registration needs explicit consent
register ayu, consent box left empty   -> refused: consent is required
register ayu, consent box ticked       -> user 1
register budi, consent box ticked      -> user 2

== 2. What is stored instead of the password
ayu's recipe: pbkdf2_sha256$100000
policy moves to: pbkdf2_sha256$600000

== 3. Login rate limiting (lab values: 3 failures in 60 s)
09:00:05 ayu, wrong password           -> refused: wrong email or password
09:00:10 ayu, wrong password           -> refused: wrong email or password
09:00:15 ayu, wrong password           -> refused: wrong email or password
09:00:20 ayu, right password           -> refused: too many attempts, wait 45 s
09:01:20 ayu, right password           -> signed in
ayu's recipe after login: pbkdf2_sha256$600000

== 4. Ordinary use
ayu posted #1; budi replied

== 5. Right of access and portability: export
{
  "exported_at": "2026-09-24T09:01:20+07:00",
  "account": {
    "id": 1,
    "email": "ayu@example.com",
    "created": "2026-09-24T09:00:00+07:00"
  },
  "consents": [
    {
      "event": "given",
      "version": "1.0",
      "at": "2026-09-24T09:00:00+07:00",
      "ip_hash": "2f7a7c6e655482f7"
    }
  ],
  "posts": [
    {
      "id": 1,
      "text": "Is express laundry open on Sunday?",
      "parent": null,
      "at": "2026-09-24T09:01:20+07:00"
    }
  ]
}

== 6. Right to correction: change email
change email, wrong password           -> refused: password confirmation failed
change email, password confirmed       -> email is now ayu.new@example.com

== 7. Right to erasure
delete ayu's account                   -> deleted-1
budi's reply still answers post #1: '[deleted]'

== 8. What is left of ayu
ayu@example.com               not found
ayu.new@example.com           not found
203.0.113.7                   not found
2f7a7c6e655482f7              not found
Is express laundry open       not found
consent log: deleted-1  given     v1.0 2026-09-24T09:00:00+07:00
consent log: 2          given     v1.0 2026-09-24T09:00:00+07:00
consent log: deleted-1  withdrawn v1.0 2026-09-24T10:01:20+07:00
```

Ikuti bagian demi bagian.

1. Pendaftaran pertama ditolak karena kotak persetujuan dibiarkan kosong; dua pendaftaran berikutnya berhasil. Ayu mengetik `Ayu@Example.com`, dan aplikasi menyimpan `ayu@example.com`, sehingga ia tidak akan tanpa sengaja memiliki dua akun.
2. String tersimpan milik Ayu berbunyi `pbkdf2_sha256$100000`: resep lama. Kebijakan kemudian beralih ke 600.000 iterasi.
3. Tiga kata sandi salah pada 09:00:05, :10, dan :15 ditolak dengan pesan yang sama. Pada 09:00:20 Ayu mengetik kata sandi yang *benar* dan tetap ditolak, tanpa kata sandinya diperiksa sama sekali, dan diminta menunggu 45 detik, persis Persamaan (11.3). Pada 09:01:20 ia berhasil masuk, dan string tersimpannya sudah ditingkatkan menjadi `pbkdf2_sha256$600000`.
4. Ayu bertanya dan Budi membalas.
5. Ekspor Ayu memuat akunnya, peristiwa persetujuannya beserta versi, waktu, dan *hash* IP-nya, serta tulisannya; ekspor itu tidak memuat balasan Budi maupun *hash* kata sandi apa pun.
6. Penggantian email dengan kata sandi yang salah ditolak; dengan kata sandi yang benar penggantian itu berhasil.
7. Satu jam kemudian Ayu menghapus akunnya dan menerima *tombstone* `deleted-1`. Balasan Budi masih menunjuk ke tulisan 1, yang kini berbunyi `[deleted]`.
8. Penelusuran setiap tabel untuk alamat email lama dan baru Ayu, alamat IP-nya, *hash* berkuncinya, dan teks pertanyaannya tidak menemukan apa pun. Log persetujuan masih membuktikan bahwa seseorang menyetujui versi 1.0 pada pukul 09:00 dan menariknya pada pukul 10:01, tetapi tidak lagi menyebut siapa orangnya.

Dengan `argon2-cffi` terpasang, `python demo.py --argon2` mencetak baris-baris yang sama kecuali dua, yang menampilkan resep Argon2id dengan konfigurasi OWASP:

```
policy moves to: $argon2id$v=19$m=19456,t=2,p=1
ayu's recipe after login: $argon2id$v=19$m=19456,t=2,p=1
```

Hash PBKDF2 milik Ayu, yang diverifikasi dengan resepnya sendiri, telah diganti *hash* Argon2id pada login berhasilnya yang pertama; itulah jalur peningkatan yang dipakai Profitina untuk *hash* lama.

11.10.6 Uji-ujinya

Langkah privasi yang tidak diuji adalah janji, bukan langkah. **Kode 11.10** memuat sepuluh uji. Dua di antaranya memeriksa *hashing* itu sendiri terhadap acuan dari luar: jawaban yang diketahui untuk PBKDF2-HMAC-SHA256 yang diterbitkan dalam RFC 7914 [19], dan string tersimpan untuk *salt* tetap yang harus direproduksi versi Java karakter demi karakter. Sisanya memeriksa perilaku yang dibahas bab ini: persetujuan harus tepat `True`; email yang tidak dikenal dan kata sandi yang salah menghasilkan pesan yang sama; jendela memblokir lalu melepaskan; *hash* lama ditingkatkan; ekspor hanya memuat data milik pengguna sendiri; perubahan memerlukan kata sandi dan mengeluarkan sesi lain; dan penghapusan tidak meninggalkan apa pun yang bersifat pribadi.

Kode 11.10 `test_privacy_app.py`: sepuluh uji, termasuk uji jawaban-diketahui untuk PBKDF2 dan penelusuran setiap tabel setelah penghapusan.

```
"""Tests for the G7 lab. Run: python -m unittest -v test_privacy_app"""
import json
import unittest

from passwords import Pbkdf2Hasher
from privacy_app import GENERIC, PrivacyApp, PrivacyError

PW = "kopi-susu-tanpa-gula-2026"
IP = "203.0.113.7"


class Clock:
    def __init__(self):
        self.now = 1790215200

    def __call__(self):
        return self.now


def new_app(clock=None):
    # Few iterations keep the tests fast; the demo uses the OWASP value.
    return PrivacyApp(clock or Clock(), b"test-secret",
                      hasher=Pbkdf2Hasher(iterations=1000))


class PrivacyTests(unittest.TestCase):
    def test_pbkdf2_known_answer(self):
        # RFC 7914, section 11: PBKDF2-HMAC-SHA256("passwd", "salt", 1, 64)
        import hashlib
        key = hashlib.pbkdf2_hmac("sha256", b"passwd", b"salt", 1, 64)
        self.assertTrue(key.hex().startswith("55ac046e56e3089fec1691c22544b605"))

    def test_recipe_is_the_same_in_java(self):
        # The Java lab computes exactly this string for the same inputs.
        expected = ("pbkdf2_sha256$1000$cHJvZml0aW5hLWxhYi0xNg==$"
                    "RJZs9e03R01zF3dwuFJoifK15hAmgK6TPoBFUVtqkrE=")
        h = Pbkdf2Hasher(iterations=1000)
        self.assertEqual(h.hash(PW, salt=b"profitina-lab-16"), expected)
        self.assertTrue(h.verify(expected, PW))
```

Kode 11.10 (Lanjutan) *test_privacy_app.py: sepuluh uji, termasuk uji jawaban-diketahui untuk PBKDF2 dan penelusuran setiap tabel setelah penghapusan.*

```
    def test_hash_is_salted_and_verifies(self):
        h = Pbkdf2Hasher(iterations=1000)
        a, b = h.hash(PW), h.hash(PW)
        self.assertNotEqual(a, b)                # a fresh salt every time
        self.assertTrue(h.verify(a, PW))
        self.assertFalse(h.verify(a, PW + "x"))
        self.assertNotIn(PW, a)

    def test_consent_must_be_explicitly_true(self):
        app = new_app()
        for value in (False, None, "yes", 1):
            with self.assertRaises(PrivacyError):
                app.register("ayu@example.com", PW, value, IP)
        uid = app.register("ayu@example.com", PW, True, IP)
        log = [c for c in app.consents if c["subject"] == uid]
        self.assertEqual(log[0]["event"], "given")
        self.assertEqual(log[0]["version"], "1.0")
        self.assertNotIn(IP, app.dump())         # only a keyed hash is kept

    def test_same_message_for_unknown_email_and_wrong_password(self):
        app = new_app()
        app.register("ayu@example.com", PW, True, IP)
        messages = []
        for email, pw in (("ayu@example.com", "wrong-password-123"),
                          ("nobody@example.com", PW)):
            with self.assertRaises(PrivacyError) as ctx:
                app.login(email, pw, "198.51.100." + str(len(messages)))
            messages.append(str(ctx.exception))
        self.assertEqual(messages, [GENERIC, GENERIC])

    def test_sliding_window_blocks_then_releases(self):
        clock = Clock()
        app = new_app(clock)
        app.register("ayu@example.com", PW, True, IP)
        for _ in range(3):
            clock.now += 1
            self.assertRaises(PrivacyError, app.login, "ayu@example.com",
                              "wrong-password-123", IP)
        with self.assertRaises(PrivacyError) as ctx:
            app.login("ayu@example.com", PW, IP)   # right password, still no
        self.assertIn("too many attempts", str(ctx.exception))
        clock.now += 60
        self.assertTrue(app.login("ayu@example.com", PW, IP))
```

Kode 11.10 (Lanjutan) *test_privacy_app.py: sepuluh uji, termasuk uji jawaban-diketahui untuk PBKDF2 dan penelusuran setiap tabel setelah penghapusan.*

```
    def test_old_hash_is_upgraded_at_login(self):
        app = new_app()
        uid = app.register("ayu@example.com", PW, True, IP)
        app.use_hasher(Pbkdf2Hasher(iterations=2000))
        self.assertIn("$1000$", app.users[uid]["password"])
        app.login("ayu@example.com", PW, IP)
        self.assertIn("$2000$", app.users[uid]["password"])

    def test_export_contains_only_own_data(self):
        app = new_app()
        app.register("ayu@example.com", PW, True, IP)
        app.register("budi@example.com", PW, True, "198.51.100.23")
        ayu = app.login("ayu@example.com", PW, IP)
        budi = app.login("budi@example.com", PW, "198.51.100.23")
        app.post(ayu, "hello from ayu")
        app.post(budi, "hello from budi")
        data = json.loads(app.export(ayu))
        self.assertEqual(data["account"]["email"], "ayu@example.com")
        self.assertEqual([p["text"] for p in data["posts"]], ["hello from ayu"])
        self.assertNotIn("budi", json.dumps(data))
        self.assertNotIn("password", json.dumps(data))

    def test_changes_need_the_password(self):
        app = new_app()
        app.register("ayu@example.com", PW, True, IP)
        t1 = app.login("ayu@example.com", PW, IP)
        t2 = app.login("ayu@example.com", PW, IP)
        self.assertRaises(PrivacyError, app.change_email, t1,
                          "not-the-password", "new@example.com")
        app.change_password(t1, PW, "a-brand-new-long-passphrase")
        self.assertIn(t1, app.sessions)
        self.assertNotIn(t2, app.sessions)       # other devices signed out

    def test_delete_leaves_nothing_personal(self):
        app = new_app()
        app.register("ayu@example.com", PW, True, IP)
        app.register("budi@example.com", PW, True, "198.51.100.23")
        ayu = app.login("ayu@example.com", PW, IP)
        budi = app.login("budi@example.com", PW, "198.51.100.23")
        first = app.post(ayu, "a question from ayu")
        app.post(ayu, "a note nobody answered")
        reply = app.post(budi, "an answer from budi", parent=first)
        app.change_email(ayu, PW, "ayu.new@example.com")
        stored_hash = app.users[1]["password"]
        ip_hash = app.ip_hash(IP)
        self.assertRaises(PrivacyError, app.delete_account, ayu, "wrong-one")
```

Kode 11.10 (Lanjutan) *test_privacy_app.py: sepuluh uji, termasuk uji jawaban-diketahui untuk PBKDF2 dan penelusuran setiap tabel setelah penghapusan.*

```
        tomb = app.delete_account(ayu, PW)
        dump = app.dump()
        for trace in ("ayu@example.com", "ayu.new@example.com", IP, ip_hash,
                      stored_hash, "a question from ayu", "nobody answered"):
            self.assertNotIn(trace, dump)
        self.assertEqual(app.posts[first]["text"], "[deleted]")
        self.assertEqual(app.posts[reply]["parent"], first)  # thread intact
        events = [c["event"] for c in app.consents if c["subject"] == tomb]
        self.assertEqual(events, ["given", "withdrawn"])
        self.assertFalse(any("ayu" in k for k in app.limiter.failures))
        self.assertRaises(PrivacyError, app.export, ayu)     # session gone
        self.assertIn("budi@example.com", dump)              # others untouched


if __name__ == "__main__":
    unittest.main()
```

Uji terakhir adalah uji yang menjadi alasan keberadaan seluruh lab. Uji itu membangun riwayat kecil yang realistis (dua pengguna, sebuah pertanyaan, sebuah catatan tanpa balasan, sebuah balasan, sebuah penggantian email), menghapus Ayu, menyerialkan setiap tabel, lalu memastikan bahwa tidak satu pun dari tujuh jejak ditemukan: kedua alamat email, alamat IP, *hash* berkuncinya, *hash* kata sandi yang tersimpan, dan teks kedua tulisannya. Uji itu juga memeriksa apa yang harus *tetap ada*: utas balasan utuh, *tombstone* memiliki tepat satu peristiwa `given` dan satu `withdrawn`, data Budi tidak tersentuh, dan sesi Ayu tidak lagi berfungsi.

Uji-uji itu mengatur iterasi PBKDF2 menjadi 1.000 agar selesai dalam sepersekian detik; demo dan konstruktor bawaan memakai nilai OWASP. Jalankan dengan `python -m unittest test_privacy_app`:

```
..........
----------------------------------------------------------------------
Ran 10 tests in 0.021s

OK
```

Pustaka standar Java tidak memiliki kerangka uji, sehingga **Kode 11.11** membawa sendiri pelari uji sepuluh baris: setiap uji adalah sebuah metode, sebuah tabel mendaftar uji-uji itu beserta namanya, dan pelari mencetak `ok` atau `FAILED` untuk masing-masing serta keluar dengan status 1 bila ada yang gagal, dan hanya itulah yang dibutuhkan skrip *build*.

Kode 11.11 `PrivacyAppTest.java`: sepuluh sifat yang sama dalam Java, dengan pelari uji minimal.

```
import java.util.HexFormat;
import java.util.List;

/** Tests for the G7 lab, with a ten-line runner instead of a framework.
 *  Run: java -cp out PrivacyAppTest */
public class PrivacyAppTest {
    static final String PW = "kopi-susu-tanpa-gula-2026";
    static final String IP = "203.0.113.7";
    static long now;

    interface Test {
        void run() throws Exception;
    }

    static void check(boolean condition, String what) {
        if (!condition) throw new AssertionError(what);
    }

    static void refused(Runnable action, String what) {
        try {
            action.run();
        } catch (PrivacyApp.PrivacyException e) {
            return;
        }
        throw new AssertionError("not refused: " + what);
    }

    static PrivacyApp newApp() {
        now = 1790215200L;
        // Few iterations keep the tests fast; the demo uses the OWASP value.
        return new PrivacyApp(() -> now, "test-secret".getBytes(),
                              new Pbkdf2Hasher(1000), 3, 60);
    }

    static void pbkdf2KnownAnswer() {
        // RFC 7914, section 11: PBKDF2-HMAC-SHA256("passwd", "salt", 1, 64)
        byte[] key = Pbkdf2Hasher.derive("passwd", "salt".getBytes(), 1);
        check(HexFormat.of().formatHex(key).startsWith("55ac046e56e3089fec1691c2"),
              "known answer");
    }
```

Kode 11.11 (Lanjutan) *PrivacyAppTest.java: sepuluh sifat yang sama dalam Java, dengan pelari uji minimal.*

```
    static void recipeIsTheSameInPython() {
        String expected = "pbkdf2_sha256$1000$cHJvZml0aW5hLWxhYi0xNg==$"
                + "RJZs9e03R01zF3dwuFJoifK15hAmgK6TPoBFUVtqkrE=";
        Pbkdf2Hasher h = new Pbkdf2Hasher(1000);
        check(h.hash(PW, "profitina-lab-16".getBytes()).equals(expected), "same");
        check(h.verify(expected, PW), "verifies");
    }

    static void hashIsSaltedAndVerifies() {
        Pbkdf2Hasher h = new Pbkdf2Hasher(1000);
        String a = h.hash(PW), b = h.hash(PW);
        check(!a.equals(b), "fresh salt");
        check(h.verify(a, PW) && !h.verify(a, PW + "x"), "verify");
        check(!a.contains(PW), "no plain text");
    }

    static void consentMustBeExplicitlyTrue() {
        PrivacyApp app = newApp();
        for (Boolean value : new Boolean[] {false, null}) {
            refused(() -> app.register("ayu@example.com", PW, value, IP),
                    "consent");
        }
        app.register("ayu@example.com", PW, true, IP);
        check(app.consents.get(0).event.equals("given"), "logged");
        check(app.consents.get(0).version.equals("1.0"), "version");
        check(!app.dump().contains(IP), "only a keyed hash of the IP");
    }

    static void sameMessageForUnknownEmailAndWrongPassword() {
        PrivacyApp app = newApp();
        app.register("ayu@example.com", PW, true, IP);
        String[] msg = new String[2];
        try {
            app.login("ayu@example.com", "wrong-password-123", "198.51.100.1");
        } catch (PrivacyApp.PrivacyException e) {
            msg[0] = e.getMessage();
        }
        try {
            app.login("nobody@example.com", PW, "198.51.100.2");
        } catch (PrivacyApp.PrivacyException e) {
            msg[1] = e.getMessage();
        }
        check(PrivacyApp.GENERIC.equals(msg[0])
              && PrivacyApp.GENERIC.equals(msg[1]),
              "same message");
    }
```

Kode 11.11 (Lanjutan) *PrivacyAppTest.java: sepuluh sifat yang sama dalam Java, dengan pelari uji minimal.*

```
    static void slidingWindowBlocksThenReleases() {
        PrivacyApp app = newApp();
        app.register("ayu@example.com", PW, true, IP);
        for (int i = 0; i < 3; i++) {
            now += 1;
            refused(() -> app.login("ayu@example.com", "wrong-password-123", IP),
                    "wrong password");
        }
        refused(() -> app.login("ayu@example.com", PW, IP), "blocked");
        now += 60;
        check(app.login("ayu@example.com", PW, IP) != null, "released");
    }

    static void oldHashIsUpgradedAtLogin() {
        PrivacyApp app = newApp();
        int uid = app.register("ayu@example.com", PW, true, IP);
        app.useHasher(new Pbkdf2Hasher(2000));
        check(app.users.get(uid).password.contains("$1000$"), "old recipe");
        app.login("ayu@example.com", PW, IP);
        check(app.users.get(uid).password.contains("$2000$"), "upgraded");
    }

    static void exportContainsOnlyOwnData() {
        PrivacyApp app = newApp();
        app.register("ayu@example.com", PW, true, IP);
        app.register("budi@example.com", PW, true, "198.51.100.23");
        String ayu = app.login("ayu@example.com", PW, IP);
        String budi = app.login("budi@example.com", PW, "198.51.100.23");
        app.post(ayu, "hello from ayu", null);
        app.post(budi, "hello from budi", null);
        String json = app.export(ayu);
        check(json.contains("\"email\": \"ayu@example.com\""), "own email");
        check(json.contains("hello from ayu") && !json.contains("budi"),
              "own only");
        check(!json.contains("password"), "no hash in the export");
    }
```

Kode 11.11 (Lanjutan) *PrivacyAppTest.java: sepuluh sifat yang sama dalam Java, dengan pelari uji minimal.*

```
    static void changesNeedThePassword() {
        PrivacyApp app = newApp();
        app.register("ayu@example.com", PW, true, IP);
        String t1 = app.login("ayu@example.com", PW, IP);
        String t2 = app.login("ayu@example.com", PW, IP);
        refused(() -> app.changeEmail(t1, "not-the-password", "new@example.com"),
                "email change");
        app.changePassword(t1, PW, "a-brand-new-long-passphrase");
        check(app.sessions.containsKey(t1), "this device stays");
        check(!app.sessions.containsKey(t2), "other devices signed out");
    }

    static void deleteLeavesNothingPersonal() {
        PrivacyApp app = newApp();
        app.register("ayu@example.com", PW, true, IP);
        app.register("budi@example.com", PW, true, "198.51.100.23");
        String ayu = app.login("ayu@example.com", PW, IP);
        String budi = app.login("budi@example.com", PW, "198.51.100.23");
        int first = app.post(ayu, "a question from ayu", null);
        app.post(ayu, "a note nobody answered", null);
        int reply = app.post(budi, "an answer from budi", first);
        app.changeEmail(ayu, PW, "ayu.new@example.com");
        String storedHash = app.users.get(1).password;
        String ipHash = app.ipHash(IP);
        refused(() -> app.deleteAccount(ayu, "wrong-one"), "wrong password");
        String tomb = app.deleteAccount(ayu, PW);
        String dump = app.dump();
        for (String trace : List.of("ayu@example.com", "ayu.new@example.com", IP,
                ipHash, storedHash, "a question from ayu", "nobody answered")) {
            check(!dump.contains(trace), "left behind: " + trace);
        }
        check(app.posts.get(first).text.equals("[deleted]"), "tombstone post");
        check(app.posts.get(reply).parent == first, "thread intact");
        long events = app.consents.stream().filter(c -> c.subject.equals(tomb))
                .map(c -> c.event).filter(e -> e.equals("given")
                        || e.equals("withdrawn")).count();
        check(events == 2, "given and withdrawn");
        check(app.limiter.failures.keySet().stream()
                 .noneMatch(k -> k.contains("ayu")),
              "limiter forgot the account");
        refused(() -> app.export(ayu), "session gone");
        check(dump.contains("budi@example.com"), "others untouched");
    }
```

Kode 11.11 (Lanjutan) *PrivacyAppTest.java: sepuluh sifat yang sama dalam Java, dengan pelari uji minimal.*

```
    public static void main(String[] args) {
        Object[][] tests = {
            {"pbkdf2 known answer", (Test) PrivacyAppTest::pbkdf2KnownAnswer},
            {"recipe is the same in Python",
                (Test) PrivacyAppTest::recipeIsTheSameInPython},
            {"hash is salted and verifies",
                (Test) PrivacyAppTest::hashIsSaltedAndVerifies},
            {"consent must be explicitly true",
                (Test) PrivacyAppTest::consentMustBeExplicitlyTrue},
            {"same message for unknown email and wrong password",
                (Test) PrivacyAppTest::sameMessageForUnknownEmailAndWrongPassword},
            {"sliding window blocks then releases",
                (Test) PrivacyAppTest::slidingWindowBlocksThenReleases},
            {"old hash is upgraded at login",
                (Test) PrivacyAppTest::oldHashIsUpgradedAtLogin},
            {"export contains only own data",
                (Test) PrivacyAppTest::exportContainsOnlyOwnData},
            {"changes need the password",
                (Test) PrivacyAppTest::changesNeedThePassword},
            {"delete leaves nothing personal",
                (Test) PrivacyAppTest::deleteLeavesNothingPersonal},
        };
        int failed = 0;
        for (Object[] t : tests) {
            try {
                ((Test) t[1]).run();
                System.out.println("ok      " + t[0]);
            } catch (Throwable e) {
                failed++;
                System.out.println("FAILED  " + t[0] + ": " + e);
            }
        }
        System.out.println(tests.length + " tests, " + failed + " failed");
        if (failed > 0) System.exit(1);
    }
}
```

```
ok      pbkdf2 known answer
ok      recipe is the same in Python
ok      hash is salted and verifies
ok      consent must be explicitly true
ok      same message for unknown email and wrong password
ok      sliding window blocks then releases
ok      old hash is upgraded at login
ok      export contains only own data
ok      changes need the password
ok      delete leaves nothing personal
10 tests, 0 failed
```

Uji bernama “recipe is the same in Python” adalah peragaan kecil tentang mengapa format *hash* yang menjelaskan dirinya sendiri penting di luar satu program: *hash* yang ditulis oleh satu implementasi, dalam satu bahasa, dapat diverifikasi di bahasa lain. Tim yang memindahkan sebuah layanan dari Python ke Java, atau sebaliknya, tidak perlu meminta penggunanya mengatur ulang kata sandi.

11.11 Profitina dalam Praktik

Langkah-langkah dalam bab ini berada di modul pelindungan data tersendiri dalam aplikasi Profitina, berdampingan dengan modul pemindaian, komunitas, dan PROFIT yang digambarkan di Bab 9 [1]. Yang sudah berjalan, per 24 September 2026:

Sudah berjalan. Pendaftaran hanya dengan email dan kata sandi; kotak persetujuan yang tidak pernah tercentang lebih dulu, dengan pemberitahuan singkat dwibahasa; setiap persetujuan dicatat dengan versi kebijakan dan waktunya. Kata sandi disimpan dengan Argon2id, dengan *hash* lama ditingkatkan saat login. Pembatasan laju login per identitas akun dan, untuk lalu lintas anonim, per alamat IP, disimpan hanya di memori. Verifikasi email dengan kode sekali pakai sebagai lapisan anti-bot pertama. Panel *Akun & Data Pribadi* di tab *Pantau*, sejak 19 Agustus 2026, dengan ekspor JSON, penggantian email, penggantian kata sandi, dan penghapusan akun permanen, masing-masing dikonfirmasi dengan kata sandi; penghapusan yang membersihkan data pribadi di semua modul dan meninggalkan *tombstone* anonim; data produksi dihapus atau dianonimkan paling lambat 30 hari setelah penghapusan. Profil dengan persetujuan per butir, privat secara bawaan; *handle* netral; tanpa kolom telepon; email tidak pernah dapat dipublikasikan. Pesan langsung yang tidak dapat dibaca administrator kecuali salinan beku dari pesan yang dilaporkan. Catatan galat tanpa isi mentah maupun *header*, dengan alamat IP di-*hash*, dibersihkan setelah 90 hari. Kebijakan Privasi (versi 1.15, berlaku sejak 9 September 2026) serta Syarat dan Ketentuan (versi 1.1) sebagai HTML dwibahasa di aplikasi, dengan situs pemasaran menautkannya. *Runbook* kegagalan pelindungan untuk tenggat 72 jam, dengan pemberitahuan yang sudah disiapkan. Satu alamat kontak untuk segala urusan: info@profitina.com [1], [7].

Aturan internal. Setiap fitur yang menyentuh data pribadi memperbarui ROPA dan, bila perlu, Kebijakan Privasi. Hak-hak, keamanan, dan pemulihan akun tetap gratis selamanya. Hanya pengguna yang dapat memulai penghapusan. Fitur akademik hanya memproses catatan bibliografis yang terbit, tidak pernah data profil, foto, atau komunitas, dan orang yang tidak ditemukan tidak mendapat skor [1].

Yang masih di depan. Kepatuhan adalah proses yang berkelanjutan, bukan sertifikat: PP 33/2026 mulai berlaku pada Januari 2027, lembaga pengawas masih dibentuk, dan Profitina, seperti setiap pengendali, akan mencocokkan prosedurnya dengan rincian peraturan itu seiring penerapannya.

11.12 Kesalahpahaman yang Sering Terjadi

“Lembaganya belum ada, jadi UU PDP belum ditegakkan.” Kewajiban undang-undang berlaku penuh sejak 17 Oktober 2024, dan Komdigi mengawasi pada masa peralihan [4]. Menunggu lembaga berarti menunggu untuk didapati tidak siap.

“Kami mengenkripsi kata sandi.” Kata sandi harus di-*hash* dengan fungsi *hash* kata sandi yang lambat dan ber-*salt*, bukan dienkripsi. Enkripsi dirancang untuk dibalik oleh siapa pun yang memegang kuncinya (Subbab 11.5.1).

“SHA-256 itu aman, jadi cocok untuk kata sandi.” SHA-256 adalah *hash* yang aman untuk memeriksa bahwa data tidak berubah, dan jauh terlalu cepat untuk kata sandi (**Tabel 11.1**, Persamaan (11.1)).

“Kotak persetujuan yang tercentang secara bawaan tetap merupakan persetujuan.” Kotak itu mencatat pilihan perancang. Pasal 22 mensyaratkan tindakan yang eksplisit dan terekam, dan persetujuan yang tidak memenuhi syarat batal demi hukum [2].

“Kami punya `consented = true` di tabel pengguna, jadi kami dapat membuktikan persetujuan.” Nilai boolean tidak menyatakan apa yang disetujui, kapan, atau di bawah versi mana, dan melupakan penarikan. Persetujuan adalah log peristiwa (Subbab 11.6.2).

"Menghapus baris pengguna berarti menghapus penggunanya." Data pribadi juga hidup di tulisan, log, sesi, *cache*, dan cadangan. Penghapusan adalah *purge* di setiap tabel, dengan *tombstone* hanya di tempat konten orang lain membutuhkannya, dan waktu yang dinyatakan untuk cadangan (**Gambar 11.5**).

"Pembatasan laju hanya untuk menghentikan satu penyerang." *Credential stuffing* memakai banyak akun dan banyak alamat dengan laju rendah masing-masing. Batasi per akun *dan* per alamat, tolak sebelum meng-*hash*, dan pertahankan pesan galat yang sama (Subbab 11.7.3).

"Fungsi ekspor sudah ada, jadi pengguna sudah memiliki hak akses." Bila pengguna tidak dapat menemukannya, mereka tidak memilikinya. Catatan Profitina sendiri memperlihatkan perbedaan antara diterapkan dan dapat dijangkau (Subbab 11.8.2).

"Meng-*hash* alamat IP membuatnya anonim." *Hash* biasa atas alamat IPv4 dapat dibalik dengan meng-*hash* keempat miliar alamat. Pakai *hash* berkunci dengan rahasia yang disimpan di luar data, simpan hanya yang diperlukan tujuannya, dan hapus bersama akunnya.

11.13 Ringkasan Bab

- UU PDP No. 27/2022, yang disahkan 17 Oktober 2022 dengan masa transisi dua tahun sampai 17 Oktober 2024, mendefinisikan data pribadi umum dan spesifik (Pasal 4), pengendali dan prosesor, asas-asas (Pasal 16), dasar pemrosesan (Pasal 20), syarat persetujuan (Pasal 21, 22), hak subjek data (Pasal 5 sampai 13), kewajiban keamanan (Pasal 35, 39), penghapusan (Pasal 43 sampai 45), pemberitahuan kegagalan pelindungan dalam 72 jam (Pasal 46), dan sanksi sampai 2% dari pendapatan tahunan (Pasal 57), ditambah ketentuan pidana.
- PP 33/2026, peraturan pelaksananya, ditetapkan dan diundangkan pada 16 Juli 2026 (225 pasal, 12 bab) dan mulai berlaku pada 16 Januari 2027. Perpres lembaga pengawas diajukan pada 20 Mei 2026; Komdigi mengawasi sementara itu; seorang pakar memperkirakan lembaga yang efektif paling cepat pada 2028.
- Privasi sejak rancangan berarti keputusan-keputusan konkret: kumpulkan hanya email dan kata sandi, jadikan privat secara bawaan, catat pemrosesan dalam ROPA, bersihkan log dengan tugas yang benar-benar berjalan, dan sediakan satu rumah untuk teks hukum.
- Kata sandi disimpan dengan *hash* yang lambat, ber-*salt*, dan menjelaskan dirinya sendiri. Persamaan (11.1) menjelaskan alasannya: pembela mengendalikan biaya setiap tebakan. Argon2id (RFC 9106) bersifat *memory-hard*; PBKDF2 dengan 600.000 iterasi adalah cadangan dari pustaka standar; *hash* ditingkatkan saat login.
- Persetujuan adalah peristiwa yang eksplisit dan tercatat dengan versi dan waktu, tidak pernah tercentang lebih dulu, dapat ditarik, dan granular sejauh produknya granular.
- Pembatas berjendela geser (Persamaan (11.2), Persamaan (11.3)) pada kunci akun maupun alamat, yang diperiksa sebelum *hashing* dan memakai satu pesan galat, menangkal *brute force* dan *credential stuffing*.
- Hak dapat dijangkau bila seseorang dapat memakainya dari antarmuka: panel *Akun & Data Pribadi* milik Profitina menyediakan ekspor, pembetulan, dan penghapusan, gratis selamanya, dengan penghapusan hanya oleh pengguna dan *tombstone* hanya di tempat orang lain membutuhkannya.
- Lab G7 menerapkan semuanya dalam Python dan Java dan membuktikan dengan uji bahwa persetujuan itu eksplisit, pembatas bekerja, *hash* ditingkatkan, ekspor lengkap dan eksklusif, dan penghapusan tidak meninggalkan apa pun yang bersifat pribadi.

11.14 Soal Latihan

Soal 11.1. Golongkan setiap butir sebagai data pribadi umum, data pribadi spesifik, atau bukan data pribadi menurut Pasal 4: (a) nama lengkap pelanggan; (b) templat sidik jari yang dipakai untuk membuka loker laundry; (c) saldo bank pelanggan; (d) jam buka laundry; (e) alergi medis yang disebut pelanggan agar deterjen tertentu tidak dipakai; (f) agama yang dicantumkan di formulir keanggotaan.

Soal 11.2. Untuk setiap kegiatan pemrosesan aplikasi laundry, sebutkan dasar pemrosesan yang paling tepat dari Pasal 20: (a) menyimpan nomor telepon pelanggan untuk menelepon ketika pesanan selesai; (b) menyimpan faktur selama jangka waktu yang diwajibkan hukum pajak; (c) mengirim buletin promosi bulanan; (d) menyimpan log server selama beberapa minggu untuk mendeteksi serangan.

Soal 11.3. Sebuah kata sandi terdiri atas 10 huruf kecil. Dengan memakai Persamaan (11.1) dan laju dalam **Tabel 11.1**, hitung waktu untuk mencoba setiap kemungkinan dengan SHA-256 ber-*salt* dan dengan PBKDF2 600.000 iterasi. Lalu jelaskan dalam dua kalimat mengapa keunggulan nyata fungsi yang lambat lebih kecil daripada yang ditunjukkan perbandingan itu, dan mengapa hal itu mendukung Argon2id.

Soal 11.4. Dua pengguna memilih kata sandi yang sama. Jelaskan seperti apa string tersimpan keduanya dengan dan tanpa *salt*, dan apa yang dapat dilakukan penyerang yang telah mencuri tabelnya dalam masing-masing kasus.

Soal 11.5. Sebuah pembatas mengizinkan $n = 3$ kegagalan dalam $w = 60$ detik untuk satu kunci. Percobaan datang pada $t = 0{,}10{,}20$ (kata sandi salah), 30 (kata sandi benar), 61 (salah), 65 (benar), dan 71 (benar). Dengan memakai Persamaan (11.2) dan Persamaan (11.3), nyatakan untuk setiap percobaan apakah percobaan itu diperiksa atau ditolak, dan berikan waktu tunggu untuk setiap penolakan. Anggaplah percobaan yang ditolak tidak dicatat sebagai kegagalan.

Soal 11.6. Sebuah pembatas berjendela tetap mengizinkan 3 kegagalan per menit jam. Tunjukkan urutan percobaan yang menghasilkan 6 kegagalan dalam 2 detik. Apa yang diizinkan jendela geser Persamaan (11.2) untuk urutan yang sama?

Soal 11.7. Tabel tulisan memuat: tulisan 1 oleh Ayu (tanpa induk), tulisan 2 oleh Budi membalas 1, tulisan 3 oleh Ayu membalas 2, tulisan 4 oleh Ayu (tanpa induk), tulisan 5 oleh Ayu membalas 4. Ayu menghapus akunnya. Dengan aturan `delete_account`, baris mana yang dihapus dan mana yang menjadi cangkang `[deleted]`? Apakah balasan Budi masih melekat pada sesuatu?

Soal 11.8. Kebijakan Privasi berubah dari versi 1.0 ke 1.1. Tulislah, dalam Python dan Java, fungsi `needs_consent` (Java: `needsConsent`) yang menyatakan apakah peristiwa `given` terakhir milik pengguna yang sedang login adalah untuk versi yang berlaku, dan fungsi `accept_policy` (Java: `acceptPolicy`) yang mencatat peristiwa `given` yang baru. Peragakan keduanya.

Soal 11.9. Sebuah kegagalan pelindungan ditemukan pada Senin pukul 21.00 WIB. Paling lambat kapan pemberitahuan tertulis menurut Pasal 46 harus dikirim? Dengan fase-fase *runbook* Profitina (penahanan pada jam ke-0 sampai ke-2, penilaian pada jam ke-2 sampai ke-12, pemberitahuan pada jam ke-12 sampai ke-48), sebutkan pukul berapa setiap fase berakhir.

Soal 11.10. Sebelum 19 Agustus 2026, fungsi ekspor dan penghapusan Profitina sudah ada tetapi tidak memiliki tombol. Gambarkan sebuah uji yang akan gagal pada saat itu dan lolos sesudahnya, dan jelaskan mengapa uji unit di lab tidak dapat mendeteksi celah itu.

Soal 11.11. Sebuah server memverifikasi 1.000 login per jam dengan PBKDF2 600.000 iterasi (230 ms masing-masing pada satu inti, **Tabel 11.1**). Berapa waktu CPU per jam yang diperlukan? Seorang penyerang mengirim 100 permintaan login per detik. Apa yang terjadi dengan dan tanpa "tolak sebelum meng-*hash*"?

Soal 11.12. Pindahkan tabel-tabel lab G7 ke SQLite (modul `sqlite3` milik Python) dan tulislah penghapusan sebagai satu transaksi pernyataan SQL. Tampilkan tabel-tabel itu setelah Ayu dihapus dari riwayat uji terakhir lab.

11.15 Penyelesaian Soal Latihan

Penyelesaian 11.1. (a) Data pribadi umum: nama lengkap tercantum di Pasal 4. (b) Spesifik: data biometrik. (c) Spesifik: data keuangan pribadi. (d) Bukan data pribadi: data itu tentang sebuah usaha, bukan orang. (e) Spesifik: informasi kesehatan, sekalipun disebut sambil lalu; data itu layak mendapat perlakuan yang lebih ketat, dan aplikasi sebaiknya menimbang apakah perlu menyimpannya sama sekali atau cukup sebuah penanda "pakai deterjen B". (f) Umum: agama tercantum di antara kategori umum.

Penyelesaian 11.2. (a) Pemenuhan kewajiban perjanjian: menelepon tentang pesanan adalah bagian dari memberikan layanan yang diminta pelanggan. (b) Pemenuhan kewajiban hukum pengendali. (c) Persetujuan: pemasaran tidak diperlukan untuk perjanjian, sehingga memerlukan persetujuannya sendiri yang eksplisit, terpisah, dan dapat ditarik. (d) Kepentingan lain yang sah: pencatatan keamanan melindungi layanan dan penggunanya, asalkan dibatasi pada yang diperlukan dan disimpan singkat saja, dan keseimbangannya dengan hak subjek didokumentasikan.

Penyelesaian 11.3. $N = 26^{10} = 141\,167\,095\,653\,376 \approx 1.41 \times 10^{14}$. Dengan SHA-256, $T = N/R \approx 1.41 \times 10^{14} / 1.67 \times 10^{6} \approx 8.4 \times 10^{7}$ detik, sekitar 2,7 tahun pada satu inti. Dengan PBKDF2 600.000 iterasi, $T \approx 1.41 \times 10^{14} / 4.35 \approx 3.2 \times 10^{13}$ detik, sekitar satu juta tahun. Perbandingannya sama seperti di Subbab 11.5.2, sekitar 3.8×10^{5}. Penyerang sungguhan memakai prosesor grafis atau perangkat keras khusus yang menghitung SHA-256 dan PBKDF2 jauh lebih cepat daripada satu inti CPU dan secara paralel besar-besaran, sehingga waktu mutlaknya menyusut drastis untuk keduanya; kebutuhan memori Argon2id membatasi berapa banyak tebakan yang dapat dijalankan paralel oleh perangkat semacam itu, sehingga keunggulan pembela tetap terjaga di tempat yang penting.

Penyelesaian 11.4. Tanpa *salt*, string tersimpan kedua pengguna identik, sehingga penyerang langsung tahu bahwa keduanya memakai kata sandi yang sama; lebih buruk lagi, penyerang dapat menghitung sekali saja *hash* jutaan kata sandi umum lalu mencocokkan setiap pengguna dari setiap tabel curian dengan satu daftar itu. Dengan *salt* acak 16 bait per pengguna, kedua string tersimpan sama sekali berbeda, kata sandi yang sama tidak terlihat, dan setiap tebakan harus dihitung ulang terpisah untuk *salt* setiap pengguna, sehingga kerja penyerang berlipat sebanyak jumlah pengguna.

Penyelesaian 11.5. $t = 0$, 10, 20: diperiksa, salah; himpunan kegagalan menjadi $\{0\}$, $\{0,10\}$, $\{0,10,20\}$. $t = 30$: $F = \{0,10,20\}$ memiliki 3 unsur, sehingga ditolak sebelum kata sandinya diperiksa, dengan retry_after $= 0 + 60 - 30 = 30$ detik. $t = 61$: jendelanya $(1,61]$, sehingga kegagalan 0 sudah keluar; $F = \{10,20\}$, percobaan diperiksa, salah, dan F menjadi $\{10,20,61\}$. $t = 65$: 3 kegagalan dalam $(5,65]$, ditolak, retry_after $= 10 + 60 - 65 = 5$ detik. $t = 71$: jendelanya $(11,71]$, $F = \{20,61\}$, diperiksa, kata sandi benar: berhasil masuk, dan kunci akun dilupakan. Menjalankan pembatas lab pada urutan yang sama mencetak keputusan yang persis sama.

Penyelesaian 11.6. Gagal 3 kali pada 09:00:59 (diizinkan: tiga yang pertama pada menit itu) dan 3 kali pada 09:01:00 (diizinkan: tiga yang pertama pada menit baru): 6 kegagalan dalam sekitar 1 detik. Dengan jendela geser, setelah tiga kegagalan pada 09:00:59 himpunan F memuat tiga waktu dalam 60 detik terakhir sampai 09:01:59, sehingga setiap percobaan sampai saat itu ditolak; hanya terjadi 3 kegagalan.

Penyelesaian 11.7. Tulisan Ayu adalah 1, 3, 4, dan 5. Sebuah tulisan disimpan sebagai cangkang hanya bila ada tulisan *milik orang lain* yang menjadikannya induk. Tulisan 1 dibalas Budi dengan tulisan 2, sehingga menjadi cangkang `[deleted]`. Tulisan 3 tidak memiliki balasan: dihapus. Tulisan 4 memiliki balasan, tetapi itu tulisan 5 milik Ayu sendiri: dihapus, bersama tulisan 5 yang tidak memiliki balasan. Tulisan 2 milik Budi masih melekat pada tulisan 1, sang cangkang; utasnya terbaca "[deleted]" lalu jawaban Budi. Tulisan 2 milik Budi tidak lagi memiliki balasan, karena tulisan 3 milik Ayu sudah hilang.

Penyelesaian 11.8. **Kode 11.12** dan **Kode 11.13** menelusuri log persetujuan untuk mencari peristiwa `given` terakhir milik pengguna dan membandingkan versinya dengan versi kebijakan yang berlaku; menerima kebijakan menambahkan peristiwa baru, sehingga log menyimpan keduanya.

Kode 11.12 `sol_policy.py`: meminta persetujuan lagi setelah kebijakan berubah, di atas lab G7.

```
"""Exercise 11.8: ask for consent again when the policy version changes."""
from passwords import Pbkdf2Hasher
from privacy_app import PrivacyApp


def needs_consent(app, token):
    uid = app.user_of(token)
    given = [c["version"] for c in app.consents
             if c["subject"] == uid and c["event"] == "given"]
    return not given or given[-1] != app.policy_version


def accept_policy(app, token, ip):
    uid = app.user_of(token)
    app.consents.append({"subject": uid, "event": "given",
                         "version": app.policy_version,
                         "at": app.stamp(), "ip_hash": app.ip_hash(ip)})


clock = lambda: 1790215200
app = PrivacyApp(clock, b"s", hasher=Pbkdf2Hasher(iterations=1000))
app.register("ayu@example.com", "kopi-susu-tanpa-gula-2026", True, "203.0.113.7")
token = app.login("ayu@example.com", "kopi-susu-tanpa-gula-2026", "203.0.113.7")
print("needs consent under 1.0:", needs_consent(app, token))
app.policy_version = "1.1"
print("needs consent under 1.1:", needs_consent(app, token))
accept_policy(app, token, "203.0.113.7")
print("after accepting 1.1:   ", needs_consent(app, token))
print([c["version"] for c in app.consents])
```

```
needs consent under 1.0: False
needs consent under 1.1: True
after accepting 1.1:    False
['1.0', '1.1']
```

Kode 11.13 `SolPolicy.java`: hal yang sama dalam Java (kolom `policyVersion` pada `PrivacyApp` sengaja tidak `final`).

```java
/** Exercise 11.8: ask for consent again when the policy version changes. */
public class SolPolicy {
    static boolean needsConsent(PrivacyApp app, String token) {
        String uid = String.valueOf(app.userOf(token));
        String last = null;
        for (PrivacyApp.Consent c : app.consents) {
            if (c.subject.equals(uid) && c.event.equals("given")) last = c.version;
        }
        return last == null || !last.equals(app.policyVersion);
    }

    static void acceptPolicy(PrivacyApp app, String token, String ip) {
        PrivacyApp.Consent c = new PrivacyApp.Consent();
        c.subject = String.valueOf(app.userOf(token));
        c.event = "given";
        c.version = app.policyVersion;
        c.at = app.stamp();
        c.ipHash = app.ipHash(ip);
        app.consents.add(c);
    }

    public static void main(String[] args) {
        PrivacyApp app = new PrivacyApp(() -> 1790215200L, "s".getBytes(),
                                        new Pbkdf2Hasher(1000), 3, 60);
        String pw = "kopi-susu-tanpa-gula-2026", ip = "203.0.113.7";
        app.register("ayu@example.com", pw, true, ip);
        String token = app.login("ayu@example.com", pw, ip);
        System.out.println("needs consent under 1.0: " + needsConsent(app, token));
        app.policyVersion = "1.1";
        System.out.println("needs consent under 1.1: " + needsConsent(app, token));
        acceptPolicy(app, token, ip);
        System.out.println("after accepting 1.1:    " + needsConsent(app, token));
    }
}
```

```
needs consent under 1.0: false
needs consent under 1.1: true
after accepting 1.1:    false
```

Antarmuka sungguhan akan memanggil `needs_consent` setelah login dan, bila hasilnya benar, menampilkan pemberitahuan baru dengan kotak yang belum tercentang sebelum hal lain.

Penyelesaian 11.9. $3 \times 24 = 72$ jam setelah Senin pukul 21.00 WIB adalah Kamis pukul 21.00 WIB, saat paling akhir untuk pemberitahuan. Penahanan berakhir Senin pukul 23.00 (jam ke-2); penilaian berakhir Selasa pukul 09.00 (jam ke-12); fase pemberitahuan menurut *runbook* berakhir Rabu pukul 21.00 (jam ke-48), satu hari penuh sebelum tenggat hukum, dan itulah ruang cadangan yang semestinya disisakan sebuah *runbook*.

Penyelesaian 11.10. Uji ujung ke ujung mengendalikan peramban sungguhan: masuk sebagai pengguna uji, membuka tab *Pantau*, mencari panel *Akun & Data Pribadi* beserta kendali “ekspor” dan “hapus” menurut label yang terlihat, menekan ekspor, lalu memeriksa bahwa berkas JSON berisi email pengguna itu terunduh. Sebelum 19 Agustus 2026 uji itu gagal pada langkah “temukan panel”; sesudahnya uji itu lolos. Uji unit seperti di lab memanggil `export` dan `delete_account` secara langsung, sehingga membuktikan bahwa fungsinya bekerja tetapi tidak mengatakan apa pun tentang apakah seseorang dapat menjangkaunya.

Penyelesaian 11.11. $1000 \times 0.23\ \text{s} = 230$ detik CPU per jam, sekitar 6,4% dari satu inti: terjangkau. Pada 100 permintaan per detik, meng-*hash* setiap percobaan akan memerlukan $100 \times 0.23 = 23$ detik-inti per detik, jauh lebih banyak daripada yang dimiliki server kecil, sehingga login yang jujur akan mengantre di belakang serangan dan layanan macet. Dengan “tolak

sebelum meng-*hash*", setiap permintaan di atas batas hanya memakan satu pencarian di *dictionary* dan satu pemangkasan antrean, dan CPU tetap tersedia bagi pengguna sungguhan. (Batas di tingkat jaringan di depan aplikasi menambah satu lapisan lagi.)

Penyelesaian 11.12. Satu tabel untuk setiap jenis catatan, dan penghapusan dari **Kode 11.5** sebagai SQL di dalam satu transaksi, sehingga semuanya terhapus atau tidak ada yang terhapus. **Kode 11.14** membangun riwayatnya, menghapus Ayu (pengguna 1), dan mencetak apa yang tersisa.

Kode 11.14 `sol_sqlite.py`: tabel-tabel lab G7 di SQLite, dengan penghapusan sebagai satu transaksi.

```
"""Exercise 11.12: the erasure of lab G7 as one SQLite transaction."""
import sqlite3

db = sqlite3.connect(":memory:")
db.executescript("""
CREATE TABLE users (id INTEGER PRIMARY KEY, email TEXT UNIQUE NOT NULL,
                    password TEXT NOT NULL, created TEXT NOT NULL);
CREATE TABLE posts (id INTEGER PRIMARY KEY, author INTEGER REFERENCES users(id),
                    text TEXT NOT NULL, parent INTEGER REFERENCES posts(id),
                    at TEXT NOT NULL);
CREATE TABLE consents (id INTEGER PRIMARY KEY, subject TEXT NOT NULL,
                       event TEXT NOT NULL, version TEXT NOT NULL,
                       at TEXT NOT NULL, ip_hash TEXT);
CREATE TABLE sessions (token TEXT PRIMARY KEY, user_id INTEGER NOT NULL);
INSERT INTO users VALUES (1, 'ayu@example.com', 'pbkdf2_sha256$...', 't0'),
                         (2, 'budi@example.com', 'pbkdf2_sha256$...', 't0');
INSERT INTO posts VALUES (1, 1, 'a question from ayu', NULL, 't1'),
                         (2, 1, 'a note nobody answered', NULL, 't1'),
                         (3, 2, 'an answer from budi', 1, 't2');
INSERT INTO consents VALUES (1, '1', 'given', '1.0', 't0', '2f7a7c6e655482f7'),
                            (2, '2', 'given', '1.0', 't0', 'aaaa');
INSERT INTO sessions VALUES ('tok-a', 1), ('tok-b', 2);
""")

def erase(db, uid, tomb, now):
    with db:                                    # one transaction: all or nothing
        db.execute("""UPDATE posts SET author = NULL, text = '[deleted]'
                      WHERE author = ? AND id IN (SELECT parent FROM posts
                      WHERE parent IS NOT NULL AND author IS NOT ?)""",
                   (uid, uid))
        db.execute("DELETE FROM posts WHERE author = ?", (uid,))
        db.execute("UPDATE consents SET subject = ?, ip_hash = NULL "
                   "WHERE subject = ?", (tomb, str(uid)))
        db.execute("INSERT INTO consents (subject, event, version, at) "
                   "VALUES (?, 'withdrawn', '1.0', ?)", (tomb, now))
        db.execute("DELETE FROM sessions WHERE user_id = ?", (uid,))
        db.execute("DELETE FROM users WHERE id = ?", (uid,))

erase(db, 1, "deleted-1", "t3")
for table in ("users", "posts", "consents", "sessions"):
    for row in db.execute("SELECT * FROM " + table):
        print(table, row)
```

```
users (2, 'budi@example.com', 'pbkdf2_sha256$...', 't0')
posts (1, None, '[deleted]', None, 't1')
posts (3, 2, 'an answer from budi', 1, 't2')
consents (1, 'deleted-1', 'given', '1.0', 't0', None)
consents (2, '2', 'given', '1.0', 't0', 'aaaa')
consents (3, 'deleted-1', 'withdrawn', '1.0', 't3', None)
sessions ('tok-b', 2)
```

Baris pengguna Ayu, sesinya, catatannya yang tak berbalas, dan *hash* IP-nya hilang; pertanyaannya bertahan hanya sebagai cangkang karena jawaban Budi menunjuk kepadanya; log persetujuan membuktikan persetujuan dan penarikan di bawah pengenal anonim `deleted-1`. Urutan dua pernyataan pertama penting: cangkang harus dibuat sebelum tulisan-tulisan lain milik pengguna itu dihapus. ## □U□Daftar Pustaka

[1] Profitina, *Profitina Whitepaper — Public Version v24*, Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[2] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*, Lembaran Negara 2022 No. 196, Oct. 17, 2022. [Online]. Available: https://peraturan.bpk.go.id/Details/229798/uu-no-27-tahun-2022

[3] Meridian Hukum, "Peraturan Pemerintah Nomor 33 Tahun 2026 tentang Peraturan Pelaksanaan Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi" (LN 2026 No. 88, TLN No. 7190). [Online]. Available: https://meridianhukum.com/peraturan/pp-no-33-tahun-2026 (accessed Sep. 24, 2026).

[4] Bisnis.com, "Pembentukan Badan PDP dipercepat, implementasi UU PDP dinilai belum efektif," Sep. 13, 2026. [Online]. Available: https://teknologi.bisnis.com/read/20260913/84/2003698/pembentukan-badan-pdp-dipercepat-implementasi-uu-pdp-dinilai-belum-efektif

[5] CNBC Indonesia, "Lembaga perlindungan data belum ada di RI, padahal aturannya sudah ada," Sep. 16, 2026. [Online]. Available: https://www.cnbcindonesia.com/tech/20260916115609-37-768328/lembaga-perlindungan-data-belum-ada-di-ri-padahal-aturannya-sudah-ada

[6] A. Cavoukian, *Privacy by Design: The 7 Foundational Principles*. Toronto, ON, Canada: Information and Privacy Commissioner of Ontario, 2009 (rev. 2011).

[7] Profitina, "Kebijakan Privasi / Privacy Policy," version 1.15, effective Sep. 9, 2026. [Online]. Available: https://app.profitina.com/legal/privacy (accessed Sep. 24, 2026).

[8] National Institute of Standards and Technology, *Digital Identity Guidelines: Authentication and Authenticator Management*, NIST SP 800-63B-4, Aug. 2025.

[9] Password Hashing Competition, "Password Hashing Competition and our recommendation for hashing passwords: Argon2," 2015. [Online]. Available: https://www.password-hashing.net/

[10] A. Biryukov, D. Dinu, D. Khovratovich, and S. Josefsson, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications," IETF RFC 9106, Sep. 2021.

[11] OWASP Foundation, "Password Storage Cheat Sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (accessed Sep. 24, 2026).

[12] K. Moriarty, B. Kaliski, and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1," IETF RFC 8018, Jan. 2017.

[13] Python Software Foundation, "hashlib — Secure hashes and message digests," *The Python Standard Library*. [Online]. Available: https://docs.python.org/3/library/hashlib.html

[14] Oracle, "Java Security Standard Algorithm Names," *Java SE 25 Documentation*. [Online]. Available: https://docs.oracle.com/en/java/javase/25/docs/specs/security/standard-names.html

[15] OWASP Foundation, "Credential Stuffing Prevention Cheat Sheet," *OWASP Cheat Sheet Series*. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Credential_Stuffing_Prevention_Cheat_Sheet.html (accessed Sep. 24, 2026).

[16] Profitina, "Syarat dan Ketentuan / Terms and Conditions," version 1.1, effective Aug. 6, 2026. [Online]. Available: https://app.profitina.com/legal/terms (accessed Sep. 24, 2026).

[17] J. Arkko, M. Cotton, and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation," IETF RFC 5737, Jan. 2010.

[18] H. Schlawack, “argon2-cffi: Argon2 for Python,” version 25.1.0. [Online]. Available: https://argon2-cffi.readthedocs.io/

[19] C. Percival and S. Josefsson, “The scrypt Password-Based Key Derivation Function,” IETF RFC 7914, Aug. 2016, sec. 11, “Test Vectors for PBKDF2 with HMAC-SHA-256.”

Bab 12

PROFIT: Koin Utilitas di Solana

Setelah menyelesaikan bab ini, Anda akan mampu menyatakan, dalam satu kalimat yang cermat, apa itu koin utilitas dan apa yang bukan, serta mengapa PROFIT termasuk jenis pertama dan bukan jenis kedua. Anda akan cukup mengenal Solana, *blockchain* publik tempat kontrak PROFIT berada, untuk membaca pengaturan sebuah token di penjelajah blok (*block explorer*) dan memahami setiap kolomnya: *mint*, akun token, enam desimal, *mint authority*, dan *freeze authority* yang tidak dimiliki PROFIT. Anda akan dapat memisahkan, tanpa ragu, apa yang sudah berjalan pada rilis September 2026 (kontrak di devnet, perolehan, pembelanjaan, transfer antarpengguna, pembelian melalui QRIS) dari apa yang masih ada di peta jalan (mainnet, penjualan, bursa nonkustodian). Anda juga akan membangun Lab G8, sebuah simulator kecil tetapi lengkap dalam Python dan Java yang mereproduksi tabel emisi publik dari whitepaper Profitina, menurunkan hukum pangkat yang tersembunyi di dalamnya, memodelkan pasokan beredar sebagai deret geometri dengan bentuk tertutup, menemukan titik ketika pembakaran menyusul emisi, dan bermain dengan versi mainan dari lantai Rupiah. Setiap angka di lab bersifat ilustratif. Tidak ada satu pun yang merupakan nasihat keuangan, dan tidak ada bagian bab ini yang meramalkan atau menjanjikan harga.

12.1 Posisi Kita Sejauh Ini

Bab 11 ditutup dengan janji yang ditepati: hak-hak atas data pribadi dalam UU PDP bukan sekadar paragraf dalam kebijakan, melainkan tombol yang dapat ditekan pengguna, dan hak yang terpenting, penghapusan, tidak dapat dipicu oleh siapa pun selain pengguna sendiri. Sebelumnya, Bab 8 menjelaskan bagaimana Profitina membiayai dirinya dengan dua paket, Free dan Pro, dan beberapa kali menyebut secara singkat, dengan kotak peringatan, sesuatu bernama PROFIT: pengguna memperolehnya dengan berkontribusi, dapat menukarkannya dengan masa Pro, PROFIT dibakar ketika dibelanjakan, dan “Bab 12 menjelaskan mengapa pembakaran itu penting”. Inilah Bab 12.

PROFIT layak mendapat bab tersendiri karena dua alasan yang menarik ke arah berlawanan.

Alasan pertama, PROFIT sungguh menarik untuk diajarkan. Token dengan jadwal emisi yang menurun dan beberapa mekanisme pembakaran adalah sebuah sistem dinamis kecil: besaran yang tumbuh menurut satu aturan dan menyusut menurut aturan lain, dan perilaku jangka panjangnya dapat dihitung secara tepat hanya dengan rumus deret geometri. Siapa pun yang pernah bertemu relasi rekurens dalam mata pelajaran algoritma, atau metode iteratif dalam metode numerik, akan segera mengenali matematikanya; yang belum pernah pun akan bertemu dengannya di sini secara perlahan, selangkah demi selangkah.

Alasan kedua, PROFIT sungguh mudah disalahpahami. Kata “koin”, “token”, “*blockchain*”, dan “Rupiah” yang muncul dalam satu paragraf membuat banyak pembaca teringat pada spekulasi, harga yang naik, dan masuk lebih awal. Justru pembacaan seperti itulah yang ingin dicegah bab ini. PROFIT pada rilis September 2026 adalah koin utilitas di devnet Solana, jaringan uji publik yang tokennya tidak memiliki nilai moneter nyata; PROFIT tidak dapat dijual untuk Rupiah; dan whitepaper publiknya menyebut PROFIT “bukan sekuritas, bukan uang elektronik, bukan janji

imbal hasil" [1]. Kita akan kembali ke kalimat ini setiap kali diperlukan, bukan karena gugup, melainkan karena inilah fakta paling berguna dalam bab ini.

Karena itu, bab ini bergerak dalam urutan yang dibutuhkan pembaca yang cermat. Pertama kita mendefinisikan koin utilitas dan menempatkannya dalam lanskap hukum Indonesia (Subbab 12.2). Lalu kita mempelajari Solana secukupnya untuk memahami arti "token SPL di devnet dengan enam desimal tanpa *freeze authority*" (Subbab 12.3 dan 12.4), dan kita mendaftar fakta publik PROFIT (Subbab 12.5). Bagian tengah bab berisi matematika dan kode: emisi (Subbab 12.6), pembakaran (Subbab 12.7), pasokan sebagai deret geometri (Subbab 12.8), titik impas (Subbab 12.9), dan lantai oracle (Subbab 12.10). Bagian akhir kembali ke produk: apa yang dapat dilakukan pengguna dengan PROFIT beserta pagar pengaman setiap tindakan (Subbab 12.11), mengapa program referal hanya punya satu tingkat (Subbab 12.12), dan sebuah tabel yang memisahkan yang sudah berjalan dari yang masih direncanakan (Subbab 12.13).

12.2 Apa Itu Koin Utilitas, dan Apa yang Bukan

12.2.1 Kebanggaan yang hampa

Mulailah dari sebuah malam biasa di platform sosial mana pun. Seseorang menjawab pertanyaan orang asing dengan teliti, lengkap dengan sumber. Orang lain mengoreksi kekeliruan dalam kiriman yang sedang ramai, dengan sopan. Orang ketiga melaporkan tautan penipuan sebelum ada yang mengekliknya. Ketiganya telah berbuat sesuatu yang berguna bagi komunitas, dan ketiganya menerima imbalan yang sama: beberapa tanda suka, mungkin satu pengikut baru, dan rasa hangat kecil yang hilang esok paginya. Visi pendirian Profitina menyebut hal ini *kebanggaan yang hampa*: imbalan yang nyata tetapi tidak dapat disimpan, dibelanjakan, atau dijadikan pijakan.

PROFIT adalah upaya Profitina untuk memberi kontribusi semacam itu imbalan yang sedikit lebih berisi. Kiriman yang membantu, umpan balik berguna yang memperbaiki model lokal (Bab 7), kanal baru yang benar-benar dipakai orang lain, teman yang bergabung dan memverifikasi surelnya: masing-masing menghasilkan sejumlah kecil PROFIT, dan PROFIT dapat dipakai di dalam platform untuk layanan yang nyata. Slogan seluruh produk, *Bersama · Berkembang · Berbagi*, mengatakan hal yang sama dalam tiga kata.

Perhatikan apa yang *tidak* dikatakan cerita ini. Cerita ini tidak mengatakan bahwa PROFIT kelak akan bernilai tinggi, bahwa pengguna awal akan kaya, atau bahwa koin ini adalah cara berinvestasi di Profitina. Cerita ini mengatakan bahwa PROFIT adalah cara untuk mengakui dan memanfaatkan kontribusi. Itulah yang menjadikannya koin utilitas.

> **DEFINISI.** *Koin utilitas* (*utility coin* atau token utilitas) adalah satuan digital yang tujuannya untuk *dipakai* di dalam sebuah produk: membayar layanan, membuka fitur, atau kelak ikut serta dalam tata kelola. Pemegangnya adalah pengguna produk, bukan pemilik usaha, dan memegangnya tidak memberi hak atas laba, bunga, atau harga di masa depan.

12.2.2 Empat hal yang bukan PROFIT

Mendefinisikan PROFIT lewat apa yang bukan dirinya sangat membantu, karena setiap "bukan" menutup satu kesalahpahaman tertentu. Whitepaper publik menggunakan kata-kata ini, dan aplikasinya mengulanginya [1], [2].

- **Bukan sekuritas.** Sekuritas, dalam pengertian hukum sehari-hari, adalah investasi pada suatu usaha dengan harapan memperoleh keuntungan dari jerih payah orang lain: saham, obligasi,

unit reksa dana. PROFIT tidak memberi klaim atas pendapatan, aset, atau keputusan Profitina, dan tidak ditawarkan sebagai investasi.

- **Bukan uang elektronik.** Uang elektronik adalah nilai tersimpan dalam Rupiah yang dapat dipakai untuk membayar pihak ketiga, seperti saldo dompet elektronik berizin. PROFIT tidak berdenominasi Rupiah, tidak dapat dipakai untuk membayar siapa pun di luar platform, dan tidak dapat ditukar kembali menjadi Rupiah pada rilis September 2026.
- **Bukan janji imbal hasil.** Tidak ada apa pun pada PROFIT yang menjamin nilai acuannya akan naik, tetap, atau bahkan ada dalam bentuk tertentu di masa depan.
- **Bukan meme.** Koin meme ada semata untuk diperdagangkan berdasarkan perhatian. PROFIT ada karena platform membutuhkan satuan untuk memberi imbalan dan membayar layanan, dan aturannya justru dirancang untuk *memperlambat* penciptaan dan perdagangan cepat yang menjadi napas koin meme.

> **PERINGATAN.** Bab ini menggambarkan sebuah fitur produk dan mengajarkan matematika dengan angka ilustratif. Bab ini bukan nasihat keuangan, bukan tawaran untuk membeli atau menjual apa pun, dan bukan ramalan harga apa pun. PROFIT pada rilis September 2026 adalah token di devnet Solana tanpa nilai moneter nyata [1].

12.2.3 Kerangka Indonesia

Uang dan aset kripto adalah kegiatan yang diatur di Indonesia, dan platform yang bertanggung jawab berangkat dari hukum, bukan dari teknologi. Tiga fakta membingkai semua yang berikut.

Pertama, Rupiah adalah mata uang untuk pembayaran di Indonesia. Undang-Undang Mata Uang, UU No. 7 Tahun 2011, mewajibkan penggunaan Rupiah untuk pembayaran dan kewajiban lain yang diselesaikan dengan uang di dalam negeri [3]. Sebuah token tidak dapat secara sah menggantikan Rupiah sebagai alat pembayaran, dan PROFIT tidak mencobanya: setiap pembelian layanan nyata di Profitina dihargai dan dibayar dalam Rupiah melalui QRIS atau dalam PROFIT sebagai kredit produk, tidak pernah sebagai mata uang pengganti dalam perekonomian yang lebih luas.

Kedua, aset kripto diawasi oleh Otoritas Jasa Keuangan (OJK). Sejak 10 Januari 2025, berdasarkan Peraturan Pemerintah No. 49 Tahun 2024, pengawasan aset kripto berpindah dari Bappebti, regulator perdagangan berjangka komoditi, ke OJK, yang mengatur perdagangannya sebagai *aset keuangan digital* melalui Peraturan OJK No. 27 Tahun 2024 [4]. Karena itu, setiap langkah di masa depan yang memungkinkan PROFIT diperdagangkan untuk Rupiah berada di dalam kerangka OJK, sementara Bank Indonesia tetap menjadi regulator sistem pembayaran seperti QRIS.

Ketiga, urutan peristiwa itu penting. Peta jalan Profitina menempatkan mainnet dan segala bentuk bursa *setelah* audit token dan perizinan, bukan sebelumnya [1]. Itulah sebabnya penjualan PROFIT dinonaktifkan hari ini, dan itulah sebabnya bab ini terus menyebut “peta jalan” untuk segala hal yang menyangkut penukaran PROFIT kembali ke Rupiah.

> **WAWASAN KUNCI.** Hukum lebih dulu, rancangan mengikutinya. Menonaktifkan penjualan PROFIT sampai mainnet dan kejelasan regulasi bukanlah fitur yang hilang; itulah fitur yang menjaga koin utilitas tetap menjadi koin utilitas.

12.3 Solana dengan Kata-Kata Sederhana

12.3.1 Buku catatan bersama yang tak bisa diubah diam-diam

Bayangkan sebuah buku catatan yang mencatat siapa memiliki apa. Ribuan orang di seluruh dunia menyimpan salinan yang identik. Ketika seseorang ingin menambahkan satu baris ("pindahkan 5 satuan dari akun Ani ke akun Budi"), ia menandatangani permintaan itu dengan kunci rahasia yang hanya ia pegang, dan para penjaga buku memeriksa tanda tangannya, memeriksa bahwa Ani memang memiliki 5 satuan, lalu menambahkan baris itu ke salinan mereka dalam urutan yang sama. Begitu cukup banyak penjaga sepakat, baris itu final: untuk mengubahnya, seseorang harus membujuk sebagian besar penjaga untuk menulis ulang sejarah bersama-sama, hal yang memang dirancang agar tidak praktis dilakukan.

Buku catatan itu adalah *blockchain*, para penjaganya adalah *validator*, permintaan bertanda tangan itu adalah *transaksi*, dan protokol penjaga urutannya adalah *konsensus*. Solana adalah salah satu *blockchain* publik semacam itu, dirancang untuk memproses banyak transaksi dengan cepat dan murah. Bagi koin imbalan yang dibagikan dalam jumlah seperti 0,01 per kali, murah dan cepat itu penting: biaya beberapa ribu Rupiah per transfer akan membuat imbalan kecil menjadi tidak masuk akal.

12.3.2 Akun, program, transaksi

Solana menata keadaannya sebagai *akun*. Akun adalah catatan kecil yang dikenali oleh alamat publik 32 bita, ditulis dalam bentuk teks bernama base58 (untaian panjang huruf dan angka yang terlihat di penjelajah blok). Setiap akun menyimpan saldo mata uang asli jaringan, SOL, dan dapat menyimpan data. Sebagian akun menyimpan *program*: kode yang berjalan di rantai. Program di Solana tidak menyimpan keadaan; program membaca dan menulis data yang disimpan di akun *lain* yang diserahkan kepadanya.

Transaksi adalah daftar instruksi, masing-masing ditujukan kepada sebuah program, bersama tanda tangan akun-akun yang menyetujuinya. Ketika Token Program (Subbab 12.4) menerima instruksi "transfer 250.000 satuan dasar dari akun A ke akun B", program itu memeriksa bahwa pemilik A menandatangani, bahwa A memiliki cukup saldo, lalu memperbarui kedua akun.

Biaya dibayar dalam *lamport*, satuan terkecil SOL; satu SOL sama dengan 10^9 lamport. Menurut dokumentasi resmi Solana, biaya dasar adalah 5.000 lamport per tanda tangan, dan biaya itu dibagi: **50% dibakar dan 50% diberikan kepada validator**; *biaya prioritas* yang bersifat opsional, yang ditambahkan pengirim agar didahulukan pada saat sibuk, seluruhnya diberikan kepada validator [5]. Paruh pertama aturan itu layak direnungkan sejenak. Jaringan tempat PROFIT hidup sudah memusnahkan sebagian dari setiap biaya yang dipungutnya. Pembakaran bukan gagasan eksotis yang diciptakan untuk PROFIT; pembakaran adalah bagian dari pipa-pipa rantai itu sendiri, dan Subbab 12.7 akan bertemu lagi dengannya.

12.3.3 Tiga klaster, satu untuk berlatih

Solana berjalan sebagai beberapa jaringan publik terpisah, disebut *klaster*, dengan perangkat lunak yang sama dan tujuan yang berbeda. **Tabel 12.1** merangkumnya dengan kata-kata dokumentasi resmi [6].

Tabel 12.1 Tiga klaster publik Solana, sebagaimana dijelaskan dalam dokumentasi resmi pada September 2026.

Klaster	Tujuan	Apakah tokennya nyata?	Titik akhir RPC publik
Mainnet	Klaster permanen dan tanpa izin bagi pengguna, pembangun, validator, dan pemegang token	Ya: SOL yang diterbitkan di mainnet itu nyata	`https://api.mainnet.solana.com`
Devnet	"Taman bermain bagi siapa saja yang ingin mencoba Solana, sebagai pengguna, pemegang token, pengembang aplikasi, atau validator"	Tidak; keran (*faucet*) membagikan SOL uji	`https://api.devnet.solana.com`
Testnet	Tempat kontributor inti menguji rilis baru di bawah beban, berfokus pada kinerja dan perilaku validator	Tidak; keran membagikan SOL uji	`https://api.testnet.solana.com`

Kontrak PROFIT berada di **devnet**. Satu fakta ini menjelaskan sebagian besar hal berikutnya: kontraknya nyata dan publik, siapa pun dapat memeriksanya, dan tidak memiliki nilai moneter, karena memang tidak ada apa pun di devnet yang bernilai. Spanduk merek Profitina dan string versinya memakai kata "testnet" dalam arti sehari-hari "belum mainnet"; nama tepat klasternya adalah devnet, dan buku ini memakai nama yang tepat.

12.3.4 Keadaan jaringan pada 2026

Pembaca yang masih ingat berita gangguan Solana pada 2021 dan 2022 layak mendapat kabar terbaru. Per Agustus 2026, mainnet Solana telah berjalan **30 bulan berturut-turut, sekitar 913 hari, tanpa gangguan menyeluruh**; penghentian total terakhir terjadi pada 6 Februari 2024 dan dipulihkan dalam waktu kurang dari lima jam [7]. Laporan yang sama menyebut tiga perubahan sebagai penyebabnya: protokol jaringan QUIC dengan kualitas layanan berbobot *stake*, pasar biaya yang kini menjadikan biaya prioritas sekitar 88% dari pendapatan biaya harian, dan keragaman klien, khususnya Firedancer, klien validator independen yang mulai berjalan di mainnet pada akhir 2025 [7]. Memiliki lebih dari satu implementasi validator yang independen berarti satu *bug* saja lebih kecil kemungkinannya menghentikan seluruh jaringan.

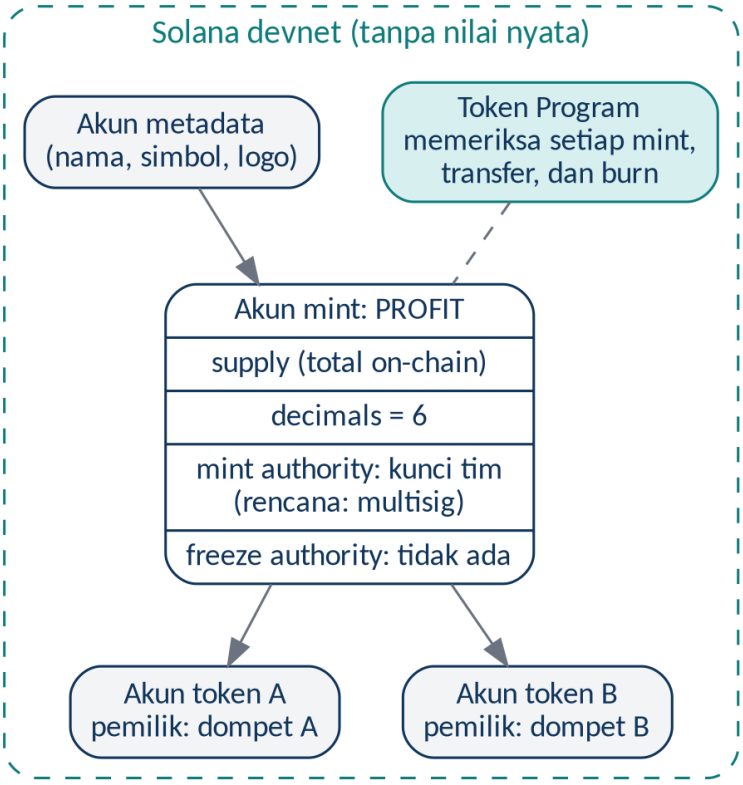
Jaringan itu juga terus berubah. Catatan perubahan resmi tanggal 18 September 2026 mencantumkan, di antara *feature gate* untuk mainnet, *Transaction V1* (format transaksi baru yang, antara lain, membuat penggelaran program sekitar empat kali lebih murah dalam biaya) dan pengurangan *slot time* menjadi 250 milidetik; *slot* adalah jendela singkat ketika satu validator, yaitu pemimpin saat itu, boleh menghasilkan satu blok [8]. Catatan perubahan yang sama memperlihatkan rilis klien Agave dan Firedancer yang bergerak seiring [8]. Versi dan waktu seperti ini menua dalam hitungan minggu; pembaca sebaiknya memperlakukannya sebagai potret September 2026 dan memeriksa catatan perubahan terkini.

> **WAWASAN KUNCI.** Bagi PROFIT, sifat Solana yang paling penting justru yang membosankan: biaya yang sangat kecil dibandingkan imbalan 0,01, devnet terpisah untuk pengujian yang jujur, dan standar token yang sudah dipahami setiap dompet dan penjelajah blok. Rekor kecepatan memang menyenangkan; keandalan yang membosankan itulah yang dibutuhkan sistem imbalan.

12.4 Token di Solana: Mint, Akun, dan Otoritas

12.4.1 Mint dan akun token

Di Solana, token seperti PROFIT bukanlah program tersendiri. Token adalah data yang dikelola oleh sebuah program bersama yang dipakai luas, *Token Program*, yang sudah tahu cara membuat token, memindahkannya, dan membakarnya. Dua jenis akun menyimpan data itu, dan **Gambar 12.1** memperlihatkan hubungannya.



Gambar 12.1 Sebuah token di Solana: satu akun mint menggambarkan token (supply, decimals, dan dua otoritas), setiap pemegang memiliki akun token yang menunjuk ke mint, dan akun metadata memberinya nama dan logo. Token Program menegakkan aturan untuk setiap perubahan.

Akun mint menggambarkan token secara keseluruhan. Dokumentasi Solana mendaftar kolom-kolomnya: *supply* (jumlah total satuan yang ada), *decimals* (berapa tempat desimal yang dimiliki satu satuan), *mint authority* ("akun yang berwenang menciptakan satuan baru token, sehingga menambah pasokan") dan *freeze authority* ("akun yang berwenang membekukan token di sebuah akun token, sehingga token itu tidak dapat ditransfer atau dibakar") [9].

Akun token mencatat berapa satuan dari satu mint tertentu yang dimiliki oleh satu pemilik tertentu. Dompet yang memegang tiga token berbeda memiliki tiga akun token. Akun token yang lazim untuk pasangan dompet dan mint tertentu adalah *associated token account*, yang alamatnya diturunkan dari alamat pemilik dan alamat mint, sehingga siapa pun dapat menghitung di mana PROFIT milik sebuah dompet berada tanpa perlu bertanya [9].

Akun ketiga, dari program *Token Metadata* milik Metaplex, melekatkan nama, simbol, dan gambar pada mint, sehingga dompet dapat menampilkan "PROFIT" dengan logo Profitina alih-alih alamat mentah [10].

12.4.2 Desimal, dan mengapa jumlah selalu bilangan bulat

Kolom yang paling mengejutkan pemula adalah *decimals*. Di rantai, setiap jumlah adalah bilangan bulat *satuan dasar* (*base unit*). Kolom desimal hanya menyatakan cara menampilkan bilangan itu kepada manusia. Jika sebuah mint memiliki d desimal dan sebuah akun token memegang n satuan dasar, jumlah yang dilihat orang adalah

$$a = \frac{n}{10^{d}} \tag{12.1}$$

PROFIT memiliki $d = 6$, sehingga satu PROFIT sama dengan $10^{6} = 1{,}000{,}000$ satuan dasar, dan jumlah terkecil yang mungkin adalah 0.000001 PROFIT, yang oleh whitepaper disebut satu µP (mikro-PROFIT) [1]. Imbalan referal 0,25 PROFIT disimpan sebagai bilangan bulat 250.000; imbalan dasar 0,01 PROFIT adalah 10.000.

Mengapa repot begini alih-alih menyimpan 0,25 secara langsung? Karena komputer menyimpan kebanyakan bilangan pecahan dalam format titik mengambang (*floating point*) biner, dan biner tidak dapat menyatakan 0,01 atau 0,1 secara tepat, sama seperti ekspansi desimal tidak dapat menyatakan 1/3 secara tepat. Galatnya kecil untuk satu bilangan, tetapi sebuah buku besar (*ledger*) menjumlahkan jutaan bilangan, dan galat kecil itu menumpuk. Langkah pertama Lab G8 memperlihatkannya. **Kode 12.1** mengurai string desimal menjadi mikro-PROFIT bulat, memformatnya kembali, menolak jumlah dengan tujuh desimal, lalu menjumlahkan 0,01 sebanyak seratus kali dengan dua cara.

Kode 12.1 `01_units.py`: jumlah PROFIT sebagai bilangan bulat mikro-PROFIT, dengan Persamaan (12.1) diterapkan ke dua arah.

```
"""Lab G8, step 1: keep PROFIT amounts as whole numbers of micro-PROFIT.

A token with 6 decimals stores every amount as an integer count of its
smallest unit. We do the same, and we show why a float is the wrong tool.
"""

DECIMALS = 6
SCALE = 10 ** DECIMALS            # 1 PROFIT = 1,000,000 micro-PROFIT


def to_micro(text):
    """Parse a decimal string such as '0.25' into micro-PROFIT, exactly."""
    whole, _, frac = text.strip().partition(".")
    if len(frac) > DECIMALS:
        raise ValueError(f"{text!r} has more than {DECIMALS} decimals")
    if not (whole or "0").isdigit() or not (frac or "0").isdigit():
        raise ValueError(f"{text!r} is not a plain decimal number")
    return int(whole or "0") * SCALE + int(frac.ljust(DECIMALS, "0") or "0")


def to_text(micro):
    """Format micro-PROFIT back as a decimal string with 6 places."""
    return f"{micro // SCALE}.{micro % SCALE:06d}"


def main():
    print("1 PROFIT =", SCALE, "micro-PROFIT")
    for text in ["0.25", "0.01", "50", "0.000001"]:
        micro = to_micro(text)
        print(f"{text:>9} PROFIT -> {micro:>9} micro -> {to_text(micro)}")

    try:
        to_micro("0.0000001")
    except ValueError as err:
        print("rejected:", err)
```

Kode 12.1 (Lanjutan) *01_units.py: jumlah PROFIT sebagai bilangan bulat mikro-PROFIT, dengan Persamaan (12.1) diterapkan ke dua arah.*

```
    # One hundred rewards of 0.01 PROFIT, added two ways.
    as_float = 0.0
    as_micro = 0
    for _ in range(100):
        as_float += 0.01
        as_micro += to_micro("0.01")
    print("float sum of 100 x 0.01 =", as_float)
    print("micro sum of 100 x 0.01 =", to_text(as_micro))
    print("float sum == 1.0 ?", as_float == 1.0)
    print("micro sum == 1 PROFIT ?", as_micro == SCALE)


if __name__ == "__main__":
    main()
```

Versi Java, **Kode 12.2**, melakukan hal yang sama dengan bilangan bulat `long` dan mencetak teks yang persis sama.

Kode 12.2 `Units.java`: konversi yang sama dalam Java, dengan hitungan mikro-PROFIT bertipe `long`.

```
/** Lab G8, step 1: keep PROFIT amounts as whole numbers of micro-PROFIT. */
public class Units {
    static final int DECIMALS = 6;
    static final long SCALE = 1_000_000L;          // 1 PROFIT = 10^6 micro

    /** Parse a decimal string such as "0.25" into micro-PROFIT, exactly. */
    static long toMicro(String text) {
        String t = text.strip();
        int dot = t.indexOf('.');
        String whole = dot < 0 ? t : t.substring(0, dot);
        String frac = dot < 0 ? "" : t.substring(dot + 1);
        if (frac.length() > DECIMALS) {
            throw new IllegalArgumentException(
                "'" + text + "' has more than " + DECIMALS + " decimals");
        }
        if (!(whole.isEmpty() ? "0" : whole).matches("\\d+")
                || !(frac.isEmpty() ? "0" : frac).matches("\\d+")) {
            throw new IllegalArgumentException(
                "'" + text + "' is not a plain decimal number");
        }
        String padded = (frac + "000000").substring(0, DECIMALS);
        long w = whole.isEmpty() ? 0 : Long.parseLong(whole);
        return w * SCALE + Long.parseLong(padded);
    }

    /** Format micro-PROFIT back as a decimal string with 6 places. */
    static String toText(long micro) {
        return (micro / SCALE) + "." + String.format("%06d", micro % SCALE);
    }

    public static void main(String[] args) {
        System.out.println("1 PROFIT = " + SCALE + " micro-PROFIT");
        for (String text : new String[] {"0.25", "0.01", "50", "0.000001"}) {
            long micro = toMicro(text);
            System.out.printf("%9s PROFIT -> %9d micro -> %s%n",
                              text, micro, toText(micro));
        }
        try {
            toMicro("0.0000001");
        } catch (IllegalArgumentException err) {
            System.out.println("rejected: " + err.getMessage());
        }
        // One hundred rewards of 0.01 PROFIT, added two ways.
        double asFloat = 0.0;
        long asMicro = 0;
        for (int i = 0; i < 100; i++) {
```

Kode 12.2 (Lanjutan) *Units.java: konversi yang sama dalam Java, dengan hitungan mikro-PROFIT bertipe long.*

```
            asFloat += 0.01;
            asMicro += toMicro("0.01");
        }
        System.out.println("float sum of 100 x 0.01 = " + asFloat);
        System.out.println("micro sum of 100 x 0.01 = " + toText(asMicro));
        System.out.println("float sum == 1.0 ? "
                           + (asFloat == 1.0 ? "True" : "False"));
        System.out.println("micro sum == 1 PROFIT ? "
                           + (asMicro == SCALE ? "True" : "False"));
    }
}
```

Kedua program mencetak:

```
1 PROFIT = 1000000 micro-PROFIT
     0.25 PROFIT ->    250000 micro -> 0.250000
     0.01 PROFIT ->     10000 micro -> 0.010000
       50 PROFIT ->  50000000 micro -> 50.000000
 0.000001 PROFIT ->         1 micro -> 0.000001
rejected: '0.0000001' has more than 6 decimals
float sum of 100 x 0.01 = 1.0000000000000007
micro sum of 100 x 0.01 = 1.000000
float sum == 1.0 ? False
micro sum == 1 PROFIT ? True
```

Empat baris terakhir adalah pelajarannya. Dijumlahkan seratus kali sebagai bilangan titik mengambang biner, 0,01 tidak menghasilkan 1,0 melainkan 1.0000000000000007, dan perbandingannya dengan 1,0 bernilai salah. Dijumlahkan sebagai bilangan bulat 10.000, hasilnya tepat 1.000.000, yaitu tepat satu PROFIT. Standar IEEE 754 yang diikuti hampir semua prosesor mendefinisikan perilaku ini secara persis [11]; ini bukan *bug* Python atau Java, melainkan sifat pecahan biner. Buku besar yang tidak boleh kehilangan atau menciptakan satu mikro-PROFIT pun menghitung dalam satuan dasar bulat, persis seperti rantai.

12.4.3 Dua otoritas

Dua otoritas di akun mint adalah tempat kepercayaan bersemayam, dan keduanya layak dibaca perlahan.

Mint authority dapat menciptakan satuan baru. Siapa pun yang memegang kuncinya dapat menambah pasokan. Untuk koin imbalan, suatu *mint authority* harus ada selama imbalan masih diterbitkan; pertanyaannya adalah siapa yang memegangnya dan dengan kendali apa. Satu kunci yang dipegang satu orang adalah titik kegagalan tunggal: jika hilang, tidak ada lagi imbalan yang dapat diterbitkan; jika bocor, siapa pun dapat mencetak koin. Jawaban baku di Solana adalah *multisig*, akun yang mensyaratkan tanda tangan sejumlah kunci dari satu himpunan kunci sebelum bertindak; Squads adalah platform multisig yang dipakai luas untuk mengelola kas dan token di Solana [12]. Rencana yang dinyatakan Profitina adalah memindahkan *mint authority* PROFIT ke sebuah multisig. *Mint authority* juga dapat *dicabut* sepenuhnya, diatur menjadi tidak ada, sehingga tidak ada satuan baru yang dapat diciptakan lagi; itulah keadaan akhir yang wajar bagi token yang pasokan maksimumnya telah terdistribusi seluruhnya.

Freeze authority dapat membekukan akun token seorang pemegang sehingga tokennya tidak dapat dipindahkan atau dibakar [9]. Kewenangan ini memiliki kegunaan yang sah (aset yang diatur mungkin diwajibkan membekukan akun atas perintah pengadilan), tetapi juga merupakan kekuasaan untuk mengurung milik orang sesuka hati. *Freeze authority* dapat diatur menjadi tidak ada ketika mint dibuat, dan begitu tidak ada, tidak seorang pun dapat membekukan akun PROFIT: bukan penyerang, bukan pengadilan melalui tim, bukan pula tim itu sendiri.

Freeze authority PROFIT adalah tidak ada [1]. Ini bukan janji tentang niat; ini adalah sifat mint yang telah digelar, yang dapat diperiksa siapa pun di penjelajah blok dalam beberapa detik. Dari semua fakta di bab ini, inilah yang paling langsung melindungi pemegang, dan biayanya hanyalah keputusan untuk melepaskan sebuah kekuasaan.

> **CONTOH.** Untuk memeriksa pengaturan sebuah token sendiri, buka Solana Explorer, ubah pemilih klaster ke *Devnet*, lalu tempelkan alamat mint dari **Tabel 12.2** ke kotak pencarian. Halaman itu menampilkan pasokan saat ini, desimal, *mint authority*, dan *freeze authority*. Untuk PROFIT, kolom *freeze authority* menyatakan tidak ada. Melakukannya sekali lebih berharga daripada membaca paragraf ini dua kali.

12.5 Fakta Publik PROFIT

Kini kita dapat membaca deskripsi publik PROFIT dengan pemahaman. **Tabel 12.2** menghimpun apa yang dinyatakan whitepaper publik, versi 24 bulan September 2026, tentang koin ini, bersama apa yang ditampilkan aplikasi [1], [2].

Tabel 12.2 Fakta publik PROFIT, dari whitepaper publik v24 dan aplikasi, per September 2026.

Sifat	Nilai pada rilis September 2026
Hakikat	Koin utilitas: bahan bakar layanan AI dan akses fitur, kelak tata kelola; "bukan meme, bukan sekuritas"
Rantai dan standar	Solana, token SPL, metadata Metaplex; logonya adalah logo Profitina
Klaster	Devnet; kontraknya berjalan dan dapat diverifikasi publik
Alamat mint devnet	`6f1dNQPhYXPo78LVLGD2WpzBH2mzCm77BG9E8PFnWweL`
Desimal	6 (1 PROFIT = 1.000.000 µP)
Freeze authority	Tidak ada
Mint authority	Direncanakan pindah ke multisig
Pasokan	Pasokan maksimum yang terkunci
Emisi	Menurun: pengali imbalan menjadi separuh setiap kali jumlah pengguna manusia tumbuh sepuluh kali lipat
Pembakaran	*Spend-burn*, *transfer-burn*, *buy-and-burn* dari pendapatan QRIS, pembakaran penalti anti-bot
Oracle	Lantai = cadangan Rupiah ÷ PROFIT beredar; acuan awal 1 PROFIT = Rp1.000, estimasi model, bukan harga pasar
Pembelian	Melalui QRIS, Rp50.000 sampai Rp1.000.000, dikreditkan pada harga oracle setelah administrator memverifikasi pembayaran
Penjualan atau penukaran ke Rupiah	Dinonaktifkan sampai mainnet dan kejelasan regulasi
Mainnet	Peta jalan: Fase 3, berlabel K4 2026, setelah audit token dan perizinan

Tiga baris dalam tabel itu masing-masing layak mendapat satu kalimat sebelum kita melanjutkan.

Baris **klaster** adalah fondasi segala hal lain: kontraknya nyata, tetapi token devnet tidak bernilai [6], sehingga tidak ada isi tabel yang menggambarkan uang.

Baris **pasokan** menyatakan bahwa pasokan maksimum terkunci. Batas atas yang tetap berarti emisi, bagaimanapun jadwalnya, tidak dapat berlangsung selamanya; jadwal hanya menentukan seberapa cepat batas itu didekati, dan pembakaran menentukan berapa banyak dari yang telah diemisikan masih beredar.

Baris **oracle** memakai kata "lantai" (*floor*) dan frasa "estimasi model" di samping angka Rupiah. Subbab 12.10 menjelaskan rumus lantai dan memperlihatkan, dengan bukti singkat, mengapa pembelian pada harga lantai tidak menggesernya. Angka Rupiah adalah acuan yang dipakai platform untuk menghargai layanannya sendiri, bukan kuotasi tempat siapa pun dapat berdagang.

> **PERINGATAN.** Alamat mint dalam **Tabel 12.2** adalah alamat devnet. Token bernama sama di mainnet, atau "PROFIT" apa pun yang ditawarkan untuk dijual di mana pun pada rilis September 2026, bukanlah koin Profitina. Satu-satunya kontak resmi untuk pertanyaan adalah info@profitina.com.

12.6 Emisi yang Menurun

12.6.1 Masalah yang diselesaikan jadwal menurun

Andaikan sebuah platform membayar setiap pengguna aktif tetap 1 PROFIT per hari, selamanya. Dengan 1.000 pengguna, platform menciptakan 1.000 PROFIT sehari; dengan sejuta pengguna, sejuta sehari. Jumlah yang diciptakan tumbuh tepat sebanding dengan komunitas, dan dengan pasokan maksimum yang terkunci, batas atas itu akan tercapai dengan tergesa-gesa, tanpa menyisakan apa pun untuk menghargai orang-orang yang datang belakangan. Lebih buruk lagi, imbalan tetap per pengguna mengundang pengguna palsu: setiap akun tambahan, asli atau bukan, menambah jumlah yang sama.

Jadwal yang *menurun* menjawab kedua masalah itu. Jadwal ini membayar dengan murah hati selagi komunitas masih kecil dan platform paling membutuhkan kontributor pertamanya, lalu membayar setiap orang lebih sedikit seiring tumbuhnya komunitas, sehingga total yang diciptakan per hari tetap tumbuh, tetapi jauh lebih lambat daripada jumlah orangnya.

> **DEFINISI.** *Emisi* adalah penciptaan satuan baru sebuah token, di sini sebagai imbalan atas kontribusi. *Pengali emisi* adalah bilangan antara 0 dan 1 yang mengalikan setiap imbalan dasar; jadwal emisi yang *menurun* membuat pengali itu mengecil ketika suatu ukuran platform, di sini jumlah pengguna manusia, bertambah.

12.6.2 Aturan publiknya

Whitepaper publik menyatakan aturannya dengan kata-kata: pengali **menjadi separuh setiap kali jumlah pengguna manusia tumbuh sepuluh kali lipat** [1]. Whitepaper menggambarkan aturan itu dengan sebuah tabel kecil, yang kita reproduksi sebagai **Tabel 12.3**.

Tabel 12.3 Tabel emisi ilustratif dari whitepaper: pengali menjadi separuh untuk setiap pertumbuhan pengguna sepuluh kali lipat.

Pengguna manusia	Pengali	PROFIT per hari pada 1 PROFIT per pengguna per hari
1.000	0,125	125
100.000	0,03125	3.125
1.000.000	0,015625	15.625

Untuk mengubah kata-kata itu menjadi rumus, misalkan u adalah jumlah pengguna manusia dan $m(u)$ adalah pengalinya. “Menjadi separuh untuk setiap pertumbuhan sepuluh kali lipat” berarti mengalikan u dengan 10 mengalikan m dengan $1/2$. Banyaknya pertumbuhan sepuluh kali lipat antara ukuran acuan u_0 dan u adalah $\log_{10}(u/u_0)$, karena $10^{\log_{10}(u/u_0)} = u/u_0$. Jadi pengali telah dibagi dua sebanyak $\log_{10}(u/u_0)$ kali:

$$m(u) = (1/2)^{\log_{10}(u/u_0)} \tag{12.2}$$

Acuan u_0 mana yang mereproduksi tabel? Pada $u = 1{,}000$ pengalinya $0.125 = (1/2)^3$, sehingga $\log_{10}(1000/u_0) = 3$, yang memberi $u_0 = 1$. Periksa baris lainnya: $\log_{10} 100{,}000 = 5$ dan $(1/2)^5 = 0.03125$; $\log_{10} 1{,}000{,}000 = 6$ dan $(1/2)^6 = 0.015625$. Ketiga baris cocok. Mulai sekarang kita ambil $u_0 = 1$ dan menulis $m(u) = (1/2)^{\log_{10} u}$.

Jika setiap pengguna memperoleh imbalan dasar r PROFIT per hari sebelum pengali, total yang diciptakan per hari adalah

$$E(u) = r \cdot u \cdot m(u) \tag{12.3}$$

dan dengan $r = 1$ seperti di tabel: $E(1{,}000) = 1{,}000 \times 0.125 = 125$, $E(100{,}000) = 100{,}000 \times 0.03125 = 3{,}125$, $E(1{,}000{,}000) = 1{,}000{,}000 \times 0.015625 = 15{,}625$, persis kolom paling kanan.

Semua ini adalah ilustrasi aturan, bukan ramalan emisi harian Profitina yang sebenarnya. Imbalan nyata adalah jumlah kecil untuk tindakan tertentu (Subbab 12.6.5), dan aturan whitepaper adalah satu-satunya bagian mesin emisi yang dipakai buku ini.

12.6.3 Hukum pangkat yang tersembunyi

Persamaan (12.3) menyimpan kejutan yang tampak setelah kita memakai satu identitas logaritma. Untuk bilangan positif a dan u,

$$a^{\log_{10} u} = u^{\log_{10} a} \tag{12.4}$$

Buktinya dua baris. Ambil $\log_{10}$ ruas kiri: $\log_{10}(a^{\log_{10} u}) = \log_{10} u \cdot \log_{10} a$. Ambil $\log_{10}$ ruas kanan: $\log_{10}(u^{\log_{10} a}) = \log_{10} a \cdot \log_{10} u$. Kedua logaritma itu adalah hasil kali yang sama, dan $\log_{10}$ bersifat satu-satu, sehingga kedua ruas sama.

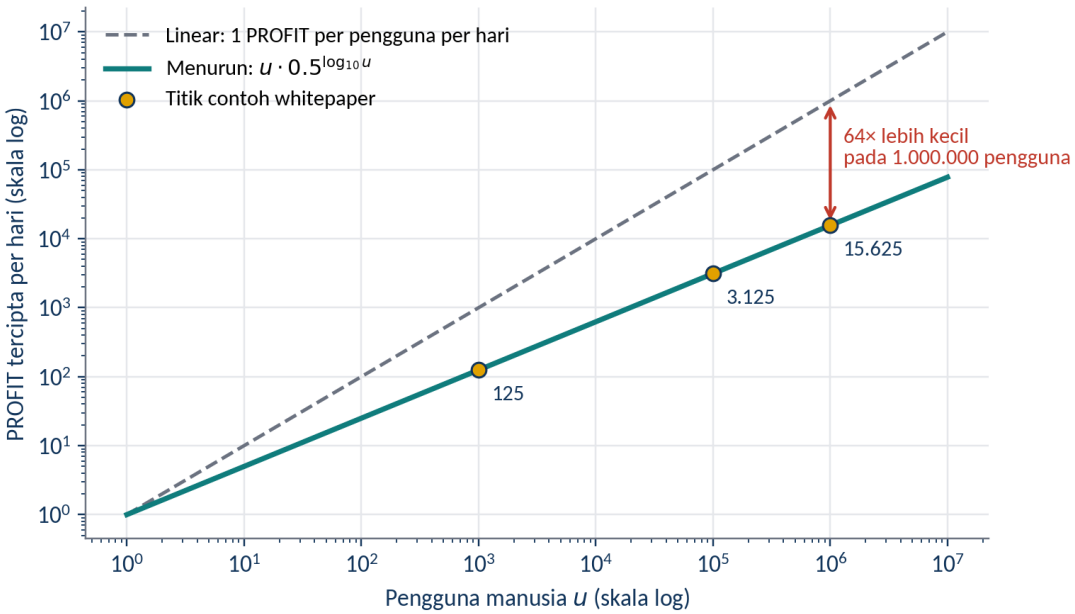
Terapkan Persamaan (12.4) dengan $a = 1/2$. Karena $\log_{10} 1/2 = -\log_{10} 2$, kita memperoleh $m(u) = u^{-\log_{10} 2}$, dan karena itu

$$E(u) = r \cdot u^{1-\log_{10} 2} = r \cdot u^{\alpha}, \qquad \alpha = 1 - \log_{10} 2 \approx 0.69897 \tag{12.5}$$

Dengan kata lain: emisi harian adalah sebuah *pangkat* dari jumlah pengguna, dengan eksponen sekitar 0,7. Emisi tetap tumbuh seiring tumbuhnya komunitas, tetapi secara sublinear. Tiga akibat langsung menyusul.

- **Pengguna sepuluh kali, emisi lima kali.** Mengalikan u dengan 10 mengalikan E dengan $10^{\alpha} = 10 \cdot 10^{-\log_{10} 2} = 10/2 = 5$. Setiap dekade pertumbuhan menciptakan lima kali lebih banyak per hari, bukan sepuluh kali.
- **Selisih dengan emisi linear makin lebar.** Rasio emisi linear u terhadap emisi menurun u^{α} adalah $u^{1-\alpha} = u^{\log_{10} 2} = 2^{\log_{10} u}$. Pada sejuta pengguna, rasio itu $2^6 = 64$: jadwal menurun menciptakan **64 kali lebih sedikit** per hari daripada jadwal linear, angka yang dikutip whitepaper [1].
- **Imbalan per pengguna turun perlahan.** Bagian setiap pengguna adalah $m(u) = u^{-0.301}$. Menggandakan komunitas menurunkan pengali setiap orang dengan faktor $2^{-0.301} \approx 0.81$, sekitar 19%. Kontributor awal mendapat lebih banyak, tetapi yang datang belakangan tidak tertutup peluangnya.

Gambar 12.2 menggambar kedua jadwal pada sumbu logaritmik, tempat hukum pangkat menjadi garis lurus yang kemiringannya sama dengan eksponennya: kemiringan 1 untuk jadwal linear, kemiringan $\alpha \approx 0.7$ untuk jadwal menurun.



Gambar 12.2 Emisi harian menurut jadwal linear dan menurut jadwal menurun dari whitepaper, pada sumbu logaritmik. Tiga titik contoh whitepaper terletak pada garis menurun; pada sejuta pengguna selisihnya 64 kali lipat.

12.6.4 Lab G8, langkah 2: mereproduksi tabel

Program kedua di lab menghitung Persamaan (12.2), Persamaan (12.3), dan Persamaan (12.5) untuk setiap pangkat sepuluh dari 1 hingga 10 juta pengguna, serta mencetak jadwal linear dan rasionya di sampingnya. **Kode 12.3** adalah versi Python.

Kode 12.3 `02_emission.py`: pengali menurun dari Persamaan (12.2), emisi harian dari Persamaan (12.3), dan bentuk hukum pangkatnya Persamaan (12.5).

```
"""Lab G8, step 2: a decaying emission multiplier (illustrative, not financial).

Public rule used here: the multiplier halves every time the number of human
users grows tenfold, m(u) = 0.5 ** log10(u / U0). With U0 = 1 and a base of
1 PROFIT per user per day, this reproduces the whitepaper's example table.
"""
import math

U0 = 1              # reference user count (the reader may change it)
BASE = 1.0          # PROFIT per user per day before the multiplier


def multiplier(users):
    return 0.5 ** math.log10(users / U0)


def emission(users):
    """PROFIT created per day when every user earns the base reward."""
    return users * BASE * multiplier(users)


def main():
    alpha = 1 - math.log10(2)
    print(f"exponent alpha = 1 - log10(2) = {alpha:.6f}")
    print(f"{'users':>10} {'multiplier':>11} {'decaying/day':>13}"
          f" {'linear/day':>11} {'ratio':>6} {'u^alpha':>11}")
    for power in range(0, 8):
        u = 10 ** power
        m = multiplier(u)
        e = emission(u)
        linear = u * BASE
        print(f"{u:>10,} {m:>11.7f} {e:>13,.3f} {linear:>11,.0f}"
              f" {linear / e:>6.0f} {u ** alpha:>11,.3f}")
    # Growth of the daily emission per tenfold growth of users.
    print(f"emission grows x{emission(10**6) / emission(10**5):.3f}"
          " when users grow x10")


if __name__ == "__main__":
    main()
```

Kode 12.4 adalah versi Java. Satu-satunya kehati-hatian yang diperlukan adalah menetapkan *locale*, agar pemisah ribuan berupa koma di setiap mesin; tanpa itu, Windows berbahasa Indonesia akan mencetak 1.000.000 dan keluaran kedua bahasa akan berbeda.

Kode 12.4 `Emission.java`: tabel yang sama dalam Java, dengan format yang identik.

```
import java.util.Locale;

/** Lab G8, step 2: a decaying emission multiplier (illustrative, not financial). */
public class Emission {
    static final double U0 = 1;         // reference user count
    static final double BASE = 1.0;     // PROFIT per user per day

    static double multiplier(double users) {
        return Math.pow(0.5, Math.log10(users / U0));
    }

    /** PROFIT created per day when every user earns the base reward. */
    static double emission(double users) {
        return users * BASE * multiplier(users);
    }

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        double alpha = 1 - Math.log10(2);
        System.out.printf("exponent alpha = 1 - log10(2) = %.6f%n", alpha);
        System.out.printf("%10s %11s %13s %11s %6s %11s%n", "users",
            "multiplier", "decaying/day", "linear/day", "ratio", "u^alpha");
        for (int power = 0; power < 8; power++) {
            long u = (long) Math.pow(10, power);
            double m = multiplier(u);
            double e = emission(u);
            double linear = u * BASE;
            System.out.printf("%,10d %11.7f %,13.3f %,11.0f %6.0f %,11.3f%n",
                u, m, e, linear, linear / e, Math.pow(u, alpha));
        }
        // Growth of the daily emission per tenfold growth of users.
        System.out.printf("emission grows x%.3f when users grow x10%n",
            emission(1e6) / emission(1e5));
    }
}
```

Keduanya mencetak:

```
exponent alpha = 1 - log10(2) = 0.698970
     users  multiplier  decaying/day  linear/day  ratio     u^alpha
         1   1.0000000         1.000           1      1       1.000
        10   0.5000000         5.000          10      2       5.000
       100   0.2500000        25.000         100      4      25.000
     1,000   0.1250000       125.000       1,000      8     125.000
    10,000   0.0625000       625.000      10,000     16     625.000
   100,000   0.0312500     3,125.000     100,000     32   3,125.000
 1,000,000   0.0156250    15,625.000   1,000,000     64  15,625.000
10,000,000   0.0078125    78,125.000  10,000,000    128  78,125.000
emission grows x5.000 when users grow x10
```

Bacalah keluaran itu kolom demi kolom. Pengali menjadi separuh di setiap baris, sesuai aturan. Baris untuk 1.000, 100.000, dan 1.000.000 pengguna mereproduksi **Tabel 12.3** dengan tepat. Kolom rasio menjadi dua kali lipat di setiap baris dan mencapai 64 pada sejuta. Kolom terakhir, yang dihitung dengan pangkat u^{α} alih-alih pengali, sama dengan kolom emisi menurun hingga digit terakhir yang dicetak; inilah Persamaan (12.5) yang terkonfirmasi secara numerik. Baris penutup mengonfirmasi aturan "pengguna sepuluh kali, emisi lima kali".

Pengali dicetak dengan tujuh desimal, bukan enam, karena alasan yang patut diketahui. Pada sepuluh juta pengguna nilainya tepat 0.0078125, dan membulatkannya ke enam desimal adalah kasus seri: Python membulatkan seri ke angka genap (0.007812), sedangkan pemformat Java membulatkannya ke atas (0.007813). Mencetak tujuh desimal menghindari seri itu dan menjaga

kedua keluaran identik. Detail kecil seperti inilah alasan keluaran kedua bahasa di lab selalu diperiksa bita demi bita.

12.6.5 Dari jadwal ke imbalan nyata

Di dalam produk, tidak ada orang yang menerima "1 PROFIT per hari". Orang menerima jumlah kecil untuk tindakan tertentu yang berguna, masing-masing dikalikan pengali emisi dan skor kebergunaan dari AI lokal, dan masing-masing dibatasi. **Tabel 12.4** mendaftar tindakan perolehan dan batasnya sebagaimana dinyatakan whitepaper publik [1].

Tabel 12.4 Tindakan perolehan, imbalan dasar, dan batasnya sebagaimana tercantum dalam whitepaper publik. Setiap imbalan dasar dikalikan pengali emisi dan skor kebergunaan; batas harian diatur ulang pukul 00.00 WIB.

Tindakan	Imbalan dasar (PROFIT)	Batas
Umpan balik atas sebuah pindaian	0,01 sampai 0,05	10 per hari
Sebuah kiriman	0,02	1
Balasan, *repost*, atau tanda suka yang diterima	0,01	2 / 1 / 5
Membuat kanal	0,05	2
Referal (kedua pihak, setelah verifikasi surel)	0,25	sekali per orang yang dirujuk; total 50

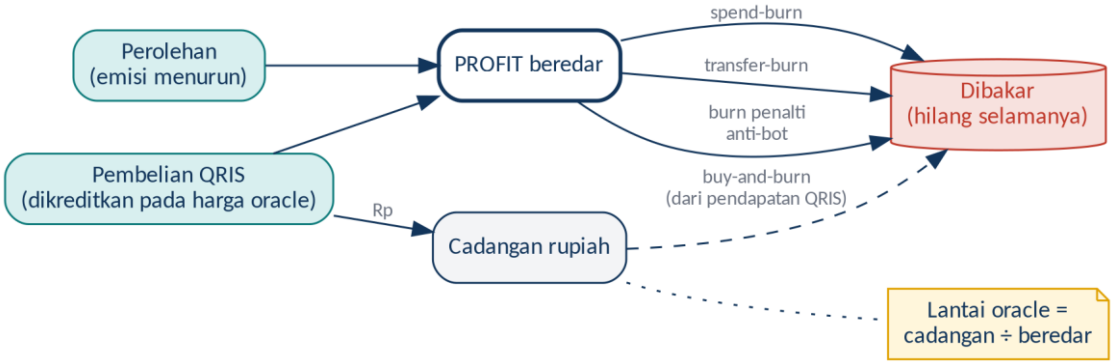
Dua aturan dari bab-bab sebelumnya melengkapi gambaran ini. Hanya manusia yang telah memverifikasi surel yang memperoleh imbalan (Bab 3), dan skor kebergunaan memberi nilai 0,00 pada kiriman remeh atau mirip bot, dengan penahanan (*escrow*) dan penarikan kembali (*clawback*) untuk apa pun yang kemudian dinilai sebagai penyalahgunaan; konten duplikat tidak memperoleh apa pun dan ditautkan ke kiriman yang lebih awal (Bab 5). Batas harian diatur ulang pukul 00.00 WIB, aturan yang dijelaskan panjang lebar di Bab 8. Karena itu, pengali dari Persamaan (12.2) bukanlah gaji, melainkan sebuah kenop: kenop yang memutar setiap imbalan sedikit lebih kecil setiap kali komunitas tumbuh sepuluh kali lipat.

12.7 Empat Jalan PROFIT Keluar: Pembakaran

12.7.1 Arti pembakaran

Membakar (*burn*) sebuah token berarti memusnahkan satuannya secara permanen. Di Solana, Token Program memiliki instruksi *burn* yang menghapus satuan dari sebuah akun token dan mengurangi pasokan mint dengan jumlah yang sama; setelah dibakar, satuan itu tidak ada di mana pun. Pembakaran adalah kebalikan dari pencetakan (*minting*), dan merupakan satu-satunya cara jumlah beredar sebuah token berpasokan tetap dapat menyusut.

Mengapa sebuah platform memusnahkan koin imbalannya sendiri? Karena emisi dan pembakaran bersama-sama menentukan berapa banyak PROFIT yang beredar, dan imbalan yang diciptakan dengan leluasa tetapi tidak pernah dimusnahkan hanya akan menumpuk. Rancangan whitepaper memberi PROFIT empat pembakaran [1], yang digambarkan sebagai aliran pada **Gambar 12.3**.



Gambar 12.3 Bagaimana PROFIT masuk dan keluar dari peredaran. Perolehan dan pembelian QRIS menambah peredaran; empat pembakaran menghapus PROFIT secara permanen; lantai oracle membagi cadangan Rupiah dengan jumlah yang beredar.

12.7.2 Empat pembakaran

1. **Spend-burn.** Ketika pengguna membelanjakan PROFIT untuk sebuah layanan, seperti tambahan kuota pindaian atau penukaran masa Pro, PROFIT yang dibelanjakan dibakar, bukan dipindahkan ke dompet perusahaan. Ini mengikuti gagasan yang diperkenalkan Ethereum melalui EIP-1559, ketika biaya dasar setiap transaksi "selalu dibakar" [13], dan yang diterapkan Solana pada separuh setiap biaya dasar [5]: memakai sistem menghabiskan sebagian satuannya.
2. **Transfer-burn.** Karena pengguna dapat saling mengirim PROFIT (Subbab 12.11), setiap transfer dikenai biaya kecil, dan biaya itu dibakar. Biaya transfer yang masuk ke platform akan menjadi pungutan; biaya yang dibakar menjadi rem kecil terhadap pemindahan koin bolak-balik.
3. **Buy-and-burn.** Sebagian pendapatan Rupiah yang nyata dari QRIS dipakai, menurut rancangan, untuk menarik PROFIT dari peredaran. Ini mengaitkan sebagian pembakaran dengan kegiatan komersial platform yang sesungguhnya, bukan semata dengan kegiatan token.
4. **Pembakaran penalti anti-bot.** Imbalan yang ditahan untuk kegiatan yang kemudian dinilai mirip bot atau menyalahgunakan sistem ditarik kembali lalu dibakar, bukan dibagikan ulang. Kecurangan tidak hanya gagal menghasilkan; kecurangan menghapus apa yang coba diambilnya.

Whitepaper menambahkan sebuah invarian untuk sistem yang telah matang: **begitu matang, pembakaran setidaknya sama dengan emisi**, $B \geq E$ [1]. Dalam sistem muda dengan sedikit pengguna dan sedikit pembelanjaan, emisi mendominasi, dan memang harus begitu agar ada yang memperoleh sesuatu. Tujuan rancangannya adalah ketika penggunaan tumbuh, pembelanjaan dan transfer tumbuh sebanding dengan jumlah orang, sedangkan emisi hanya tumbuh sebagai $u^{0.7}$, sehingga pembakaran akhirnya menyusul. Subbab 12.9 menghitung kapan hal itu terjadi, dengan asumsi sederhana.

> **WAWASAN KUNCI.** Pembakaran bukan trik untuk menaikkan harga. Pada koin utilitas, pembakaran adalah pembukuan yang bertujuan: setiap pembakaran terkait dengan penggunaan nyata (pembelanjaan, transfer, pendapatan, atau tertangkapnya penyalahgunaan), sehingga jumlah beredar mengikuti jumlah kegiatan nyata.

12.8 Pasokan sebagai Deret Geometri

12.8.1 Model yang cukup sederhana untuk diselesaikan

Sekarang kita membangun model pasokan beredar yang cukup sederhana untuk diselesaikan secara tepat dan cukup kaya untuk mengajarkan sesuatu. Kita ambil tiga asumsi, masing-masing penyederhanaan yang disengaja.

1. Jumlah pengguna tetap, sehingga emisi harian E konstan (untuk 100.000 pengguna menurut Persamaan (12.3), $E = 3{,}125$ PROFIT per hari).
2. Setiap hari, sebagian tetap b dari apa pun yang beredar dibakar melalui pembelanjaan dan transfer. Laju pembakaran $b = 0.005$ berarti setengah persen dari pasokan beredar habis terpakai setiap hari.
3. Hari dimulai dengan pembakaran dan diakhiri dengan emisi, dan peredaran dimulai dari nol.

Misalkan S_t adalah pasokan beredar pada akhir hari ke-t, dengan $S_0 = 0$. Maka setiap hari

$$S_{t+1} = (1 - b)\, S_t + E \tag{12.6}$$

Ini adalah *relasi rekurens linear orde pertama*: nilai esok adalah kelipatan tetap dari nilai hari ini ditambah sebuah konstanta. Sebelum menyelesaikannya, hitunglah beberapa hari secara manual dengan $E = 3{,}125$ dan $b = 0.005$, sehingga $1 - b = 0.995$:

- $S_1 = 0.995 \times 0 + 3{,}125 = 3{,}125$
- $S_2 = 0.995 \times 3{,}125 + 3{,}125 = 3{,}109.375 + 3{,}125 = 6{,}234.375$
- $S_3 = 0.995 \times 6{,}234.375 + 3{,}125 = 6{,}203.203125 + 3{,}125 = 9{,}328.203125$

Setiap hari menambah sedikit kurang dari 3.125, karena pembakaran tumbuh bersama pasokan.

12.8.2 Mengurai relasi rekurens

Tulis $q = 1 - b$ agar aljabarnya ringkas, lalu terapkan Persamaan (12.6) pada dirinya sendiri:

$$S_t = qS_{t-1} + E = q(qS_{t-2} + E) + E = q^2 S_{t-2} + qE + E$$

Teruskan sampai S_0 muncul. Setelah t langkah,

$$S_t = q^t S_0 + E\,(1 + q + q^2 + \cdots + q^{t-1})$$

Isi kurung itu adalah *deret geometri*: jumlah yang setiap sukunya adalah suku sebelumnya dikali rasio yang sama q. Bentuk tertutupnya adalah salah satu identitas paling berguna dalam matematika [14]. Sebut jumlah itu G lalu kalikan dengan q:

$$G = 1 + q + \cdots + q^{t-1}, \qquad qG = q + q^2 + \cdots + q^t$$

Kurangkan baris kedua dari baris pertama. Semua suku saling menghapus kecuali suku pertama G dan suku terakhir qG, sehingga tersisa $G - qG = 1 - q^t$; jadi untuk $q \neq 1$

$$\sum_{k=0}^{t-1} q^k = \frac{1 - q^t}{1 - q} \tag{12.7}$$

Dengan $S_0 = 0$ dan $1 - q = b$, pasokan setelah t hari adalah

$$S_t = E \cdot \frac{1-(1-b)^t}{b} \tag{12.8}$$

Periksa terhadap perhitungan manual: $S_3 = 3{,}125 \times (1 - 0.995^3)/0.005 = 625{,}000 \times (1 - 0.985074875) = 625{,}000 \times 0.014925125 = 9{,}328.203125$. Bentuk tertutup dan perhitungan langkah demi langkah sama hingga digit terakhir.

12.8.3 Keadaan tunak dan seberapa cepat dicapai

Karena $0 < 1 - b < 1$, pangkat $(1-b)^t$ menyusut menuju nol ketika t membesar, dan Persamaan (12.8) mendekati sebuah limit:

$$S_\infty = \lim_{t\to\infty} S_t = \frac{E}{b} \tag{12.9}$$

Keadaan tunak (*steady state*) ini memiliki makna langsung: inilah pasokan ketika pembakaran harian $b \cdot S$ tepat sama dengan emisi harian E. Di bawahnya, emisi menang dan pasokan naik; di atasnya, pembakaran menang dan pasokan turun. Dengan $E = 3{,}125$ dan $b = 0.005$, $S_\infty = 625{,}000$ PROFIT. Menggandakan laju pembakaran membuat keadaan tunak menjadi separuh; memangkas emisi menjadi separuh juga membuatnya separuh.

Berapa lama sampai pasokan mendekati keadaan tunak? Bagian keadaan tunak yang tercapai setelah t hari adalah $S_t/S_\infty = 1 - (1-b)^t$. Menyamakannya dengan 0.9 lalu menyelesaikan untuk t dengan logaritma memberi hari pertama ketika 90% tercapai:

$$t_{90} = \left\lceil \frac{\ln 0.1}{\ln(1-b)} \right\rceil \tag{12.10}$$

Untuk $b = 0.005$: $\ln 0.1 = -2.302585$ dan $\ln 0.995 = -0.005012542$, sehingga hasil baginya 459.36 dan $t_{90} = 460$ hari. Laju pembakaran kecil berarti keadaan tunak tinggi yang didekati perlahan; laju besar berarti keadaan tunak rendah yang dicapai dengan cepat.

12.8.4 Lab G8, langkah 3: buku besar dalam mikro-PROFIT melawan bentuk tertutup

Program ketiga menyimulasikan Persamaan (12.6) hari demi hari seperti sebuah buku besar, dalam mikro-PROFIT bulat, lalu membandingkannya dengan Persamaan (12.8). Pembakaran dihitung dengan pembagian bilangan bulat, karena buku besar tidak dapat membakar pecahan satuan dasar; inilah satu-satunya tempat simulasi dan rumus harus berbeda, dan program mengukur seberapa besar bedanya. Program juga memeriksa hukum kekekalan yang dipenuhi setiap buku besar yang jujur: pada setiap hari, pasokan sama dengan seluruh yang diemisikan dikurangi seluruh yang dibakar. **Kode 12.5** adalah versi Python.

Kode 12.5 `03_supply_sim.py`: relasi rekurens Persamaan (12.6) dijalankan sebagai buku besar bilangan bulat, diperiksa terhadap bentuk tertutup Persamaan (12.8), dengan keadaan tunak Persamaan (12.9) dan t_{90} dari Persamaan (12.10).

```
"""Lab G8, step 3: circulating supply under daily emission and burning.

Illustrative, not financial. Every day the platform emits E PROFIT and burns
a fixed share b of what is circulating. The ledger works in whole micro-PROFIT;
the closed form S_t = E * (1 - (1 - b)**t) / b checks it.
usage: python 03_supply_sim.py [users] [burn_bps] [days]
"""
import math
import sys

SCALE = 1_000_000                       # micro-PROFIT per PROFIT
CHECKPOINTS = [0, 30, 90, 180, 365, 730]


def half_up(x):
    """Round half away from zero, the same way in Python and Java."""
    return math.floor(x + 0.5)


def daily_emission_micro(users):
    """Whitepaper example rule: 1 PROFIT/user/day x 0.5**log10(users)."""
    profit = users * 0.5 ** math.log10(users)
    return half_up(profit * SCALE)


def simulate(e_micro, burn_bps, days):
    """Return lists of supply, total emitted and total burned, in micro."""
    supply, emitted, burned = [0], [0], [0]
    s = em = bu = 0
    for _ in range(days):
        burn = s * burn_bps // 10_000   # floor: a ledger cannot burn a fraction
        s = s - burn + e_micro
        em += e_micro
        bu += burn
        supply.append(s)
        emitted.append(em)
        burned.append(bu)
    return supply, emitted, burned


def closed_form(e, b, t):
    return e * (1 - (1 - b) ** t) / b
```

Kode 12.5 (Lanjutan) *03_supply_sim.py: relasi rekurens Persamaan (12.6) dijalankan sebagai buku besar bilangan bulat, diperiksa terhadap bentuk tertutup Persamaan (12.8), dengan keadaan tunak Persamaan (12.9) dan t{90} dari Persamaan (12.10).*

```
def bar(value, top, width=40):
    return "#" * half_up(width * value / top)


def main():
    users = int(sys.argv[1]) if len(sys.argv) > 1 else 100_000
    burn_bps = int(sys.argv[2]) if len(sys.argv) > 2 else 50
    days = int(sys.argv[3]) if len(sys.argv) > 3 else 730
    e_micro = daily_emission_micro(users)
    e, b = e_micro / SCALE, burn_bps / 10_000
    steady = e / b
    print(f"users={users:,}  emission E={e:,.6f} PROFIT/day"
          f"  burn b={b:.4f}/day  steady state E/b={steady:,.0f}")
    supply, emitted, burned = simulate(e_micro, burn_bps, days)
    print(f"{'day':>4} {'ledger supply':>18} {'closed form':>18}"
          f" {'gap (micro)':>11}")
    for t in CHECKPOINTS:
        if t > days:
            break
        exact = closed_form(e, b, t)
        gap = supply[t] - half_up(exact * SCALE)
        print(f"{t:>4} {supply[t] / SCALE:>18,.6f} {exact:>18,.6f}"
              f" {gap:>11,}")
    print("conservation: supply == emitted - burned on every day:",
          all(s == em - bu for s, em, bu in zip(supply, emitted, burned)))
    print(f"after {days} days: emitted {emitted[-1] / SCALE:,.0f},"
          f" burned {burned[-1] / SCALE:,.0f}")
    print("supply as a share of the steady state (each # = 2.5%):")
    for t in range(0, days + 1, 73):
        print(f"{t:>4} |{bar(supply[t] / SCALE, steady):<40}|"
              f" {100 * supply[t] / SCALE / steady:5.1f}%")
    print("how the burn rate shapes the curve:")
    for bps in (20, 50, 100):
        rate = bps / 10_000
        t90 = math.ceil(math.log(0.1) / math.log(1 - rate))
        print(f"  b={rate:.4f}: steady state {e / rate:>12,.0f} PROFIT,"
              f" 90% reached on day {t90}")


if __name__ == "__main__":
    main()
```

Dua detail dalam kode itu layak dikomentari. Fungsi bantu `half_up` membulatkan setengah menjauhi nol dengan `floor(x + 0.5)`, karena `round` bawaan Python membulatkan seri ke angka genap sedangkan `Math.round` di Java membulatkannya ke atas, padahal kedua versi harus sepakat. Selain itu, program menerima tiga argumen opsional (pengguna, laju pembakaran dalam basis poin, dan jumlah hari), sehingga pembaca dapat mencoba skenario lain tanpa menyunting kode. **Kode 12.6** adalah versi Java.

Kode 12.6 `SupplySim.java`: simulasi buku besar yang sama dalam Java, dengan larik mikro-PROFIT bertipe `long`.

```java
import java.util.Locale;

/**
 * Lab G8, step 3: circulating supply under daily emission and burning.
 * Illustrative, not financial. usage: java SupplySim [users] [burn_bps] [days]
 */
public class SupplySim {
    static final long SCALE = 1_000_000L;            // micro-PROFIT per PROFIT
    static final int[] CHECKPOINTS = {0, 30, 90, 180, 365, 730};

    /** Round half away from zero, the same way in Python and Java. */
    static long halfUp(double x) {
        return (long) Math.floor(x + 0.5);
    }

    /** Whitepaper example rule: 1 PROFIT/user/day x 0.5^log10(users). */
    static long dailyEmissionMicro(long users) {
        double profit = users * Math.pow(0.5, Math.log10(users));
        return halfUp(profit * SCALE);
    }

    static double closedForm(double e, double b, int t) {
        return e * (1 - Math.pow(1 - b, t)) / b;
    }

    static String bar(double value, double top) {
        return "#".repeat((int) halfUp(40 * value / top));
    }

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        long users = args.length > 0 ? Long.parseLong(args[0]) : 100_000;
        int burnBps = args.length > 1 ? Integer.parseInt(args[1]) : 50;
        int days = args.length > 2 ? Integer.parseInt(args[2]) : 730;
        long eMicro = dailyEmissionMicro(users);
        double e = (double) eMicro / SCALE, b = burnBps / 10_000.0;
        double steady = e / b;
        System.out.printf("users=%,d  emission E=%,.6f PROFIT/day  burn b=%.4f"
            + "/day  steady state E/b=%,.0f%n", users, e, b, steady);
```

Kode 12.6 (Lanjutan) *SupplySim.java: simulasi buku besar yang sama dalam Java, dengan larik mikro-PROFIT bertipe long.*

```
        long[] supply = new long[days + 1];
        long[] emitted = new long[days + 1];
        long[] burned = new long[days + 1];
        for (int t = 1; t <= days; t++) {
            long s = supply[t - 1];
            long burn = s * burnBps / 10_000;   // floor: no fractional burn
            supply[t] = s - burn + eMicro;
            emitted[t] = emitted[t - 1] + eMicro;
            burned[t] = burned[t - 1] + burn;
        }
        System.out.printf("%4s %18s %18s %11s%n",
            "day", "ledger supply", "closed form", "gap (micro)");
        for (int t : CHECKPOINTS) {
            if (t > days) break;
            double exact = closedForm(e, b, t);
            long gap = supply[t] - halfUp(exact * SCALE);
            System.out.printf("%4d %,18.6f %,18.6f %,11d%n",
                t, (double) supply[t] / SCALE, exact, gap);
        }
        boolean conserved = true;
        for (int t = 0; t <= days; t++) {
            conserved &= supply[t] == emitted[t] - burned[t];
        }
        System.out.println("conservation: supply == emitted - burned on every"
            + " day: " + (conserved ? "True" : "False"));
        System.out.printf("after %d days: emitted %,.0f, burned %,.0f%n", days,
            (double) emitted[days] / SCALE, (double) burned[days] / SCALE);
        System.out.println("supply as a share of the steady state"
            + " (each # = 2.5%):");
        for (int t = 0; t <= days; t += 73) {
            double v = (double) supply[t] / SCALE;
            System.out.printf("%4d |%-40s| %5.1f%%%n",
                t, bar(v, steady), 100 * v / steady);
        }
        System.out.println("how the burn rate shapes the curve:");
        for (int bps : new int[] {20, 50, 100}) {
            double rate = bps / 10_000.0;
            long t90 = (long) Math.ceil(Math.log(0.1) / Math.log(1 - rate));
            System.out.printf("  b=%.4f: steady state %,12.0f PROFIT,"
                + " 90%% reached on day %d%n", rate, e / rate, t90);
        }
    }
}
```

Tanpa argumen (100.000 pengguna, pembakaran 50 basis poin, yaitu 0,5% per hari, selama 730 hari), keduanya mencetak:

```
users=100,000  emission E=3,125.000000 PROFIT/day  burn b=0.0050/day  steady state
E/b=625,000
 day      ledger supply        closed form gap (micro)
   0           0.000000           0.000000           0
  30      87,259.880067      87,259.880053          14
  90     226,931.983853     226,931.983816          37
 180     371,466.967245     371,466.967187          58
 365     524,699.178701     524,699.178622          79
 730     608,903.592463     608,903.592370          93
conservation: supply == emitted - burned on every day: True
after 730 days: emitted 2,281,250, burned 1,672,346
supply as a share of the steady state (each # = 2.5%):
   0 |                                        |   0.0%
  73 |############                            |  30.6%
 146 |####################                    |  51.9%
 219 |###########################             |  66.6%
 292 |###############################         |  76.9%
 365 |###################################     |  84.0%
 438 |#####################################   |  88.9%
 511 |#######################################  |  92.3%
 584 |######################################## |  94.6%
 657 |######################################## |  96.3%
 730 |######################################## |  97.4%
how the burn rate shapes the curve:
  b=0.0020: steady state    1,562,500 PROFIT, 90% reached on day 1151
  b=0.0050: steady state      625,000 PROFIT, 90% reached on day 460
  b=0.0100: steady state      312,500 PROFIT, 90% reached on day 230
```

Keluaran itu memiliki lima bagian, dan masing-masing mengonfirmasi sesuatu.

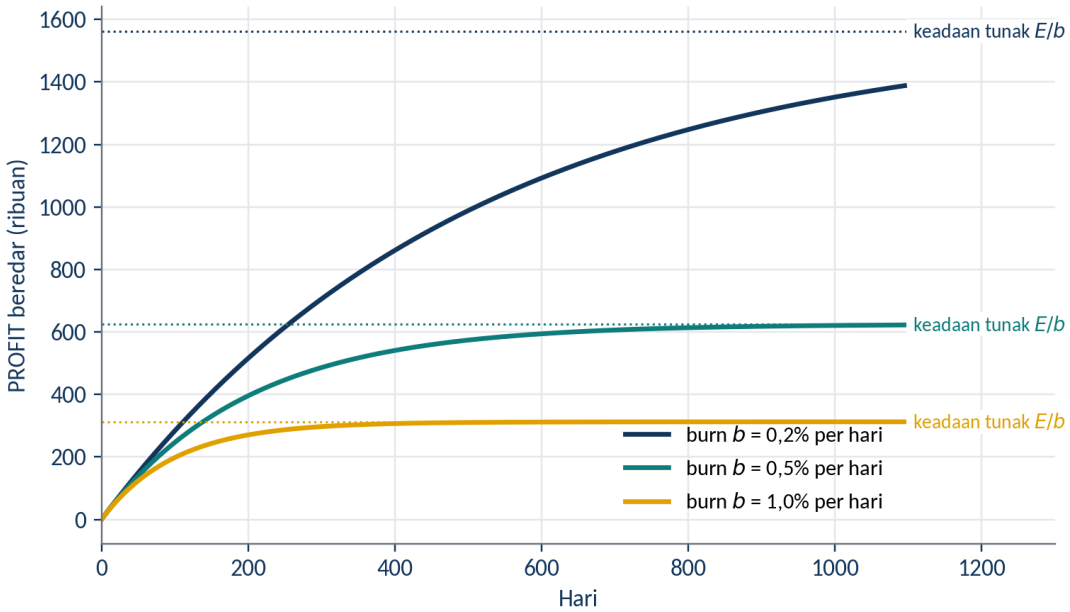
- Baris pertama menghitung E dari jumlah pengguna dan $S_\infty = E/b = 625{,}000$ dari Persamaan (12.9).
- Tabel membandingkan buku besar dengan bentuk tertutup. Kolom *gap*, dalam mikro-PROFIT, tidak pernah negatif dan tumbuh perlahan menjadi 93 mikro-PROFIT setelah dua tahun: buku besar selalu membulatkan pembakaran ke bawah, sehingga setiap hari ia membakar sepecahan mikro-PROFIT lebih sedikit daripada rumus, dan sisa-sisa pecahan itu menumpuk. Sembilan puluh tiga per sejuta PROFIT pada pasokan 608.903 adalah selisih relatif sekitar 1.5×10^{-10}. Rumus dan buku besar sepakat, dan program menyatakan dengan tepat seberapa dekat kesepakatannya.
- Baris kekekalan bernilai `True`: pada setiap hari dari 731 hari, pasokan sama dengan yang diemisikan dikurangi yang dibakar, hingga satuan mikro-PROFIT.
- Diagram batang memperlihatkan pasokan sebagai bagian dari keadaan tunak setiap 73 hari. Pasokan naik cepat pada awalnya lalu melandai, mencapai 84,0% pada satu tahun dan 97,4% pada dua tahun.
- Tiga baris terakhir memperlihatkan keadaan tunak dan t_{90} untuk tiga laju pembakaran. Pada $b = 0.005$, $t_{90} = 460$, sama dengan perhitungan manual di atas.

Program yang sama dengan argumen lain menjelajahi dunia lain. Berikut hasilnya untuk sejuta pengguna dan pembakaran 1% per hari selama satu tahun (`python 03_supply_sim.py 1000000 100 365`, atau `java SupplySim 1000000 100 365`):

```
users=1,000,000  emission E=15,625.000000 PROFIT/day  burn b=0.0100/day  steady state
E/b=1,562,500
 day      ledger supply        closed form gap (micro)
   0           0.000000           0.000000           0
  30     406,718.166592     406,718.166581          11
  90     930,106.292719     930,106.292690          29
 180   1,306,550.047376   1,306,550.047331          45
 365   1,522,628.180592   1,522,628.180543          49
conservation: supply == emitted - burned on every day: True
after 365 days: emitted 5,703,125, burned 4,180,497
supply as a share of the steady state (each # = 2.5%):
   0 |                                        |   0.0%
  73 |####################                    |  52.0%
 146 |##############################          |  76.9%
 219 |####################################    |  88.9%
 292 |##########################################  |  94.7%
 365 |########################################## |  97.4%
how the burn rate shapes the curve:
  b=0.0020: steady state    7,812,500 PROFIT, 90% reached on day 1151
  b=0.0050: steady state    3,125,000 PROFIT, 90% reached on day 460
  b=0.0100: steady state    1,562,500 PROFIT, 90% reached on day 230
```

Pengguna sepuluh kali lipat membuat emisi lima kali lebih besar, 15.625 alih-alih 3.125, sebagaimana diramalkan Persamaan (12.5), dan keadaan tunaknya adalah $15{,}625/0.01 = 1{,}562{,}500$. Pembakaran yang lebih cepat juga membawa kurva mendekati keadaan tunak itu jauh lebih cepat: 97,4% setelah satu tahun, padahal pembakaran yang lebih lambat pada percobaan pertama memerlukan dua tahun.

Gambar 12.4 menggambar model itu untuk tiga laju pembakaran, dari buku besar bilangan bulat yang sama, selama tiga tahun.



Gambar 12.4 Pasokan beredar di bawah emisi tetap 3.125 PROFIT per hari dan tiga laju pembakaran harian, dari buku besar bilangan bulat Lab G8. Setiap kurva mendekati keadaan tunaknya sendiri E/b; laju pembakaran yang lebih tinggi memberi keadaan tunak yang lebih rendah dan tercapai lebih cepat. Nilai ilustratif saja.

PERINGATAN. Model ini memiliki pengguna tetap, emisi tetap, dan pembakaran yang merupakan bagian tetap dari pasokan. Platform nyata tidak memiliki satu pun dari ketiganya. Model ini berguna karena dapat diselesaikan secara tepat dan menyingkap *bentuk*

dinamikanya; model ini tidak meramalkan pasokan PROFIT yang sebenarnya, dan tidak ada bagiannya yang menyangkut harga.

12.9 Kapan Pembakaran Menyusul Emisi?

12.9.1 Per pengguna, pertanyaannya menjadi sederhana

Subbab 12.7 mengutip invarian rancangan untuk sistem yang telah matang: pembakaran setidaknya sama dengan emisi. Kapan platform yang sedang tumbuh sampai ke sana? Model keadaan tunak di Subbab 12.8 menetapkan jumlah pengguna; sekarang kita membiarkannya berubah dan mengajukan pertanyaan lain.

Andaikan, rata-rata, setiap pengguna aktif menyebabkan β PROFIT terbakar per hari melalui pembelanjaan dan biaya transfer. Maka pembakaran harian tumbuh sebanding dengan pengguna, $B(u) = \beta u$, sedangkan emisi harian tumbuh sebagai $E(u) = u \cdot m(u)$ menurut Persamaan (12.3) dengan $r = 1$. Bagi keduanya dengan u: pembakaran per pengguna adalah konstanta β, sedangkan emisi per pengguna adalah $m(u) = u^{-\log_{10} 2}$, yang menurun seiring tumbuhnya komunitas. Jadi pembakaran menyusul emisi tepat ketika

$$m(u) \le \beta \Leftrightarrow u^{-\log_{10} 2} \le \beta \Leftrightarrow u \ge u^* = \beta^{-1/\log_{10} 2} \tag{12.11}$$

Langkah terakhir memangkatkan kedua ruas $u^{-\log_{10} 2} \le \beta$ dengan $-1/\log_{10} 2$, yang negatif sehingga membalik arah ketaksamaan. Karena $1/\log_{10} 2 \approx 3.3219$, jumlah pengguna titik impas adalah $u^* = \beta^{-3.3219}$.

Kerjakan satu kasus secara manual. Untuk $\beta = 0.02$ PROFIT per pengguna per hari: $\log_{10} 0.02 = -1.69897$, dan $-1.69897 \times (-3.3219) = 5.6439$, sehingga $u^* = 10^{5.6439} \approx 440{,}409$ pengguna. Di bawah sekitar 440 ribu pengguna, emisi melampaui pembakaran; di atasnya, pembakaran menang.

Hukum pangkat membawa akibat yang menyenangkan. Pada $u^*/10$, emisi per pengguna adalah dua kali β (satu kali pembagian dua lebih sedikit), dan pada $10u^*$ nilainya separuh β. Jadi rasio pembakaran terhadap emisi adalah 0.5 satu dekade di bawah titik impas, 1 tepat di titik impas, dan 2 satu dekade di atasnya, berapa pun β.

12.9.2 Lab G8, langkah 4: bentuk tertutup, bagi dua, dan oracle mainan

Program keempat menghitung u^* dua kali: sekali dengan Persamaan (12.11) dan sekali dengan metode *bagi dua* (*bisection*), metode andal paling sederhana untuk menyelesaikan persamaan secara numerik [15]. Metode bagi dua mempertahankan sebuah selang yang pasti memuat jawabannya, melihat titik tengahnya, lalu membuang separuh yang tidak mungkin memuat jawaban; setelah 60 kali pembagian dua dari selang selebar 12 (dalam pangkat sepuluh), lebar yang tersisa adalah $12/2^{60} \approx 10^{-17}$, jauh di bawah apa yang dapat dibedakan sebuah `double`. Menyelesaikan pada $x = \log_{10} u$ alih-alih pada u itu sendiri menjaga selangnya kecil dan langkahnya merata. Memiliki dua cara independen untuk memperoleh angka yang sama adalah versi seorang analis numerik dari sebuah tes: jika keduanya berselisih, salah satunya keliru.

Program yang sama juga menjalankan oracle mainan dari Subbab 12.10, sehingga kita menampilkannya lengkap di sini dan membahas paruh kedua keluarannya di subbab itu. **Kode 12.7** adalah versi Python.

Kode 12.7 `04_breakeven_oracle.py`: jumlah pengguna titik impas menurut Persamaan (12.11) dan menurut metode bagi dua, serta lantai oracle mainan.

```
"""Lab G8, step 4: where burning overtakes emission, and a toy oracle floor.

Illustrative, not financial. Per user and per day, the platform emits
m(u) = 0.5 ** log10(u) PROFIT and burns beta PROFIT through spending.
Burn >= emission once m(u) <= beta, i.e. u >= beta ** (-1 / log10(2)).
"""
import math

SCALE = 1_000_000                          # micro-PROFIT per PROFIT
LOG2 = math.log10(2)


def per_user_emission(users):
    return 0.5 ** math.log10(users)


def break_even_closed(beta):
    return beta ** (-1 / LOG2)


def break_even_bisect(beta, lo=0.0, hi=12.0, steps=60):
    """Bisection on x = log10(u) for m(10**x) - beta = 0."""
    for _ in range(steps):
        mid = (lo + hi) / 2
        if per_user_emission(10 ** mid) > beta:
            lo = mid                       # still emitting more than burning
        else:
            hi = mid
    return 10 ** hi


def show_break_even():
    print("per-user burn beta -> users u* where burn >= emission")
    print(f"  {'beta':>6} {'u* closed form':>15} {'u* bisection':>15}"
          f" {'burn/emission at u*/10':>23} {'u*':>5} {'10u*':>5}")
    for beta in (0.005, 0.01, 0.02, 0.05):
        u_star = break_even_closed(beta)
        u_bis = break_even_bisect(beta)
        r = [beta / per_user_emission(u_star * k) for k in (0.1, 1, 10)]
        print(f"  {beta:>6} {u_star:>15,.0f} {u_bis:>15,.0f}"
              f" {r[0]:>23.2f} {r[1]:>5.2f} {r[2]:>5.2f}")


def floor_price(reserve_rp, circulating_micro):
    return reserve_rp * SCALE / circulating_micro
```

Kode 12.7 (Lanjutan) *04_breakeven_oracle.py: jumlah pengguna titik impas menurut Persamaan (12.11) dan menurut metode bagi dua, serta lantai oracle mainan.*

```
def show_oracle():
    print("toy oracle: floor = rupiah reserve / circulating PROFIT")
    reserve = 100_000_000                  # rupiah
    circ = 100_000 * SCALE                 # micro-PROFIT
    steps = [("start", 0, 0)]
    steps.append(("community emission +5,000", 0, 5_000 * SCALE))
    steps.append(("spend-burn -3,000", 0, -3_000 * SCALE))
    steps.append(("QRIS purchase Rp1,000,000", 1_000_000, None))
    steps.append(("transfer-burn -200", 0, -200 * SCALE))
    print(f"  {'event':<27} {'reserve (Rp)':>14} {'circulating':>14}"
          f" {'floor (Rp)':>10}")
    for name, add_rp, add_micro in steps:
        if add_micro is None:              # buy at the current floor price
            add_micro = add_rp * circ // reserve
        reserve += add_rp
        circ += add_micro
        print(f"  {name:<27} {reserve:>14,} {circ / SCALE:>14,.2f}"
              f" {floor_price(reserve, circ):>10,.2f}")


def main():
    show_break_even()
    show_oracle()


if __name__ == "__main__":
    main()
```

Kode 12.8 adalah versi Java. Larik objek `Long`, dengan `null` untuk langkah pembelian, mencerminkan `None` di Python: penanda satu-satunya peristiwa yang jumlah PROFIT-nya tidak diberikan melainkan dihitung dari lantai saat itu.

Kode 12.8 `BreakEvenOracle.java`: dua perhitungan yang sama dalam Java.

```
import java.util.Locale;

/**
 * Lab G8, step 4: where burning overtakes emission, and a toy oracle floor.
 * Illustrative, not financial. Burn >= emission once 0.5^log10(u) <= beta,
 * i.e. u >= beta^(-1 / log10 2).
 */
public class BreakEvenOracle {
    static final long SCALE = 1_000_000L;          // micro-PROFIT per PROFIT
    static final double LOG2 = Math.log10(2);

    static double perUserEmission(double users) {
        return Math.pow(0.5, Math.log10(users));
    }

    static double breakEvenClosed(double beta) {
        return Math.pow(beta, -1 / LOG2);
    }

    /** Bisection on x = log10(u) for m(10^x) - beta = 0. */
    static double breakEvenBisect(double beta) {
        double lo = 0.0, hi = 12.0;
        for (int i = 0; i < 60; i++) {
            double mid = (lo + hi) / 2;
            if (perUserEmission(Math.pow(10, mid)) > beta) {
                lo = mid;                  // still emitting more than burning
            } else {
                hi = mid;
            }
        }
        return Math.pow(10, hi);
    }

    static void showBreakEven() {
        System.out.println(
            "per-user burn beta -> users u* where burn >= emission");
        System.out.printf("  %6s %15s %15s %23s %5s %5s%n", "beta",
            "u* closed form", "u* bisection", "burn/emission at u*/10", "u*",
            "10u*");
        for (double beta : new double[] {0.005, 0.01, 0.02, 0.05}) {
            double uStar = breakEvenClosed(beta);
            double uBis = breakEvenBisect(beta);
            double[] r = new double[3];
            double[] k = {0.1, 1, 10};
            for (int i = 0; i < 3; i++) {
                r[i] = beta / perUserEmission(uStar * k[i]);
```

Kode 12.8 (Lanjutan) *BreakEvenOracle.java: dua perhitungan yang sama dalam Java.*

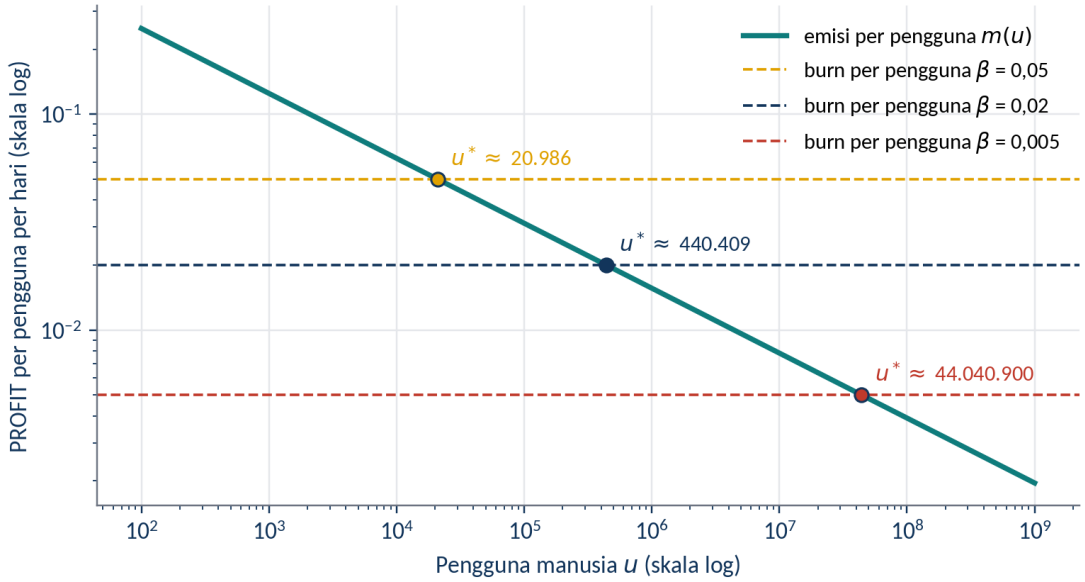
```
            }
            System.out.printf("  %6s %,15.0f %,15.0f %23.2f %5.2f %5.2f%n",
                Double.toString(beta), uStar, uBis, r[0], r[1], r[2]);
        }
    }

    static double floorPrice(long reserveRp, long circulatingMicro) {
        return (double) reserveRp * SCALE / circulatingMicro;
    }

    static void showOracle() {
        System.out.println(
            "toy oracle: floor = rupiah reserve / circulating PROFIT");
        long reserve = 100_000_000L;              // rupiah
        long circ = 100_000L * SCALE;             // micro-PROFIT
        String[] names = {"start", "community emission +5,000",
            "spend-burn -3,000", "QRIS purchase Rp1,000,000",
            "transfer-burn -200"};
        long[] addRp = {0, 0, 0, 1_000_000L, 0};
        Long[] addMicro = {0L, 5_000L * SCALE, -3_000L * SCALE, null,
            -200L * SCALE};
        System.out.printf("  %-27s %14s %14s %10s%n",
            "event", "reserve (Rp)", "circulating", "floor (Rp)");
        for (int i = 0; i < names.length; i++) {
            long add = addMicro[i] == null
                ? addRp[i] * circ / reserve       // buy at the current floor
                : addMicro[i];
            reserve += addRp[i];
            circ += add;
            System.out.printf("  %-27s %,14d %,14.2f %,10.2f%n", names[i],
                reserve, (double) circ / SCALE, floorPrice(reserve, circ));
        }
    }

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        showBreakEven();
        showOracle();
    }
}
```

Keduanya mencetak:

```
per-user burn beta -> users u* where burn >= emission
    beta  u* closed form    u* bisection  burn/emission at u*/10    u*  10u*
   0.005      44,040,900      44,040,900                    0.50  1.00  2.00
    0.01       4,404,090       4,404,090                    0.50  1.00  2.00
    0.02         440,409         440,409                    0.50  1.00  2.00
    0.05          20,986          20,986                    0.50  1.00  2.00
toy oracle: floor = rupiah reserve / circulating PROFIT
  event                        reserve (Rp)    circulating floor (Rp)
  start                         100,000,000     100,000.00   1,000.00
  community emission +5,000     100,000,000     105,000.00     952.38
  spend-burn -3,000             100,000,000     102,000.00     980.39
  QRIS purchase Rp1,000,000     101,000,000     103,020.00     980.39
  transfer-burn -200            101,000,000     102,820.00     982.30
```

Tabel pertama memperlihatkan bentuk tertutup dan metode bagi dua sepakat hingga digit terakhir yang dicetak untuk empat laju pembakaran, dan ketiga kolom rasio bernilai 0.50, 1.00, dan 2.00 di setiap baris, sesuai perkiraan. Angka-angka itu juga memperlihatkan betapa pekanya u^* terhadap β: memangkas pembakaran per pengguna dari 0,01 menjadi 0,005 mengalikan komunitas titik impas dengan $2^{3.32} = 10$. **Gambar 12.5** menggambar hal yang sama: garis emisi

per pengguna yang menurun dan tiga garis mendatar pembakaran per pengguna, yang berpotongan di titik-titik impas.



Gambar 12.5 Emisi per pengguna $m(u)$ turun seiring tumbuhnya komunitas; pembakaran per pengguna yang konstan β adalah garis mendatar. Pembakaran menyusul emisi di sebelah kanan setiap perpotongan u^*. Nilai β ilustratif.

PERINGATAN. Nilai β di lab adalah masukan pembaca, dipilih agar matematikanya terlihat. Nilai itu bukan hasil pengukuran Profitina, dan titik-titik impasnya bukan ramalan kapan sesuatu akan terjadi pada PROFIT.

12.10 Oracle: Sebuah Lantai, Bukan Harga

12.10.1 Apa yang dihitung oracle

Platform yang menjual PROFIT untuk Rupiah melalui QRIS, dan menghargai sebagian layanannya dalam PROFIT, membutuhkan acuan berapa Rupiah nilai satu PROFIT *di dalam platform*. Profitina menyebut mekanisme penyedia acuan itu *oracle*, dan whitepaper publik menyatakan rumusnya: lantai adalah cadangan Rupiah dibagi PROFIT yang beredar [1]. Dengan R sebagai cadangan dalam Rupiah dan C sebagai pasokan beredar,

$$f = \frac{R}{C} \tag{12.12}$$

Acuan awalnya adalah **1 PROFIT = Rp1.000**, yang di whitepaper dan aplikasi disebut "estimasi model, bukan harga pasar atau janji" [1], [2]. Selama testnet, nilainya distabilkan pada atau di atas acuan awal, dan aplikasi menampilkan rata-rata bergerak 7 hari dengan pengaman terhadap manipulasi; penetapan harga dua arah adalah bagian dari mainnet [2].

Kata *lantai* dipilih dengan cermat. Seandainya setiap PROFIT yang beredar dapat ditukar dengan bagiannya dari cadangan, masing-masing akan menerima tepat f Rupiah; lantai adalah apa yang ditanggung cadangan, bukan harga yang akan dibayar seseorang. Dan karena penjualan dinonaktifkan, pada kenyataannya tidak ada yang dapat menukar PROFIT dengan Rupiah pada rilis September 2026. Lantai adalah satuan hitung untuk harga-harga platform itu sendiri.

12.10.2 Apa yang menggeser lantai, dan apa yang tidak

Persamaan (12.12) hanya memiliki dua bahan, sehingga hanya dua jenis peristiwa yang dapat menggesernya.

- **Emisi menurunkan lantai**, karena C bertambah sementara R tetap.
- **Pembakaran menaikkan lantai**, karena C berkurang sementara R tetap.
- **Pembelian pada harga lantai tidak mengubah lantai.** Pengguna yang membayar x Rupiah pada harga lantai $f = R/C$ menerima x/f PROFIT. Cadangan menjadi $R + x$ dan pasokan menjadi $C + x/f$, sehingga lantai barunya adalah

$$f' = \frac{R + x}{C + x/f} = \frac{R + x}{C + xC/R} = \frac{R + x}{C\,(R + x)/R} = \frac{R}{C} = f \tag{12.13}$$

Dengan kata lain: membeli pada harga lantai menambah cadangan dan pasokan tepat dalam perbandingan yang sudah dimiliki lantai. Karena itu, pembelian tidak dapat dipakai untuk mendorong lantai naik atau turun, dan itu salah satu alasan rancangannya mengkreditkan pembelian pada harga oracle.

Paruh kedua keluaran lab menelusuri lima peristiwa pada platform mainan dengan cadangan Rp100.000.000 dan 100.000 PROFIT beredar, sehingga lantainya dimulai dari acuan Rp1.000:

- Emisi komunitas sebesar 5.000 PROFIT menaikkan C menjadi 105.000 dan menurunkan lantai menjadi $100{,}000{,}000/105{,}000 = 952.38$.
- *Spend-burn* sebesar 3.000 menurunkan C menjadi 102.000 dan menaikkan lantai menjadi 980.39.
- Pembelian QRIS senilai Rp1.000.000 pada harga lantai menambah 1.020 PROFIT (dihitung dalam mikro-PROFIT bulat) dan Rp1.000.000 cadangan. Lantai tetap 980.39, sebagaimana dijanjikan Persamaan (12.13).
- *Transfer-burn* sebesar 200 menaikkan lantai menjadi 982.30.

Setiap angka ini dikarang untuk keperluan pelajaran. Yang penting adalah arah setiap panah, bukan besarnya.

> **PERINGATAN.** Lantai bukan harga pasar, bukan nilai penukaran, dan bukan janji nilai apa pun di masa depan. Pada rilis September 2026 tidak ada pasar tempat PROFIT dapat dijual untuk Rupiah pada harga ini maupun harga lain [1].

12.11 Membeli, Membelanjakan, dan Mentransfer: Pagar Pengaman

Koin yang dapat diperoleh, dibeli, dibelanjakan, dan dikirim ke orang lain memiliki empat pintu, dan setiap pintu membutuhkan kunci yang menjaga koin itu tetap sebagai utilitas. Subbab ini menelusuri pintu-pintu itu dalam urutan yang dijumpai pengguna.

12.11.1 Membeli melalui QRIS

QRIS adalah standar kode QR nasional Bank Indonesia untuk pembayaran, dan pada pertengahan 2026 QRIS memiliki 65,77 juta pengguna dan 44,86 juta titik penerimaan pedagang, 96,68% di antaranya usaha mikro, kecil, dan menengah [16]. Pengguna yang ingin memiliki PROFIT memilih jumlah antara Rp50.000 dan Rp1.000.000, membayar dengan memindai kode QRIS memakai aplikasi bank atau dompet elektronik mana pun yang ikut serta, dan menerima PROFIT pada

harga oracle setelah administrator memverifikasi pembayarannya [1], [2]. Bab 8 menjelaskan mengapa pembelian diverifikasi oleh manusia pada tahap ini: volume pembelian masih kecil, aturan "ukur minat sebelum membangun pembayaran" berlaku, dan pemeriksaan manual murah dibangun serta mudah diaudit.

Membeli adalah satu-satunya pintu tempat Rupiah masuk. Tidak ada pintu tempat Rupiah keluar: **penjualan atau penukaran PROFIT ke Rupiah dinonaktifkan** sampai mainnet dan kejelasan regulasi [1].

12.11.2 Membelanjakan untuk layanan nyata

PROFIT dapat dibelanjakan untuk layanan nyata di aplikasi: tambahan kuota pindaian untuk hari itu (tambahan berlaku hanya pada hari pembelian, menurut WIB), Academic Citation Scan, dan masa Pro (Bab 8). Harga diatur oleh platform dan ditampilkan sebelum pengguna mengonfirmasi; pembelian yang gagal dikembalikan; PROFIT yang dibelanjakan dibakar [1]. Di sekitar pembelanjaan, whitepaper menetapkan empat pagar pengaman, dan masing-masing layak mendapat satu kalimat.

1. **Pembelanjaan tidak mencetak.** Pembelanjaan tidak pernah menciptakan PROFIT baru di mana pun; pembelanjaan hanya memindahkan satuan ke pembakaran.
2. **Pembelian tidak pernah menaikkan peringkat.** Papan peringkat mengukur kontribusi, dengan bobot pada kualitas (Bab 4). Membeli atau membelanjakan PROFIT tidak dapat membeli posisi yang lebih baik.
3. **Tidak ada jalan pintas melewati Fair-Play atau keamanan.** Tidak ada jumlah PROFIT yang membuka izin mengirim tanpa pemeriksaan Bab 5: Link Safety, skor kebergunaan, deteksi duplikat, moderasi.
4. **Hak tetap gratis selamanya.** Mengekspor data, memperbaikinya, dan menghapus akun (Bab 11), serta setiap fitur keamanan, tidak pernah dihargai dalam PROFIT maupun Rupiah.

Ada pula penggunaan yang sama sekali tidak membelanjakan: membuat kanal mensyaratkan **saldo sedikitnya 50 PROFIT, yang tidak pernah dipotong** (Bab 4). Saldo itu adalah tanda keikutsertaan, bukan biaya.

12.11.3 Mentransfer antarpengguna

Sejak akhir Agustus 2026, pengguna dapat saling mengirim PROFIT berdasarkan *handle* [1]. Empat aturan membentuk pintu ini.

- **Biaya kecil, dibakar.** Setiap transfer dikenai biaya kecil, dan biaya itu dibakar (*transfer-burn* dari Subbab 12.7).
- **Hanya PROFIT yang diperoleh secara organik yang dapat ditransfer.** PROFIT yang dibeli dengan Rupiah, atau yang diberikan administrator, tidak dapat dikirim ke pengguna lain. Alasannya adalah celah yang ditutup aturan ini: seandainya PROFIT yang dibeli dapat ditransfer, seseorang dapat membayar Rupiah, mengirim PROFIT-nya kepada orang kedua, dan orang kedua dapat memakainya seolah-olah uang yang diterima dari orang pertama, sehingga platform berubah menjadi saluran tidak resmi untuk memindahkan nilai Rupiah antarorang. Membatasi transfer hanya pada PROFIT hasil perolehan menjaga transfer tetap sebagai cara berbagi pengakuan, bukan jalur pembayaran.
- **Log transfer publik.** Transfer tercatat dalam log publik, dengan nama disamarkan bagi pengunjung anonim. Keterbukaan tentang aliran koin imbalan mencegah kolusi dan membuat pola terlihat oleh semua orang.

- **Penahanan untuk penerima yang belum terverifikasi.** PROFIT yang dikirim kepada pengguna yang belum memverifikasi surel ditahan sampai ia melakukannya, sejalan dengan aturan bahwa hanya manusia terverifikasi yang ikut dalam perekonomian ini.

12.11.4 Pemberian administrator, secara terbuka

Administrator dapat memberikan PROFIT atas kebijakannya untuk lomba, kolaborasi, atau kompensasi. Pemberian itu memiliki batas per transaksi dan per hari serta diambil dari **kumpulan seumur hidup yang dibatasi 0,01% dari pasokan maksimum**; riwayatnya publik, dan PROFIT pemberian dikecualikan dari semua papan peringkat [1]. Kekuasaan yang dapat menciptakan nilai bagi orang-orang pilihan adalah jenis kekuasaan yang seharusnya kecil, dibatasi, dan terlihat, dan rancangan ini membuatnya memenuhi ketiganya.

> **WAWASAN KUNCI.** Setiap pagar pengaman menjawab pertanyaan yang sama: dapatkah pintu ini dipakai untuk mengubah PROFIT menjadi cara memindahkan uang atau membeli status? Jika jawabannya ya, pintu itu dipersempit sampai jawabannya tidak.

12.12 Satu Tingkat Referal, dan Mengapa Tidak Lebih

12.12.1 Aturannya

Pengguna yang mengundang teman menerima **0,25 PROFIT, demikian pula temannya**, setelah teman itu memverifikasi surelnya; setiap pasangan memperolehnya sekali seumur hidup, dan seseorang dapat diberi imbalan untuk paling banyak 50 referal [1]. Sebuah Papan Peringkat Referal terpisah memperlihatkan siapa yang telah mengajak orang bergabung. Itulah seluruh programnya: satu tingkat, imbalan kecil yang simetris, dibayar hanya untuk manusia yang terverifikasi.

12.12.2 Mengapa tingkat kedua adalah hal yang berbeda

Banyak program referal tidak hanya membayar untuk teman yang Anda undang, tetapi juga untuk teman yang *mereka* undang, dan seterusnya ke bawah beberapa tingkat. Profitina sengaja tidak melakukannya, dan whitepaper memberi alasannya dengan kata-kata lugas: tingkat berlapis "secara struktural mirip skema piramida" [1].

Matematikanya memperlihatkan mengapa. Andaikan setiap peserta merekrut k orang baru. Pada tingkat 0 ada satu orang; pada tingkat 1 ada k; pada tingkat 2, k^2; pada tingkat n, k^n. Banyaknya orang yang dibutuhkan untuk mengisi n tingkat di bawah satu orang kembali merupakan deret geometri, menurut Persamaan (12.7):

$$\sum_{i=1}^{n} k^i = k\,\frac{k^n - 1}{k - 1} \tag{12.14}$$

Dengan $k = 5$ dan hanya sepuluh tingkat, hasilnya $5 \times (5^{10} - 1)/4 = 12{,}207{,}030$ orang, lebih banyak daripada penduduk kota Jakarta; dengan $k = 10$ dan sepuluh tingkat, hasilnya lebih dari sebelas miliar, melebihi penduduk Bumi. Struktur imbalan yang membayar orang atas rekrutan dari rekrutannya hanya dapat terus membayar selama pohon eksponensial ini terus tumbuh, dan pohon itu pasti berhenti tumbuh jauh sebelum tingkat kesepuluh. Mereka yang bergabung belakangan, di dasar, membayar mereka yang di puncak. Itulah definisi piramida, apa pun namanya, dan Undang-Undang Perdagangan Indonesia, UU No. 7 Tahun 2014, melarang pelaku usaha distribusi menerapkan skema piramida [17].

Referal satu tingkat tumbuh secara linear dengan orang yang diundang langsung oleh setiap pengguna, dibatasi, dan membayar kedua pihak dengan jumlah kecil yang sama. Program ini menghargai tindakan memperkenalkan teman kepada produk yang berguna, dan tidak lebih.

> **CONTOH.** Ani mengundang Budi, dan Budi memverifikasi surelnya: Ani dan Budi masing-masing menerima 0,25 PROFIT (dikali pengali yang biasa). Budi lalu mengundang Citra, yang juga memverifikasi: Budi dan Citra masing-masing menerima 0,25 PROFIT. Ani tidak menerima apa pun untuk Citra. Jika Ani mengundang Budi lagi dengan akun kedua, tidak terjadi apa-apa: pasangan itu sudah pernah diberi imbalan, dan akun kedua untuk orang yang sama justru merupakan jenis kegiatan yang dirancang untuk ditangkap oleh pemeriksaan Fair-Play di Bab 5.

12.13 Sudah Berjalan, dan di Peta Jalan

Segala hal tentang PROFIT yang menyangkut nilai harus dipilah ke dalam dua kotak yang tidak pernah tertukar. **Tabel 12.5** melakukan pemilahan itu untuk rilis September 2026.

Tabel 12.5 Apa yang sudah berjalan dan apa yang ada di peta jalan untuk PROFIT, per September 2026.

Kemampuan	Status pada rilis September 2026
Token SPL PROFIT di devnet Solana, dapat diverifikasi publik	Berjalan (sejak akhir Juli 2026)
6 desimal; *freeze authority* tidak ada	Berjalan, sifat mint yang telah digelar
Perolehan dengan pengali menurun, batas, dan skor kebergunaan	Berjalan
Spend-burn, *transfer-burn*, pembakaran penalti anti-bot	Berjalan
Buy-and-burn dari pendapatan QRIS	Bagian dari rancangan; bermakna hanya jika ada pendapatan nyata
Lantai oracle dengan acuan awal Rp1.000, rata-rata 7 hari ditampilkan di aplikasi	Berjalan, sebagai estimasi model
Membeli PROFIT melalui QRIS (Rp50.000 sampai Rp1.000.000), diverifikasi administrator	Berjalan
Membelanjakan untuk tambahan kuota pindaian, Academic Citation Scan, dan masa Pro	Berjalan
Transfer antarpengguna, hanya PROFIT hasil perolehan, log publik, penahanan	Berjalan (sejak akhir Agustus 2026)
Referal satu tingkat, 0,25 PROFIT untuk setiap pihak	Berjalan (sejak Agustus 2026)
Pemberian administrator dari kumpulan terbatas yang publik	Berjalan
Mint authority dipindahkan ke multisig	Direncanakan
Menjual atau menukarkan PROFIT ke Rupiah	Dinonaktifkan sampai mainnet dan kejelasan regulasi
PROFIT di mainnet Solana	Peta jalan: Fase 3, berlabel K4 2026, setelah audit token dan perizinan
Ultralight DEX: penukaran nonkustodian melalui agregator Jupiter	Peta jalan: Fase 3, gerbang yang sama
Tata kelola oleh pemegang PROFIT	Disebut sebagai penggunaan kelak; belum ada rancangan yang diterbitkan

Dua baris dalam tabel membutuhkan sepatah kata kejujuran yang tidak dapat ditampung tabel.

Yang pertama adalah *buy-and-burn*. Mekanismenya bagian dari rancangan, tetapi besarnya bergantung pada pendapatan, dan Bab 8 mencatat temuan Agustus 2026 dengan lugas: nol

pembelian PROFIT sepanjang seluruh riwayat produksi pada saat itu [1]. Pembakaran yang dihidupi pendapatan hanya sebesar pendapatan yang menghidupinya.

Yang kedua adalah tanggal Fase 3. "K4 2026" adalah label di whitepaper, dan whitepaper itu sendiri melekatkan gerbangnya: audit token dan perizinan di bawah regulator yang disebut di Subbab 12.2.3 [1]. Gerbang adalah syarat, bukan kalender. Jika syaratnya belum terpenuhi pada suatu tanggal, tanggalnya yang bergeser; syaratnya tidak.

> **DEFINISI.** Bursa *nonkustodian* memungkinkan dua pihak menukar token langsung dari dompet masing-masing, melalui program di rantai, tanpa menyerahkan token mereka kepada perantara yang memegangnya. *Agregator* seperti Jupiter menelusuri beberapa tempat penukaran semacam itu dan mengarahkan penukaran melalui jalur terbaik yang tersedia.

12.14 Profitina dalam Praktik

Semua isi bab ini berpadanan dengan produk yang berjalan sebagai berikut, pada rilis September 2026.

- **Tempat pengguna melihat PROFIT.** Aplikasi memiliki area PROFIT yang menampilkan saldo pengguna, acuan oracle saat ini dengan rata-rata bergerak 7 hari dan catatan "estimasi model, bukan harga pasar", formulir pembelian QRIS, daftar layanan yang dapat dibayar dengan PROFIT beserta harganya, formulir transfer, dan log transfer publik [2]. Imbalan muncul di notifikasi dasbor ketika dikreditkan (Bab 4).
- **Tempat rantai muncul.** Mint devnet dari **Tabel 12.2** adalah wajah token di rantai, dan siapa pun dapat memeriksanya di Solana Explorer dengan klaster diatur ke devnet [18]. Buku besar platform mencatat setiap imbalan, pembelanjaan, pembakaran, dan transfer dalam mikro-PROFIT bulat, satuan yang sama dengan yang dipakai mint.
- **Cara aturan ditegakkan.** Perolehan melewati gerbang verifikasi surel, batas harian yang diatur ulang pukul 00.00 WIB, skor kebergunaan, dan deteksi duplikat (Bab 3, 5, dan 8). Pembelanjaan melewati empat pagar pengaman di Subbab 12.11. Transfer melewati aturan hanya-hasil-perolehan, biaya yang dibakar, dan log publik.
- **Yang tetap di tangan manusia.** Pembelian melalui QRIS dikreditkan setelah administrator memverifikasi pembayaran; pemberian administrator dibatasi dan publik; keputusan produk, termasuk setiap perubahan aturan perekonomian, diambil oleh manusia dan diterbitkan dalam whitepaper, yang versi 24-nya berlaku pada September 2026 [1].
- **Yang tetap di luar.** Tidak ada grafik harga, tidak ada spanduk "beli sekarang", tidak ada janji. Satu-satunya angka yang ditampilkan aplikasi di samping Rupiah adalah jumlah pembelian dan estimasi model, masing-masing dengan catatannya.

Lab G8 tidak mereproduksi kode maupun konfigurasi Profitina. Lab ini hanya memakai aturan yang diterbitkan whitepaper (pengali menjadi separuh untuk setiap pertumbuhan pengguna sepuluh kali lipat) dan masukan pembaca sendiri untuk segala hal lain. Pembaca yang ingin melatih gagasan yang sama pada produknya sendiri dapat memakai Profitina Laundry, studi kasus pengajaran dalam seri buku ini: poin loyalitas yang diperoleh pelanggan per kilogram cucian, yang laju perolehannya menjadi separuh setiap kali basis pelanggan tumbuh sepuluh kali lipat, dan yang dibakar ketika ditukar dengan setrika gratis, adalah sistem yang sama dalam bentuk mini, dengan deret geometri yang sama dan pagar pengaman yang sama.

12.15 Kesalahpahaman yang Sering Terjadi

KESALAHPAHAMAN 1: "PROFIT adalah investasi di Profitina." Bukan. PROFIT adalah koin utilitas: koin ini menghargai kontribusi dan membayar layanan di dalam platform. PROFIT tidak memberi kepemilikan atas perusahaan, tidak memberi klaim atas pendapatan, dan tidak menjanjikan imbal hasil [1].

KESALAHPAHAMAN 2: "Angka Rp1.000 adalah harga PROFIT." Angka itu adalah estimasi model yang dipakai sebagai acuan awal lantai oracle pada Persamaan (12.12). Tidak ada pasar untuk PROFIT, dan penjualan dinonaktifkan sampai mainnet dan kejelasan regulasi.

KESALAHPAHAMAN 3: "Devnet berarti tokennya palsu." Kontraknya nyata, publik, dan dapat diverifikasi; arti devnet adalah bahwa token di sana tidak memiliki nilai moneter. Itulah tujuan klaster pengembangan [6].

KESALAHPAHAMAN 4: "Emisi yang menurun berarti imbalan berhenti." Emisi terus tumbuh bersama komunitas, sebagai $u^{0.7}$ menurut Persamaan (12.5); yang menurun adalah pengali setiap pengguna. Pengguna sepuluh kali lipat berarti emisi harian lima kali lipat, bukan nol.

KESALAHPAHAMAN 5: "Pembakaran ada untuk mendorong harga naik." Pada koin utilitas, setiap pembakaran terkait dengan penggunaan nyata: pembelanjaan, transfer, pendapatan, atau penyalahgunaan yang tertangkap. Pembakaran menjaga peredaran seiring dengan kegiatan; lantai oracle bergeser sebagai akibatnya, dan lantai bukanlah harga.

KESALAHPAHAMAN 6: "Membeli banyak PROFIT menaikkan lantai." Pembelian pada harga lantai membiarkan lantai tepat di tempatnya, sebagaimana dibuktikan Persamaan (12.13).

KESALAHPAHAMAN 7: "Transfer memungkinkan saya membayar teman." Hanya PROFIT hasil perolehan organik yang dapat ditransfer, sehingga transfer membagikan pengakuan, bukan nilai Rupiah. PROFIT yang dibeli atau diberikan tidak dapat dikirim.

KESALAHPAHAMAN 8: "Lebih banyak tingkat referal hanya berarti lebih murah hati." Lebih banyak tingkat menciptakan pohon eksponensial, Persamaan (12.14), yang tidak mungkin terus tumbuh; yang bergabung belakangan membayar yang lebih awal. Satu tingkat adalah pilihan rancangan, bukan keterbatasan.

KESALAHPAHAMAN 9: "Bilangan titik mengambang cukup baik untuk saldo." Tidak: 0,01 yang dijumlahkan seratus kali tidak menghasilkan 1,0 dalam titik mengambang biner. Buku besar menghitung satuan dasar bulat, sebagaimana diperlihatkan Persamaan (12.1) dan Lab G8.

12.16 Ringkasan Bab

- Koin utilitas dipakai di dalam sebuah produk; koin ini bukan sekuritas, bukan uang elektronik, bukan janji imbal hasil, dan bukan meme. PROFIT adalah koin utilitas, dan pada rilis September 2026 merupakan token SPL di devnet Solana tanpa nilai moneter [1].
- Di Indonesia, Rupiah adalah mata uang untuk pembayaran [3], dan aset kripto diawasi OJK sejak 10 Januari 2025 [4]. Mainnet PROFIT dan bursa apa pun baru datang setelah audit token dan perizinan; penjualan dinonaktifkan sampai saat itu.
- Solana menyimpan keadaan dalam akun, menjalankan program tanpa keadaan, dan memungut biaya dalam lamport; separuh setiap biaya dasar dibakar [5]. Devnet adalah klaster latihan publiknya [6]. Per Agustus 2026, mainnet telah berjalan 30 bulan tanpa gangguan menyeluruh [7].

- Sebuah token adalah akun mint (*supply*, *decimals*, *mint authority*, *freeze authority*) ditambah akun-akun token [9]. Jumlah adalah satuan dasar bulat, ditampilkan menurut Persamaan (12.1); PROFIT memiliki 6 desimal. *Freeze authority*-nya tidak ada; *mint authority*-nya direncanakan pindah ke multisig.
- Emisi menurun: pengali Persamaan (12.2) menjadi separuh untuk setiap pertumbuhan pengguna sepuluh kali lipat, yang mereproduksi tabel whitepaper (**Tabel 12.3**). Menurut Persamaan (12.4), emisi harian adalah hukum pangkat Persamaan (12.5) dengan eksponen $\alpha \approx 0.699$: pengguna sepuluh kali memberi emisi lima kali, dan 64 kali lebih kecil daripada linear pada sejuta pengguna.
- Empat pembakaran menghapus PROFIT: *spend-burn*, *transfer-burn*, *buy-and-burn*, dan pembakaran penalti anti-bot; rancangan yang matang menargetkan pembakaran setidaknya sama dengan emisi.
- Dengan emisi tetap dan pembakaran proporsional, pasokan mengikuti Persamaan (12.6), yang solusinya melalui deret geometri Persamaan (12.7) adalah Persamaan (12.8); pasokan mendekati keadaan tunak Persamaan (12.9) dan mencapai 90% setelah t_{90} hari menurut Persamaan (12.10). Buku besar bilangan bulat sepakat dengan bentuk tertutup hingga selisih beberapa puluh mikro-PROFIT dan mengekalkan setiap satuan.
- Per pengguna, pembakaran menyusul emisi di atas u^* dari Persamaan (12.11); metode bagi dua mengonfirmasi bentuk tertutupnya.
- Lantai oracle adalah Persamaan (12.12). Emisi menurunkannya, pembakaran menaikkannya, dan pembelian pada harga lantai membiarkannya tetap, Persamaan (12.13). Lantai adalah estimasi model, bukan harga.
- Pagar pengaman: pembelian melalui QRIS diverifikasi manusia; pembelanjaan dengan empat pagar pengaman; transfer hanya untuk PROFIT hasil perolehan, dengan biaya yang dibakar, log publik, dan penahanan; pemberian dari kumpulan terbatas yang publik; referal satu tingkat, karena struktur berlapis tumbuh seperti Persamaan (12.14).
- **Tabel 12.5** memisahkan yang sudah berjalan dari peta jalan: mainnet dan Ultralight DEX adalah Fase 3, dengan gerbang audit dan perizinan.

12.17 Soal Latihan

1. **Satuan dasar.** Dengan Persamaan (12.1) dan $d = 6$, tuliskan 0,07 PROFIT dan 12,345678 PROFIT sebagai bilangan bulat mikro-PROFIT. Apa yang dilakukan `to_micro` pada **Kode 12.1** terhadap teks `"0.1234567"`, dan mengapa itu perilaku yang tepat bagi sebuah buku besar?

2. **Pengali secara manual.** Hitung $m(50{,}000)$ dari Persamaan (12.2) dengan $u_0 = 1$, lalu $E(50{,}000)$ dari Persamaan (12.3) dengan $r = 1$. Periksa jawaban Anda dengan bentuk hukum pangkat Persamaan (12.5).

3. **Selisih yang makin lebar.** Buktikan bahwa rasio emisi linear terhadap emisi menurun adalah $2^{\log_{10} u}$, lalu pakai hasil itu untuk menghitung rasionya pada 10^8 pengguna.

4. **Pasokan secara manual.** Dengan $E = 1{,}000$ PROFIT per hari dan $b = 0.01$, hitung S_1 sampai S_4 dari Persamaan (12.6) dengan $S_0 = 0$. Kemudian hitung S_4 dari Persamaan (12.8), keadaan tunak dari Persamaan (12.9), dan t_{90} dari Persamaan (12.10).

5. **Mulai dari pasokan bukan nol.** Tunjukkan bahwa untuk pasokan awal S_0 berapa pun, solusi Persamaan (12.6) adalah $S_t = S_\infty + (S_0 - S_\infty)(1-b)^t$ dengan $S_\infty = E/b$. Apa yang terjadi jika $S_0 = S_\infty$? Jika $S_0 > S_\infty$?

6. **Titik impas.** Pakai Persamaan (12.11) untuk mencari u^* bagi $\beta = 0.03$ PROFIT per pengguna per hari. Tanpa kalkulator, berapa rasio pembakaran terhadap emisi pada $10u^*$?

7. **Lantai.** Sebuah platform mainan memiliki cadangan Rp50.000.000 dan 40.000 PROFIT beredar. Hitung lantainya menurut Persamaan (12.12). Hitung lagi setelah pembakaran 4.000 PROFIT, dan secara terpisah setelah emisi 6.000 PROFIT. Kemudian tunjukkan, dengan Persamaan (12.13), apa akibat pembelian senilai Rp2.000.000 pada lantai semula.

8. **Piramida.** Dengan Persamaan (12.14), berapa orang yang dibutuhkan untuk mengisi 12 tingkat di bawah satu orang jika masing-masing merekrut 3? Jelaskan dalam dua kalimat mengapa referal satu tingkat tidak memiliki masalah ini.

9. **Jalankan lab.** Jalankan `03_supply_sim.py` (atau `SupplySim`) dengan argumen `1000 20 730`: komunitas 1.000 pengguna dan pembakaran 0,2% per hari selama dua tahun. Laporkan E, keadaan tunak, dan bagian yang tercapai setelah dua tahun, lalu jelaskan mengapa percobaan ini jauh lebih jauh dari keadaan tunaknya dibandingkan percobaan bawaan.

10. **Berjalan atau peta jalan?** Untuk setiap pernyataan, katakan apakah hal itu sudah berjalan pada rilis September 2026, ada di peta jalan, atau bukan keduanya: (a) seorang pengguna menjual 100 PROFIT seharga Rp100.000; (b) seorang pengguna membelanjakan PROFIT untuk tambahan kuota pindaian; (c) tim membekukan akun PROFIT milik pengguna yang mencurigakan; (d) seorang pengguna mengirim PROFIT hasil perolehan kepada teman; (e) PROFIT ditukar dengan SOL melalui Ultralight DEX.

12.18 Penyelesaian Soal Latihan

Penyelesaian 1. Satu PROFIT sama dengan 10^6 mikro-PROFIT, sehingga $n = a \times 10^6$. Untuk $a = 0.07$: $n = 70{,}000$. Untuk $a = 12.345678$: $n = 12{,}345{,}678$. Teks `"0.1234567"` memiliki tujuh digit di belakang titik, sehingga `to_micro` memunculkan `ValueError` dengan pesan bahwa teks itu memiliki lebih dari 6 desimal. Itu tepat karena rantai tidak dapat menyatakan desimal ketujuh: pembulatan diam-diam akan menciptakan atau memusnahkan nilai, sedangkan penolakan membuat pemanggil yang memutuskan.

Penyelesaian 2. $\log_{10} 50{,}000 = \log_{10} 5 + 4 = 0.69897 + 4 = 4.69897$, sehingga $m(50{,}000) = 0.5^{4.69897} = 2^{-4.69897} \approx 0.0385$. Maka $E = 50{,}000 \times 0.0385 \approx 1{,}925$ PROFIT per hari. Menurut Persamaan (12.5), $E = 50{,}000^{0.69897} = 10^{4.69897 \times 0.69897} = 10^{3.2844} \approx 1{,}925$. Kedua bentuk sepakat (dengan lebih banyak digit, keduanya memberi 1.925,04).

Penyelesaian 3. Emisi linear adalah u, emisi menurun adalah $u \cdot m(u)$, sehingga rasionya $1/m(u) = 2^{\log_{10} u}$, karena $m(u) = (1/2)^{\log_{10} u}$ menurut Persamaan (12.2). Pada $u = 10^8$, $\log_{10} u = 8$ dan rasionya $2^8 = 256$.

Penyelesaian 4. $q = 0.99$. $S_1 = 1{,}000$; $S_2 = 0.99 \times 1{,}000 + 1{,}000 = 1{,}990$; $S_3 = 0.99 \times 1{,}990 + 1{,}000 = 1{,}970.1 + 1{,}000 = 2{,}970.1$; $S_4 = 0.99 \times 2{,}970.1 + 1{,}000 = 2{,}940.399 + 1{,}000 = 3{,}940.399$. Bentuk tertutup: $0.99^4 = 0.96059601$, sehingga $S_4 = 1{,}000 \times (1 - 0.96059601)/0.01 = 100{,}000 \times 0.03940399 = 3{,}940.399$, sama. Keadaan tunak: $S_\infty = 1{,}000/0.01 = 100{,}000$. $t_{90} = \lceil \ln 0.1 / \ln 0.99 \rceil = \lceil -2.302585 / -0.010050 \rceil = \lceil 229.1 \rceil = 230$ hari, sesuai dengan baris terakhir keluaran lab untuk $b = 0.01$.

Penyelesaian 5. Menguraikan seperti di Subbab 12.8.2 memberi $S_t = q^t S_0 + E(1 - q^t)/b$ dengan $q = 1 - b$. Karena $E/b = S_\infty$, ini sama dengan $S_t = q^t S_0 + S_\infty - S_\infty q^t = S_\infty + (S_0 - S_\infty) q^t$. Jika $S_0 = S_\infty$, suku kedua bernilai nol untuk setiap t: pasokan tidak pernah berubah, karena

pembakaran setiap hari $bS_\infty = E$ tepat meniadakan emisi; S_∞ adalah *titik tetap* relasi rekurens itu. Jika $S_0 > S_\infty$, suku kedua positif dan menyusut dengan faktor q setiap hari, sehingga pasokan *turun* menuju S_∞: pembakaran melampaui emisi sampai keadaan tunak tercapai.

Penyelesaian 6. $\log_{10} 0.03 = -1.52288$; dikalikan $-1/\log_{10} 2 = -3.32193$ hasilnya 5.05889, sehingga $u^* = 10^{5.05889} \approx 114{,}523$ pengguna. Pada $10u^*$ pengali telah dibagi dua sekali lagi, sehingga emisi per pengguna adalah $\beta/2$ dan rasio pembakaran terhadap emisi adalah 2.

Penyelesaian 7. Lantai: 50,000,000/40,000 = Rp1,250. Setelah membakar 4.000: 50,000,000/36,000 ≈ Rp1,388.89, lebih tinggi. Setelah mengemisikan 6.000 (kembali dari 40.000): 50,000,000/46,000 ≈ Rp1,086.96, lebih rendah. Pembelian Rp2.000.000 pada Rp1.250 memberi 2,000,000/1,250 = 1,600 PROFIT; lantai barunya 52,000,000/41,600 = 1,250, tidak berubah, sebagaimana dijamin Persamaan (12.13).

Penyelesaian 8. $3 \times (3^{12} - 1)/2 = 3 \times 531{,}440/2 = 797{,}160$ orang. Referal satu tingkat hanya membayar untuk teman yang diundang langsung oleh seseorang, dibatasi 50, dan membayar kedua pihak secara setara, sehingga tidak ada yang dibayar atas rekrutan orang lain dan tidak ada yang bergantung pada pohon yang terus membesar.

Penyelesaian 9. Program mencetak:

```
users=1,000  emission E=125.000000 PROFIT/day  burn b=0.0020/day  steady state E/b=62,500
 day      ledger supply        closed form gap (micro)
   0          0.000000           0.000000           0
  30      3,643.252892       3,643.252878          14
  90     10,305.020319      10,305.020277          42
 180     18,910.945544      18,910.945467          77
 365     32,402.696669      32,402.696532         137
 730     48,006.437382      48,006.437184         198
conservation: supply == emitted - burned on every day: True
after 730 days: emitted 91,250, burned 43,244
supply as a share of the steady state (each # = 2.5%):
   0 |                                        |   0.0%
  73 |#####                                   |  13.6%
 146 |##########                              |  25.3%
 219 |###############                         |  35.5%
 292 |###################                     |  44.3%
 365 |#######################                 |  51.8%
 438 |###########################             |  58.4%
 511 |###############################         |  64.0%
 584 |###################################     |  68.9%
 657 |#######################################   |  73.2%
 730 |###########################################|  76.8%
how the burn rate shapes the curve:
  b=0.0020: steady state       62,500 PROFIT, 90% reached on day 1151
  b=0.0050: steady state       25,000 PROFIT, 90% reached on day 460
  b=0.0100: steady state       12,500 PROFIT, 90% reached on day 230
```

Dengan 1.000 pengguna, $E = 125$ PROFIT per hari (baris pertama **Tabel 12.3**); dengan $b = 0.002$, keadaan tunaknya 125/0.002 = 62,500; setelah 730 hari pasokannya 48.006, yaitu 76,8% dari keadaan tunak. Percobaan bawaan mencapai 97,4% dalam waktu yang sama karena laju pembakarannya 0,5%, bukan 0,2%. Menurut Persamaan (12.10), skala waktunya ditentukan oleh b saja: dengan $b = 0.002$, mencapai 90% memerlukan 1.151 hari, lebih dari tiga tahun, berapa pun emisinya.

Penyelesaian 10. (a) Tidak berjalan dan tidak tersedia: penjualan dinonaktifkan sampai mainnet dan kejelasan regulasi. (b) Berjalan. (c) Mustahil: *freeze authority*-nya tidak ada, sehingga tidak seorang pun dapat membekukan akun PROFIT; tindakan keamanan tingkat akun di aplikasi (seperti moderasi) tidak pernah menyentuh PROFIT pengguna. (d) Berjalan, dengan biaya yang

dibakar, log publik, dan penahanan jika teman itu belum memverifikasi surel. (e) Peta jalan: Fase 3, setelah audit token dan perizinan.

Daftar Pustaka

[1] Profitina, *Profitina Whitepaper — Public Version v24* (ID/EN), Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[2] Profitina, *Profitina — Dasbor GEO*, PROFIT area and plan cards, release 0.14.5-testnet. [Online]. Available: https://app.profitina.com (accessed Sep. 24, 2026).

[3] Republik Indonesia, *Undang-Undang Nomor 7 Tahun 2011 tentang Mata Uang* (Law No. 7 of 2011 on Currency), 2011.

[4] Herbert Smith Freehills Kramer, "OJK assumes regulatory oversight of digital financial assets," May 15, 2025. [Online]. Available: https://www.hsfkramer.com/notes/fintech/2025-posts/ojk-assumes-regulatory-oversight-of-digital-financial-assets

[5] Solana Foundation, "Fees," *Solana Documentation*. [Online]. Available: https://solana.com/docs/core/fees (accessed Sep. 24, 2026).

[6] Solana Foundation, "Clusters and Public RPC Endpoints," *Solana Documentation*. [Online]. Available: https://solana.com/docs/references/clusters (accessed Sep. 24, 2026).

[7] Solana Compass, "Solana Mainnet 30-Month Uptime Streak: Firedancer, QUIC, and Priority Fees," Aug. 10, 2026. [Online]. Available: https://solanacompass.com/news/solana-marks-30-consecutive-months-without-a-network-wide-outage

[8] Solana Foundation, "Solana Changelog: September 18, 2026," Sep. 2026. [Online]. Available: https://solana.com/news/solana-changelog-september-18-2026

[9] Solana Foundation, "Tokens on Solana," *Solana Documentation*. [Online]. Available: https://solana.com/docs/tokens (accessed Sep. 24, 2026).

[10] Metaplex Foundation, "Token Metadata," *Metaplex Documentation*. [Online]. Available: https://www.metaplex.com/docs/smart-contracts/token-metadata (accessed Sep. 24, 2026).

[11] IEEE, *IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2019, Jul. 2019.

[12] Squads, "Welcome to Squads Multisig," *Squads Documentation*. [Online]. Available: https://docs.squads.so/main (accessed Sep. 24, 2026).

[13] V. Buterin, E. Conner, R. Dudley, M. Slipper, I. Norden, and A. Bakhta, "EIP-1559: Fee market change for ETH 1.0 chain," Ethereum Improvement Proposals, no. 1559, Apr. 2019. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1559

[14] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed. Reading, MA, USA: Addison-Wesley, 1994, ch. 2.

[15] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical Analysis*, 10th ed. Boston, MA, USA: Cengage Learning, 2016, sec. 2.1.

[16] ANTARA News, "QRIS reaches 65.77 mln users, accepted by 44.86 mln merchants: BI," Aug. 21, 2026. [Online]. Available: https://en.antaranews.com/news/428168/qris-reaches-6577-mln-users-accepted-by-4486-mln-merchants-bi

[17] Republik Indonesia, *Undang-Undang Nomor 7 Tahun 2014 tentang Perdagangan* (Law No. 7 of 2014 on Trade), art. 9, 2014.

[18] Solana Foundation, "Solana Explorer," devnet cluster. [Online]. Available: https://explorer.solana.com/?cluster=devnet (accessed Sep. 24, 2026).

Bab 13

Profitina Group dan Jalan ke Depan

Setelah menyelesaikan bab ini, Anda akan mampu menggambarkan Profitina Group secara utuh: sebuah payung yang menaungi produk-produk bersaudara yang berbagi enam lapisan bersama, masing-masing produk dengan tugasnya sendiri. Anda akan mengetahui apa yang dikerjakan Profitina Capital dengan statistik risiko seperti *value at risk*, *expected shortfall*, *drawdown* maksimum, dan konsentrasi, dan Anda akan mampu menghitung masing-masing secara manual pada contoh sepuluh hari. Anda akan dapat mengikuti Profitina TTM ketika mengubah satu paragraf teks menjadi film pendek, dan Profitina AI ketika mengarahkan sebuah permintaan antara model di mesin milik pengguna dan penyedia *cloud*, serta memahami mengapa Profitina Laundry, sebuah usaha binatu fiktif, menjadi tempat berlatih para pembaca seri buku ini. Anda akan dapat menyebutkan tiga prinsip yang mengikat semua produk itu (AI *local-first*; aplikasi menyarankan dan pengguna memutuskan; tes sebagai gerbang) dan menunjukkan di mana masing-masing muncul. Anda akan membangun Lab G9 dalam Python dan Java: pemeriksa yang bertanya kepada katalog publik OpenAlex seberapa lengkap dan konsisten rekaman seorang peneliti, yang menolak mengarang hasil untuk nama yang tidak dapat ditemukannya, dan yang tidak pernah memeringkat siapa pun. Anda juga akan melihat apa yang menanti di jalan ke depan, dengan garis tegas antara yang sudah terjadi dan yang baru direncanakan. Bab ini, sekaligus buku ini, ditutup dengan sepatah kata penutup.

13.1 Posisi Kita Sejauh Ini

Bab 12 menuntaskan perjalanan menjelajahi perekonomian Profitina. PROFIT ternyata adalah koin utilitas dengan seperangkat aturan yang kecil dan tepat: token devnet dengan enam desimal tanpa *freeze authority*, pengali emisi yang menjadi separuh untuk setiap pertumbuhan pengguna sepuluh kali lipat, empat pembakaran, lantai Rupiah yang tidak dapat digeser oleh pembelian pada harga lantai, dan sebuah tabel yang memisahkan apa yang sudah berjalan dari apa yang menunggu audit dan izin. Sebelumnya, Bab 2 sampai 11 menelusuri produknya sendiri: pergeseran ke mesin jawaban dan GEO, pindaian beserta perbaikannya, komunitas, keamanan, AI lokal, siklus umpan balik, model bisnis, arsitektur dengan gerbang rilisnya, situs pemasaran, dan privasi sejak rancangan.

Semua itu adalah satu produk, app.profitina.com, dengan situs pemasarannya profitina.com. Namun Profitina juga nama sebuah keluarga kecil produk, dibangun oleh orang-orang yang sama dengan kebiasaan yang sama. Sebagian produk itu berbagi kode satu sama lain; salah satunya ada semata-mata untuk mengajar. Bab terakhir ini mundur selangkah untuk melihat keluarga itu, prinsip-prinsip yang menyatukannya, dan jalan ke depan.

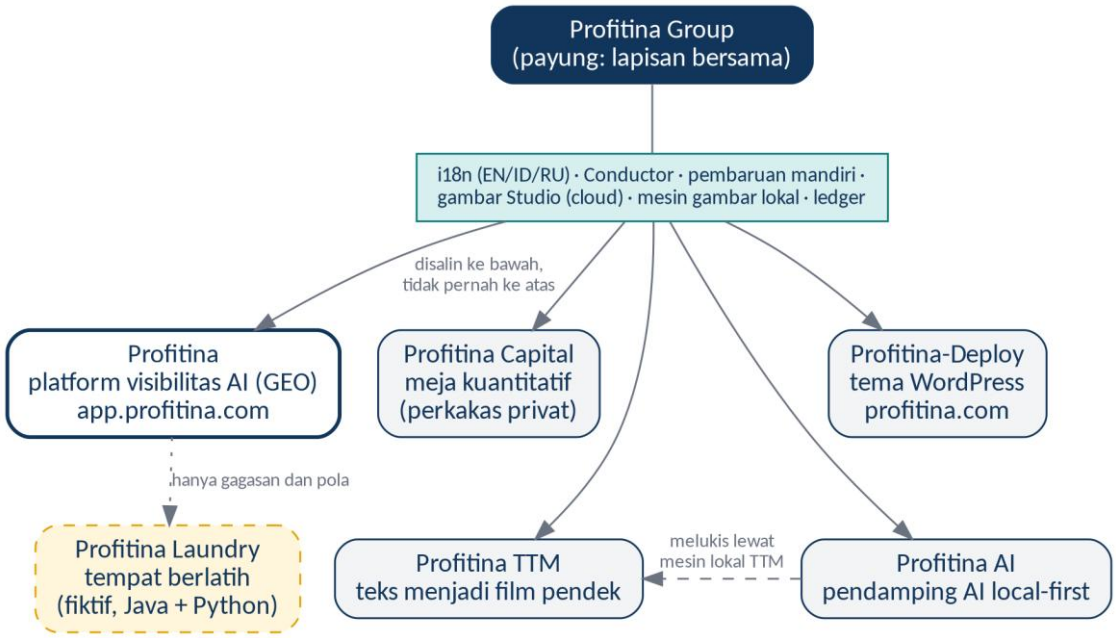
Seperti bab lainnya, bab ini memiliki lab, dan lab yang pas pula. Researcher Visibility Report, salah satu fitur terbaru Profitina (Bab 3), mengukur seberapa lengkap dan mudah ditemukannya rekaman seorang ilmuwan di OpenAlex, katalog terbuka hasil riset dunia. Lab G9 membangun versi kecil yang transparan dari gagasan yang sama. Lab ini cocok menjadi lab terakhir karena merangkum begitu banyak benang buku ini: API publik yang dipakai dengan santun, kegagalan yang jujur ketika jaringan tidak tersedia, kontrol negatif yang menolak mengarang hasil,

normalisasi string sebagai alat kualitas data, dan laporan yang membantu seseorang memperbaiki sesuatu tanpa memeringkatnya terhadap siapa pun.

13.2 Satu Payung, Beberapa Rumah

13.2.1 Bentuk kelompoknya

Bayangkan sebuah kompleks keluarga di desa Jawa: beberapa rumah mengelilingi satu halaman, setiap rumah tangga dengan dapur dan pekerjaannya sendiri, tetapi satu sumur, satu gerbang, dan satu aturan tentang halaman. *Profitina Group* tertata seperti itu. **Gambar 13.1** memperlihatkannya.



Gambar 13.1 Profitina Group. Lapisan bersama berada di puncak dan disalin ke bawah ke setiap produk, tidak pernah kembali ke atas. Profitina Laundry, tempat berlatih, hanya meminjam gagasan dan pola, tidak pernah kode atau data.

Di puncak berdiri kelompok itu sendiri, satu repositori yang menyimpan potongan-potongan yang dibutuhkan setiap produk. Di bawahnya ada produk-produknya: Profitina, platform visibilitas AI yang menjadi pokok buku ini; Profitina-Deploy, tema WordPress untuk profitina.com (Bab 10); Profitina Capital, sebuah meja kuantitatif; Profitina TTM, yang mengubah teks menjadi film pendek; dan Profitina AI, aplikasi pendamping AI yang *local-first*. Di sisi, dengan bingkai putus-putus karena ia milik ruang kelas dan bukan milik produksi, ada Profitina Laundry.

13.2.2 Enam lapisan bersama

Kelompok ini menyimpan enam lapisan yang dibutuhkan lebih dari satu produk. Masing-masing adalah balok bangunan yang boros, dan berisiko, bila ditulis berkali-kali.

1. **Bahasa antarmuka.** Kamus antarmuka berbahasa Inggris, Indonesia, dan Rusia, dengan kunci yang sama di setiap bahasa. Profitina sendiri menjaga kamus ID dan EN-nya dengan tes yang gagal bila sebuah kunci ada di satu bahasa tetapi tidak di bahasa lain: pada September 2026, 1.531 kunci kamus harus simetris di antara kedua bahasa.

2. **Sebuah "Conductor".** Komponen yang memutuskan model AI mana yang sebaiknya menangani sebuah pekerjaan, berdasarkan jenis tugas, model yang tersedia, dan kesehatan terkini model-model itu.
3. **Pembaruan mandiri.** Mekanisme yang membuat sebuah produk desktop tahu bahwa ada versi yang lebih baru lalu memasangnya dengan aman.
4. **Mesin gambar Studio (*cloud*).** Cara yang seragam untuk meminta gambar kepada beberapa layanan gambar *cloud*.
5. **Mesin gambar lokal bawaan.** Antarmuka yang sama, dijawab oleh model difusi yang berjalan di kartu grafis milik pengguna.
6. **Sebuah "ledger" penyedia yang sedang beristirahat.** Memori bersama yang kecil tentang penyedia AI mana yang untuk sementara beristirahat setelah galat autentikasi, batas kuota, atau gangguan layanan, sehingga pengalaman buruk satu produk dengan sebuah penyedia menyelamatkan produk lain dari mengulanginya.

Lapisan keenam layak mendapat satu kalimat teori, karena ia adalah pola buku teks yang menyamar. Ketika layanan jarak jauh gagal, klien yang naif segera mencoba lagi, gagal lagi, mencoba lagi, dan mengubah satu gangguan menjadi badai permintaan yang sia-sia. *Circuit breaker* (pemutus sirkuit) justru mencatat kegagalan itu dan berhenti memanggil layanan tersebut untuk sementara; setelah masa jeda, ia meloloskan satu permintaan percobaan, dan hanya jika percobaan itu berhasil lalu lintas normal dilanjutkan [1]. Ledger itu adalah pemutus sirkuit yang ingatannya dibagi antarproduk.

13.2.3 Satu aturan untuk berbagi

Kode bersama menimbulkan bahaya baru: dua salinan yang makin lama makin berbeda sampai tak seorang pun tahu mana yang benar. Kelompok ini menjawabnya dengan satu aturan ketat: **ubah di Group, salin ke unit, tidak pernah sebaliknya.** Perbaikan pada lapisan bersama dilakukan sekali, di puncak, lalu disalin ke setiap produk; sebuah produk tidak pernah menambal salinan lapisan bersamanya sendiri dan meninggalkan yang lain. Aturan ini sama dengan yang dijumpai Bab 11 dalam bentuk lain untuk Kebijakan Privasi: "hapus duplikasi, jangan menyinkronkannya". Ketika duplikasi tidak dapat dihapus, seperti kode yang harus dikirim di dalam beberapa produk terpisah, setidaknya duplikasi itu harus mengalir ke satu arah saja.

Dua kebiasaan lain menjaga kelompok ini tetap jujur. Pekerjaan dilakukan di dua komputer, dengan rutinitas ketat untuk menarik versi terbaru sebelum mulai dan menyinkronkan setelah selesai, sehingga tidak ada mesin yang menyimpan perubahan yang belum dilihat mesin lainnya. Selain itu, setiap *commit* melewati *pemindai rahasia* yang menolak merekam apa pun yang tampak seperti kunci, token, atau kata sandi. Kebiasaan kedua ini lebih penting daripada kelihatannya: rahasia yang sampai ke riwayat versi harus dianggap telah bocor, betapa pun cepatnya dihapus sesudahnya.

> **WAWASAN KUNCI.** Sebuah keluarga produk tetap padu berkat beberapa aturan yang mudah dinyatakan dan sulit dilanggar: satu arah untuk kode bersama, satu rutinitas untuk dua mesin, dan satu gerbang yang menghentikan rahasia sebelum sempat tertulis.

13.3 Profitina Capital: Meja Kuantitatif

13.3.1 Apa itu

Profitina Capital adalah meja eksekusi kuantitatif milik kelompok ini: aplikasi lokal untuk mengelola portofolio kripto di berbagai bursa, pinjaman, *staking*, dan kolam likuiditas. Ini adalah perkakas privat. Ia bukan produk publik, tidak mengelola uang orang lain, dan tidak ada bagian subbab ini yang merupakan nasihat keuangan. Ia muncul di buku ini karena matematika di dalamnya adalah matematika mata pelajaran metode numerik atau algoritma yang biasa, diterapkan dengan disiplin.

Rilis September 2026-nya, versi 3.8, menambahkan sebuah *meja risiko* dengan *value at risk* dan *expected shortfall* pada 95% dan 99%, volatilitas, *drawdown* maksimum, kepekaan (beta) terhadap harga BTC dan SOL, korelasi, konsentrasi yang diukur dengan indeks Herfindahl–Hirschman, skenario tekanan (*stress scenario*), dan perkiraan *impermanent loss* untuk kolam likuiditas. Rilis itu juga menambahkan atribusi kinerja (imbal hasil berbobot waktu dan berbobot uang, rasio Sharpe, Sortino, dan Calmar), laporan bulanan, analisis biaya transaksi (*slippage* dan *implementation shortfall*), batas kebijakan yang membunyikan peringatan bila dilanggar, dan pemeriksaan kesehatan data yang menjadi masukan semuanya.

Daftarnya panjang. Sisa subbab ini mengambil lima butir darinya dan menghitung masing-masing secara manual pada satu contoh kecil, karena masing-masing adalah rumus yang dapat dipahami setiap pembaca dan tidak ada yang ajaib.

13.3.2 Lima ukuran pada sepuluh hari

Ambil sebuah portofolio yang imbal hasil hariannya selama sepuluh hari, dalam persen, adalah

$$r = (1.2,\ -0.8,\ 0.5,\ -2.1,\ 0.9,\ -0.3,\ 1.5,\ -1.4,\ 0.7,\ -0.6)$$

Value at risk. *Value at risk* pada tingkat keyakinan c menjawab pertanyaan "kerugian berapa yang hanya terlampaui pada bagian $(1-c)$ hari-hari terburuk?". Dengan data historis, urutkan imbal hasil dari yang terburuk ke yang terbaik lalu baca kerugian pada kuantil $(1-c)$, yaitu q_{1-c}:

$$\mathrm{VaR}_c = -\,q_{1-c}(r) \tag{13.1}$$

Setelah diurutkan, imbal hasil kita adalah $-2.1, -1.4, -0.8, -0.6, -0.3, 0.5, 0.7, 0.9, 1.2, 1.5$. Pada $c = 80\%$, 20% terburuk dari sepuluh hari adalah dua hari, dan kuantilnya terletak pada imbal hasil terburuk kedua, -1.4. Jadi $\mathrm{VaR}_{80\%} = 1.4\%$: pada empat dari lima hari, kerugiannya tidak lebih buruk dari 1,4%. (Meja sungguhan memakai 95% atau 99% dan ratusan hari; sepuluh hari dan 80% menjaga aritmetikanya tetap terlihat.)

Expected shortfall. VaR menyatakan di mana hari-hari buruk dimulai, bukan seberapa buruk hari-hari itu. *Expected shortfall*, disebut juga *conditional value at risk* (CVaR), adalah rata-rata kerugian pada hari-hari di atau melampaui ambang VaR [2]:

$$\mathrm{CVaR}_c = -\,\mathbb{E}[\,r \mid r \le q_{1-c}(r)\,] \tag{13.2}$$

Di sini dua hari terburuk adalah -2.1 dan -1.4, sehingga $\mathrm{CVaR}_{80\%} = (2.1 + 1.4)/2 = 1.75\%$. *Expected shortfall* selalu setidaknya sebesar VaR, dan merupakan angka yang lebih jujur ketika kerugian memiliki ekor panjang.

Drawdown maksimum. Mulailah portofolio pada 100 lalu gandakan imbal hasilnya secara majemuk: 100; 101,20; 100,39; 100,89; 98,77; 99,66; 99,36; 100,85; 99,44; 100,14; 99,54. *Drawdown* pada setiap saat adalah penurunan dari nilai tertinggi yang pernah terlihat sejauh itu, dan *drawdown maksimum* adalah penurunan terbesar semacam itu:

$$\mathrm{MDD} = \max_{t} \frac{\max_{s \le t} V_s - V_t}{\max_{s \le t} V_s} \tag{13.3}$$

Puncaknya 101,20 setelah hari ke-1; titik terendah sesudahnya adalah 98,77 setelah hari ke-4; jadi $\mathrm{MDD} = (101.20 - 98.77)/101.20 \approx 2.40\%$. *Drawdown* adalah angka yang paling cocok dengan cara orang benar-benar merasakan risiko: bukan goyangan rata-rata, melainkan rentang terburuk dari puncak ke lembah.

Rasio Sharpe. *Rasio Sharpe* membagi rata-rata imbal hasil berlebih dengan simpangan bakunya, mengukur imbal hasil per satuan keragaman [3]:

$$\mathrm{SR} = \frac{\bar{r} - r_f}{\sigma_r} \tag{13.4}$$

Dengan tingkat bebas risiko $r_f = 0$ demi kesederhanaan, rata-ratanya $\bar{r} = -0.04\%$ per hari dan simpangan baku sampelnya $\sigma_r \approx 1.187\%$, sehingga $\mathrm{SR} \approx -0.034$ per hari. Rasio Sharpe yang negatif menyatakan, dengan benar, bahwa sepuluh hari ini sedikit merugi sambil banyak berguncang.

Konsentrasi. Jika sebuah portofolio memegang aset dengan bobot $w_1, \ldots, w_n$ (bagian dari nilainya yang jumlahnya 1), *indeks Herfindahl–Hirschman* adalah jumlah kuadrat bobotnya:

$$\mathrm{HHI} = \sum_{i=1}^{n} w_i^2 \tag{13.5}$$

Untuk bobot 0.5,0.3,0.2: $\mathrm{HHI} = 0.25 + 0.09 + 0.04 = 0.38$. Indeks ini bernilai 1 untuk portofolio yang hanya berisi satu aset dan $1/n$ untuk n bobot yang sama, sehingga kebalikannya $1/0.38 \approx 2.6$ dapat dibaca sebagai “portofolio ini berperilaku seperti 2,6 aset berbobot sama”, lebih sedikit daripada tiga aset yang dipegangnya.

> **PERINGATAN.** Angka-angka ini menggambarkan sepuluh hari rekaan. Angka-angka itu ada untuk mengajarkan rumus, bukan untuk merekomendasikan aset, strategi, atau produk apa pun. Statistik risiko menggambarkan masa lalu; statistik itu tidak menjanjikan apa pun tentang masa depan.

13.3.3 Apa yang diajarkan Capital

Bagi mahasiswa, Capital adalah kumpulan latihan yang kebetulan nyata. Pengurutan dan kuantil adalah jantung VaR dan *expected shortfall*. Satu kali penelusuran dengan maksimum berjalan menghitung *drawdown* maksimum dalam waktu linear, sebuah algoritma kecil yang memuaskan. Korelasi dan beta adalah masalah kuadrat terkecil. Skenario tekanan adalah perhitungan ulang “bagaimana jika” atas masukan yang diubah. Batas kebijakan dengan peringatan adalah pola yang sama dengan pemantauan Profitina sendiri (Bab 9): ukur, bandingkan dengan ambang yang dipilih pengguna, lalu beri tahu manusia.

Dua butir dalam daftar itu kurang dikenal dan masing-masing layak mendapat satu paragraf. *Analisis biaya transaksi* bertanya berapa sebenarnya biaya sebuah perdagangan dibandingkan harga pada saat keputusan diambil. Selisihnya, yang disebut *implementation shortfall*, mencakup biaya yang dibayar, *slippage* antara harga yang diharapkan dan harga eksekusi, serta biaya bagian pesanan yang tidak terisi. Ini adalah versi perdagangan dari pelajaran yang dipetik setiap insinyur: rencana dan hasil adalah dua pengukuran yang berbeda, dan hanya yang kedua yang nyata. *Pemeriksaan kesehatan data* bertanya, sebelum statistik apa pun dihitung, apakah harga dan saldo yang menjadi masukannya segar, lengkap, dan masuk akal: hari yang hilang, kuotasi yang basi, atau saldo yang melonjak seribu kali lipat ditandai, bukan dirata-ratakan diam-diam. Angka risiko yang dihitung dari data yang rusak lebih buruk daripada tanpa angka, karena ia tampak seperti pengetahuan. Prinsipnya sama dengan kontrol negatif Lab G9 dan "nol yang diam" di Subbab 13.4.3: ketika masukan tidak dapat dipercaya, katakan demikian alih-alih menghasilkan jawaban yang meyakinkan.

13.4 Profitina TTM: Dari Teks ke Film Pendek

13.4.1 Alur pemrosesannya

Profitina TTM (*text to multimedia*; versi 2.8.1 pada September 2026) menerima sebuah teks atau skrip dan menghasilkan film pendek. Ia bekerja sebagai alur tahapan (*pipeline*), masing-masing dengan masukan dan keluaran yang jelas.

1. **Perencana.** Memecah teks menjadi adegan dan memutuskan kira-kira berapa lama masing-masing berlangsung.
2. **Sutradara.** Memberi film itu bentuk. Ia menata adegan-adegan di sepanjang enam ketukan cerita (*hook, setup, rise, turn, climax, landing*), mendeteksi suasana setiap adegan, dan memilih resep pencahayaan dan lensa yang sesuai.
3. **Pelukisan.** Menghasilkan satu gambar untuk setiap bidikan.
4. **Suara.** Membacakan narasi.
5. **Musik dan suasana.** Menghasilkan musik dan suara latar secara prosedural, dari aturan dan bukan dari rekaman.
6. **Render.** Merakit semuanya menjadi video dengan ffmpeg, perangkat sumber terbuka baku untuk audio dan video [4].

Enam ketukan itu adalah gagasan bercerita yang lama dalam busana modern. Penonton ditangkap oleh sebuah kait (*hook*), diperkenalkan oleh pembukaan (*setup*), dibawa oleh ketegangan yang menanjak (*rise*), dikejutkan oleh belokan (*turn*), ditahan di puncak (*climax*), lalu diturunkan dengan lembut pada pendaratan (*landing*). Alur yang mengenal ketukan-ketukan ini dapat menempatkan bidikan dekat ketika cerita berbelok dan bidikan lebar ketika cerita mendarat.

13.4.2 Melukis di dua jalur

Sejak versi 2.8, TTM melukis secara *hibrida*. Mesin gambar *cloud* dicoba lebih dulu, dan mesin difusi lokal bawaan yang berjalan di kartu grafis laptop itu sendiri, sebuah RTX 4060, ikut bekerja setiap kali hasilnya setidaknya sama baik dan tidak lebih lambat. Mesin lokal itu memakai model difusi cepat dengan sedikit langkah seperti SDXL-Lightning, yang menyuling sebuah model gambar besar sehingga dapat menghasilkan gambar dalam segelintir langkah alih-alih puluhan [5]. Aturan "ikut bekerja bila setidaknya sama baik dan tidak lebih lambat" semestinya terdengar akrab: aturan ini sama dengan gerbang kualitas Content Engine Profitina (Bab 6), yang hanya

menyajikan draf AI bila draf itu mengalahkan templat. Di kedua tempat, sebuah model memperoleh tempatnya melalui pengukuran.

TTM juga memiliki "kamera" 2,5D. Dari satu gambar diam, ia memperkirakan kedalaman dengan Depth Anything V2, sebuah model estimasi kedalaman monokular [6], lalu menggerakkan kamera maya sedikit sehingga benda dekat bergeser lebih jauh daripada benda jauh. Mata membaca *paralaks* ini sebagai kedalaman, dan gambar diam seolah-olah bernapas.

13.4.3 Gagal dengan lantang

Keputusan rancangan terpenting di TTM sama sekali bukan tentang gambar. Versi-versi awalnya, ketika sebuah mesin pelukis gagal, diam-diam merender gradien warna yang mulus di tempat gambar yang hilang, dan film pun selesai "dengan sukses" dengan bidikan-bidikan kosong di dalamnya. TTM kini **gagal dengan lantang**: jika sebuah bidikan tidak dapat dilukis, prosesnya berhenti atau melaporkan dengan tepat bidikan mana yang hilang dan mengapa, alih-alih merender pengganti yang diam. Sejarah rekayasa Profitina sendiri mencatat pelajaran yang sama sebagai "nol yang diam": sebuah `try`/`except` di sekitar penghitungan yang gagal pernah hampir menampilkan angka 0 yang meyakinkan, dan sejak itu aturannya adalah galat harus lantang. Hasil yang tampak selesai padahal belum lebih buruk daripada galat yang jujur, karena tak seorang pun tahu ada yang perlu diperbaiki.

13.5 Profitina AI: Pendamping AI Local-First

13.5.1 Apa yang dikerjakannya

Profitina AI (versi 1.7.2 pada September 2026) adalah aplikasi pendamping AI yang berjalan di peramban dan sebagai aplikasi desktop Windows. Pengguna bercakap dengannya; di balik percakapan itu, Conductor dari Subbab 13.2.2 memutuskan apakah sebuah permintaan dikirim ke model yang berjalan lokal melalui Ollama (misalnya model Qwen) atau ke penyedia *cloud*. Aplikasi ini mendukung utas percakapan, dan agen yang bekerja bersama pada satu tugas; perkakas untuk berkas, proses, dan dokumen Office (cari dan ganti di dalam `.docx`, membangun daftar isi, mengekspor ke PDF); tugas terjadwal; bahasa antarmuka kelompok; dan sebuah Studio untuk gambar yang dapat melukis melalui mesin lokal TTM.

Local-first berarti model lokal adalah bawaan dan *cloud* adalah pengecualian yang dipilih pengguna. Artinya bukan "hanya lokal". Perbedaan itulah yang memisahkan Profitina AI dari Profitina, produk buku ini: Profitina AI boleh memanggil penyedia *cloud* bila penggunanya memintanya, sedangkan **Profitina, platform GEO, sama sekali tidak memakai AI *cloud***. Model-modelnya berjalan di servernya sendiri, melalui Ollama, dalam dua jalur di Bab 6, dan data pengguna tidak pernah meninggalkan server untuk pemrosesan AI. Kedua produk itu bersaudara, dan Profitina AI bukan bagian dari app.profitina.com.

Bagaimana sebuah *conductor* memutuskan? Pada tingkat yang digambarkan buku ini, ia menimbang tiga pertanyaan sederhana untuk setiap permintaan. *Bolehkah data ini meninggalkan mesin?* Jika pengguna belum memilih penyedia *cloud* untuknya, jawabannya tidak, dan permintaan itu tetap lokal apa pun keadaan lainnya. *Model tersedia mana yang cocok untuk jenis tugas ini?* Klasifikasi singkat membutuhkan model kecil yang cepat; dokumen panjang membutuhkan model yang lebih besar; gambar membutuhkan mesin gambar. *Mana di antaranya yang sehat saat ini?* Di sinilah ledger bersama dari Subbab 13.2.2 membuktikan gunanya: penyedia yang baru saja mengembalikan galat kuota atau autentikasi ditandai sedang beristirahat

dan dilewati sampai masa jedanya berakhir, sehingga pengguna tidak perlu menunggu kegagalan yang sudah dialami produk lain. Tugas terjadwal, seperti ringkasan mingguan sebuah folder dokumen, melewati tiga pertanyaan yang sama pada saat tugas itu dijalankan, bukan pada saat dibuat, karena jawabannya dapat berubah di antara kedua saat itu.

13.5.2 Tidak pernah berhenti diam-diam

Setelah versi 1.7.0, Profitina AI memakai semboyan yang terdengar mirip TTM: *tidak pernah berhenti diam-diam*. Setiap giliran percakapan diakhiri dengan kartu eksplisit yang menyatakan apa yang telah dan belum dikerjakan; pemutus gelung menghentikan agen yang mengulang dirinya sendiri; dan keluaran parsial apa pun yang ada selalu dikembalikan, bahkan ketika sebuah tugas tidak dapat diselesaikan. Pengguna tidak boleh dibiarkan menatap ikon berputar sambil bertanya-tanya apakah ada yang sedang terjadi.

Letakkan ketiga produk bersaudara itu berdampingan dan polanya tidak dapat dikelirukan. Profitina berkata “belum yakin” ketika tidak dapat mendeteksi kategori, dan “tidak ada model yang tersedia” ketika model lokal mati (Bab 3 dan 6). TTM gagal dengan lantang alih-alih melukis gradien. Profitina AI tidak pernah berhenti diam-diam. Ini bukan tiga kebetulan. Ini satu prinsip: **katakan kepada pengguna apa yang sebenarnya terjadi**, yang akan dibahas lagi di subbab berikutnya.

13.6 Profitina Laundry: Tempat Berlatih

Profitina Laundry adalah sebuah usaha binatu kecil yang fiktif, beserta aplikasi yang menjalankannya. Ia ada semata-mata untuk mengajar. Buku Pemrograman Web dan Pemrograman Berbasis Kerangka Kerja dalam seri ini membangunnya bab demi bab, dalam Java dan Python, dan bahan yang direncanakan di sekitarnya mencakup kasus penggunaan, diagram hubungan entitas, ratusan transaksi contoh, dan README untuk setiap bab.

Mengapa mengarang sebuah binatu padahal ada platform nyata di depan mata? Tiga alasan.

Alasan pertama adalah keamanan. Produk nyata memiliki pengguna nyata, data nyata, dan rahasia nyata. Pembaca yang berlatih padanya dapat membocorkan sesuatu, merusak sesuatu, atau tergoda menyalin apa yang tidak boleh disalin. Binatu fiktif tidak memiliki apa pun untuk dibocorkan: pelanggannya tidak ada, pesanannya contoh, dan kodenya milik pembaca.

Alasan kedua adalah ukuran. Sebuah binatu cukup kecil untuk dipahami sepenuhnya: pelanggan membawa pakaian, toko menimbangnya, mencuci, menyetrika, menagih per kilogram, lalu menyerahkannya kembali. Setiap konsep aplikasi web (akun, pesanan, harga, status, tanda terima, laporan) muncul di dalamnya tanpa ribuan detail sebuah platform produksi.

Alasan ketiga adalah kejujuran tentang apa yang dibagi. Profitina Laundry tidak memiliki hubungan dengan kode atau data produk yang sebenarnya. Yang dipinjamnya dari Profitina adalah gagasan dan pola: tugas terjadwal, kuota harian yang diatur ulang pukul 00.00 WIB, mesin keadaan untuk sebuah pesanan, log persetujuan, blok data terstruktur untuk usaha lokal. Beberapa lab di buku ini menunjuk kepadanya tepat karena alasan itu: generator JSON-LD di Bab 2 menggambarkan binatu itu, paket dan kuota di Bab 8 dapat saja milik layanan kilat sebuah binatu, dan Bab 12 menyarankan poin loyalitas binatu yang laju perolehannya menjadi separuh ketika basis pelanggan tumbuh sepuluh kali lipat.

Ada pula alasan keempat yang lebih senyap: kestabilan. Platform produksi berubah di bawah kaki pembacanya; basis kode Profitina sendiri menerima 391 *commit* antara 3 Juli dan akhir

September 2026, dan bab yang menggambarkan bagian dalamnya baris demi baris akan usang sebelum sempat dicetak. Tempat berlatih hanya berubah ketika buku yang memakainya berubah, sehingga soal latihan yang ditulis hari ini tetap berlaku ketika seorang mahasiswa menjumpainya tahun depan. Kestabilan itulah yang memungkinkan sebuah buku ajar dan sebuah produk hidup berjalan berdampingan tanpa saling mengganggu.

> **DEFINISI.** *Tempat berlatih* (*teaching ground*) adalah sistem fiktif yang sengaja dibuat kecil, tempat pembelajar melatih teknik-teknik nyata tanpa pengguna nyata, data nyata, atau risiko nyata. Ia berbagi gagasan dengan sistem produksi, tidak pernah kode atau data.

13.7 Prinsip-Prinsip yang Mengikat

Lima produk, tiga bahasa antarmuka, dua komputer, satu tempat berlatih. Yang menjadikan semuanya satu keluarga bukanlah logo, melainkan seperangkat kecil prinsip yang muncul, dengan busana berbeda, di setiap produk. **Gambar 13.2** menggambar tiga prinsip yang paling penting sebagai satu siklus.

Gambar 13.2 Tiga prinsip sebagai satu siklus. Data masuk ke model lokal; model memberi saran beserta alasannya; manusia memutuskan; tes berdiri di antara setiap perubahan dan rilisnya, yang baru dianggap selesai setelah pemeriksaan kesehatan mengonfirmasinya.

13.7.1 AI local-first

Prinsip pertama adalah bahwa AI, secara bawaan, berjalan di perangkat keras yang dikendalikan produk itu sendiri. Di Profitina, platform GEO, prinsip ini mutlak: Ollama dengan model Qwen di dua jalur, tanpa jalur kode AI terhosting, tanpa data pengguna yang dikirim keluar untuk pemrosesan AI (Bab 6). Di Profitina AI, prinsip ini adalah bawaan dengan pengecualian yang terlihat: lokal kecuali pengguna memilih penyedia *cloud*. Di TTM, prinsip ini berwujud mesin difusi lokal yang ikut bekerja setiap kali ia sebaik dan secepat *cloud*.

Mengapa lokal? Bab 6 memberi tiga alasan, dan ketiganya berlaku untuk seluruh kelompok. *Privasi*: data yang tidak pernah meninggalkan mesin tidak dapat bocor dari mesin orang lain. *Kemandirian*: produk yang intinya bergantung pada API pihak ketiga berbayar mewarisi harga, batas, dan gangguan API itu; ledger penyedia yang beristirahat milik kelompok ini ada justru karena penyedia *cloud* memang beristirahat. *Kejujuran tentang biaya*: model lokal memiliki biaya yang dapat diukur di mesin sendiri, termasuk yang kecil dan mengejutkan, seperti jeda 2,25 detik yang turun menjadi 0,17 detik begitu Profitina memanggil Ollama di `127.0.0.1` alih-alih `localhost` di Windows.

Local-first ada harganya, dan wajar untuk menyebutnya: model lokal lebih kecil daripada model *cloud* terbesar, dan satu mesin hanya memiliki satu GPU. Kelompok ini membayar harga itu dengan pengukuran (gerbang kualitas di Bab 6 dan aturan TTM "setidaknya sama baik dan tidak lebih lambat"), bukan dengan harapan.

13.7.2 Aplikasi menyarankan; pengguna memutuskan

Prinsip kedua adalah bahwa perangkat lunak mengusulkan dan manusia yang menentukan. Pindaian Profitina memberi skor dan daftar perbaikan berprioritas, dengan metodologi yang diperlihatkan; pengguna memutuskan apa yang diubah di situsnya sendiri. Content Engine menulis draf; pengguna menyunting lalu menerbitkannya, atau tidak. Alur umpan balik mengkurasi usulan dengan model lokal, tetapi "keputusan produk selalu diambil oleh manusia" (Bab 7). Capital membunyikan peringatan ketika sebuah batas kebijakan dilanggar; ia tidak berdagang atas prakarsanya sendiri karena itu. Pemeriksa peneliti di Lab G9 melaporkan celah beserta saran; ia tidak menilai orangnya.

Prinsip ini juga prinsip hukum. UU PDP memberi orang hak untuk berkeberatan atas keputusan yang semata-mata didasarkan pada pemrosesan otomatis (Bab 11), dan produk yang dibangun di sekitar "pengguna memutuskan" memenuhi hak itu sejak rancangan, bukan sebagai pengecualian. Prinsip ini juga prinsip etika: sistem yang menyarankan boleh keliru tanpa menimbulkan celaka, karena ada manusia yang berdiri di antara kekeliruannya dan dunia.

13.7.3 Tes sebagai gerbang

Prinsip ketiga adalah bahwa tidak ada yang sampai ke pengguna kecuali tes otomatis berwarna hijau. Di Profitina, prinsip ini konkret dan terukur. Pada saat bab ini ditulis, 24 September 2026, rangkaian tes SQLite lengkap untuk rilis 0.14.5 mencatat **1.577 tes lulus dan 1 dilewati**, dan subset prioritas yang dijalankan terhadap PostgreSQL 16 mencatat **440 lulus**, di samping penjaga JavaScript untuk antarmuka pengguna. Rilisnya sendiri adalah gerbang enam langkah (Bab 9): kunci versi tepat yang akan dirilis, cadangkan, jalankan tes, periksa bahwa tidak ada yang berubah selama tes, salin, mulai ulang, lalu nyatakan berhasil **hanya ketika `/api/health` menjawab dengan versi baru**.

Rangkaian tes adalah gerbang, bukan laporan, karena apa yang terjadi ketika tes gagal: tidak ada yang dikirim. Laporan dapat dibaca belakangan, diperdebatkan, atau diabaikan; gerbang tidak dapat dilewati dengan memutar. Gagasan yang sama mengalir ke seluruh kelompok dalam bentuk yang lebih kecil: pemindai rahasia pada setiap *commit*, perbandingan bita demi bita keluaran Python dan Java di setiap lab buku ini, dan kontrol negatif Researcher Visibility Report.

13.7.4 Dan satu lagi: katakan apa yang sebenarnya terjadi

Subbab 13.4 dan 13.5 menemukan prinsip keempat yang mengalir di bawah tiga prinsip lainnya: *katakan apa yang sebenarnya terjadi*. "Belum yakin." "Tidak ada model yang tersedia." "Gagal dengan lantang." "Tidak pernah berhenti diam-diam." "Hasil visibilitas adalah estimasi metodologis, bukan jaminan." "Estimasi model, bukan harga pasar." "Nol pembelian PROFIT." Masing-masing adalah kalimat yang akan dihindari produk yang kurang cermat, dan masing-masing membuat produk lebih tepercaya justru karena diucapkan.

> **WAWASAN KUNCI.** Prinsip murah untuk dinyatakan dan mahal untuk dijaga. Prinsip dijaga oleh mekanisme: model lokal alih-alih kunci API, tombol yang menunggu manusia, gerbang yang menghentikan rilis, kalimat yang mengakui keterbatasan.

13.8 Lab G9: Memeriksa Rekaman Peneliti dengan OpenAlex

13.8.1 Fitur di balik lab ini

Karya seorang peneliti hanya dapat dikutip oleh mesin jawaban AI, atau ditemukan oleh koleganya, jika karya itu dapat ditemukan sama sekali, dan ditemukan sebagai karya orang yang tepat. Di katalog ilmiah, hal ini keliru dengan cara-cara yang membosankan tetapi dapat diperbaiki: karya tanpa DOI, pengenal ORCID yang tidak ditautkan (pengenal persisten yang membedakan seorang peneliti dari peneliti lain yang bernama sama), dan, paling sering, institusi yang sama ditulis dengan berbagai cara.

Researcher Visibility Report Profitina (Bab 3) mengukur tepat hal-hal ini, dengan memakai **OpenAlex** [7]. Laporan itu menyebut celah-celah konkret beserta langkah perbaikannya, misalnya "Belum ada ORCID yang ditautkan" atau "Afiliasi Anda ditulis dengan N ejaan berbeda". Laporan itu memproses karya yang telah diterbitkan dan metadata bibliografisnya, tidak pernah alamat, foto, atau tautan media sosial seseorang. Laporan itu juga memiliki **kontrol negatif** permanen: nama fiktif selalu menghasilkan "tidak ditemukan", tidak pernah skor yang tampak masuk akal; pada rilis September 2026, 8 dari 8 subjek uji fiktif tetap bernilai 0,0 [7]. Versi tingkat organisasinya, *Organization Visibility*, ketika diterapkan pada satu universitas Indonesia, menemukan **466 ejaan afiliasi yang berbeda dalam 500 karya sampel**, serta 244 dari 607 peneliti tanpa ORCID yang ditautkan [7]. Laporan itu sengaja hanya melaporkan celah agregat yang dapat ditutup sebuah organisasi, dan tidak pernah papan peringkat orang.

Lab G9 membangun pemeriksa kecil yang transparan dengan semangat yang sama. Ini bukan laporan Profitina: lab ini hanya memakai kolom publik, aturan sederhana yang dipilih untuk pengajaran, dan ambang yang dapat diubah pembaca.

13.8.2 OpenAlex secara singkat

OpenAlex adalah katalog terbuka dan gratis berisi karya ilmiah, peneliti, sumber terbitan, institusi, dan topik, yang dibuat oleh lembaga nirlaba OurResearch dan diluncurkan pada 2022 sebagai penerus terbuka Microsoft Academic Graph yang telah dihentikan [8]. Katalog ini dapat diakses melalui REST API yang mengembalikan JSON. Tiga panggilan cukup untuk lab ini:

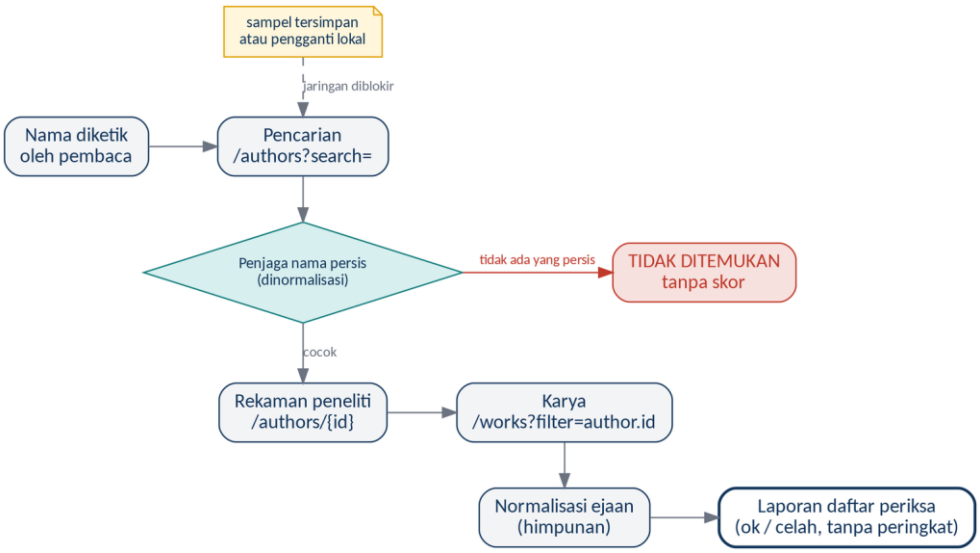
- `GET /authors?search=<name>` mencari rekaman peneliti berdasarkan nama dan mengembalikan satu halaman kandidat.
- `GET /authors/<id>` mengembalikan satu rekaman peneliti berdasarkan pengenal OpenAlex-nya, seperti `A5048491430`.
- `GET /works?filter=author.id:<id>` mendaftar karya yang diatribusikan kepada rekaman itu, masing-masing dengan DOI-nya, status akses terbukanya, dan *authorships*-nya: untuk setiap orang pada makalah itu, nama sebagaimana tercetak dan string afiliasi sebagaimana tercetak.

Aturan akses berubah pada 2026. Pada Januari 2026, OpenAlex mengumumkan kunci API dan penetapan harga berdasarkan pemakaian, yang "tetap gratis bagi sebagian besar pengguna" [9]. Per September 2026, dokumentasinya menggambarkan anggaran harian: permintaan satu rekaman berdasarkan pengenalnya gratis, seribu permintaan daftar-dan-saring memakan USD 0,10 anggaran dan seribu pencarian USD 1; tanpa kunci, pengguna memperoleh anggaran USD 0,10 per hari, dan kunci gratis menaikkannya menjadi USD 1; lebih dari 100 permintaan per detik, atau anggaran yang habis, menghasilkan `429 Too Many Requests` [10]. Program lab membaca kunci opsional dari variabel lingkungan `OPENALEX_API_KEY` dan menambahkannya ke setiap

permintaan; beberapa permintaan dalam satu pemeriksaan muat dengan leluasa dalam anggaran tanpa kunci.

13.8.3 Apa yang diperiksa pemeriksa

Gambar 13.3 memperlihatkan langkah-langkahnya.



Gambar 13.3 Lab G9. Sebuah nama dicari; hanya kecocokan persis setelah normalisasi yang lolos penjaga, dan selain itu dilaporkan sebagai tidak ditemukan tanpa skor. Ketika jaringan diblokir, sampel tersimpan atau pengganti lokal menjawab tiga panggilan yang sama.

1. **Cari** nama yang diketik dan terima kandidat-kandidatnya.
2. **Penjaga.** Terima sebuah kandidat hanya jika nama tampilannya, atau salah satu nama alternatif yang tercatat, cocok persis dengan nama yang diketik setelah normalisasi. Selain itu, cetak NOT FOUND dan jangan beri skor. Inilah kontrol negatif: mesin pencari yang ditanya tentang orang yang tidak ada akan dengan senang hati mengembalikan orang nyata yang paling mirip, dan pemeriksa yang memercayai hasil teratas akan memberi skor kepada orang asing.
3. **Rekaman.** Ambil rekaman peneliti: ORCID, jumlah karya, varian nama yang tercatat.
4. **Karya.** Ambil karya-karyanya dan hitung yang memiliki DOI dan yang berakses terbuka.
5. **Normalisasi.** Kumpulkan string afiliasi yang menyertai nama peneliti ini pada karya-karya itu dan hitung berapa yang benar-benar berbeda setelah normalisasi, dibandingkan dengan banyaknya institusi berbeda hasil pemetaan OpenAlex.
6. **Laporkan** setiap hasil sebagai `ok` atau `gap`, tanpa skor keseluruhan dan tanpa perbandingan dengan siapa pun.

Untuk laju-lajunya, pemeriksa membutuhkan persentase yang dibulatkan ke bilangan bulat, dan, seperti di Lab G8, pembulatannya harus sama di Python dan Java. Dengan bilangan bulat p (bagian) dan w (keseluruhan), persentase yang dibulatkan setengah ke atas adalah

$$\mathrm{pct}(p, w) = \left\lfloor \frac{200p + w}{2w} \right\rfloor \tag{13.6}$$

karena $\frac{200p+w}{2w} = \frac{100p}{w} + \frac{1}{2}$, dan mengambil lantai dari suatu bilangan ditambah setengah membulatkannya ke bilangan bulat terdekat, dengan setengah dibulatkan ke atas. Pembagian bilangan bulat langsung menghitung lantai itu, sehingga sama sekali tidak ada titik mengambang yang terlibat.

Ambang pengajarannya sederhana dan terlihat di kode: cakupan DOI setidaknya 90% adalah `ok`, akses terbuka setidaknya 50% adalah `ok`, paling banyak dua bentuk nama berbeda adalah `ok`, dan ejaan afiliasi berbeda yang tidak lebih banyak daripada institusi adalah `ok`. Semua itu titik awal untuk diskusi, bukan standar; Soal Latihan 9 meminta pembaca berargumen untuk ambang yang berbeda.

13.8.4 Langkah 1: menghitung ejaan dengan himpunan

Berapa banyak cara hal yang sama ditulis? Jawabannya adalah ukuran sebuah *himpunan* string yang telah dinormalisasi. Normalisasi harus menghapus perbedaan yang tidak penting dan mempertahankan yang penting. Untuk string afiliasi, lab ini mengubah ke huruf kecil, menghapus aksen (sehingga “Département” dan “Departement” cocok), mengubah setiap tanda baca menjadi spasi, dan memadatkan deretan spasi menjadi satu. Untuk nama orang, lab melakukan hal yang sama lalu mengurutkan kata-katanya, sehingga “Piwowar, Heather” dan “Heather Piwowar” menjadi kunci yang sama. Penghapusan aksen memakai bentuk normalisasi Unicode NFKD, yang memecah huruf seperti “é” menjadi “e” diikuti aksen penggabung yang kemudian dapat dibuang [11].

Kode 13.1 adalah versi Python, dengan peragaan kecil pada enam bentuk nama dan enam string afiliasi untuk satu universitas di Indonesia.

Kode 13.1 `normalize.py`: dua penormal dan sebuah himpunan yang menghitung ejaan yang benar-benar berbeda.

```
"""Lab G9, step 1: how many ways is the same thing written?

Two small normalizers: one for affiliation strings, one for personal names.
A set of normalized strings counts the spellings that really differ.
"""
import unicodedata


def norm_text(s):
    """Lowercase, drop accents, turn punctuation into spaces, squeeze spaces."""
    s = unicodedata.normalize("NFKD", s)
    s = "".join(c for c in s if not unicodedata.combining(c))
    s = "".join(c if c.isalnum() else " " for c in s.lower())
    return " ".join(s.split())


def norm_name(s):
    """Like norm_text, but word order does not matter ('Piwowar, Heather')."""
    return " ".join(sorted(norm_text(s).split()))


def spellings(strings, key):
    """Return (distinct raw strings, distinct keys)."""
    return len(set(strings)), len({key(s) for s in strings})


def main():
    names = ["Heather Piwowar", "Piwowar, Heather", "Heather, Piwowar",
             "Heather A. Piwowar", "Heather A Piwowar", "H. Piwowar"]
    for n in names:
        print(f"{n:<20} -> {norm_name(n)!r}")
    raw, keys = spellings(names, norm_name)
    print(f"names: {raw} raw spellings, {keys} after normalization")

    affiliations = [
        "Institut Teknologi Sepuluh Nopember",
        "Institut Teknologi Sepuluh Nopember, Surabaya",
        "INSTITUT TEKNOLOGI SEPULUH NOPEMBER",
        "Institut Teknologi Sepuluh Nopember.",
        "Département d'Informatique, Institut Teknologi Sepuluh Nopember",
        "ITS Surabaya",
    ]
    for a in affiliations:
        print(f"{norm_text(a)!r}")
    raw, keys = spellings(affiliations, norm_text)
    print(f"affiliations: {raw} raw spellings, {keys} after normalization")
```

Kode 13.1 (Lanjutan) *normalize.py: dua penormal dan sebuah himpunan yang menghitung ejaan yang benar-benar berbeda.*

```
if __name__ == "__main__":
    main()
```

Kode 13.2 adalah versi Java. `java.text.Normalizer` melakukan dekomposisi NFKD, dan ekspresi reguler `\p{M}` menghapus tanda-tanda penggabung [12].

Kode 13.2 `Normalize.java`: penormal yang sama dalam Java.

```
import java.text.Normalizer;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.function.Function;

/** Lab G9, step 1: how many ways is the same thing written? */
public class Normalize {
    /** Lowercase, drop accents, turn punctuation into spaces, squeeze spaces. */
    static String normText(String s) {
        String d = Normalizer.normalize(s, Normalizer.Form.NFKD)
                             .replaceAll("\\p{M}", "");
        StringBuilder b = new StringBuilder();
        for (char c : d.toLowerCase().toCharArray()) {
            b.append(Character.isLetterOrDigit(c) ? c : ' ');
        }
        return String.join(" ", b.toString().trim().split("\\s+"));
    }

    /** Like normText, but word order does not matter ("Piwowar, Heather"). */
    static String normName(String s) {
        String[] words = normText(s).split(" ");
        Arrays.sort(words);
        return String.join(" ", words);
    }

    /** Return {distinct raw strings, distinct keys}. */
    static int[] spellings(List<String> strings, Function<String, String> key) {
        Set<String> keys = new HashSet<>();
        for (String s : strings) keys.add(key.apply(s));
        return new int[] {new HashSet<>(strings).size(), keys.size()};
    }

    public static void main(String[] args) {
        List<String> names = List.of("Heather Piwowar", "Piwowar, Heather",
            "Heather, Piwowar", "Heather A. Piwowar", "Heather A Piwowar",
            "H. Piwowar");
        for (String n : names) {
            System.out.printf("%-20s -> '%s'%n", n, normName(n));
        }
        int[] r = spellings(names, Normalize::normName);
        System.out.printf("names: %d raw spellings, %d after normalization%n",
                          r[0], r[1]);
```

Kode 13.2 (Lanjutan) *Normalize.java: penormal yang sama dalam Java.*

```
        List<String> affiliations = List.of(
            "Institut Teknologi Sepuluh Nopember",
            "Institut Teknologi Sepuluh Nopember, Surabaya",
            "INSTITUT TEKNOLOGI SEPULUH NOPEMBER",
            "Institut Teknologi Sepuluh Nopember.",
            "Département d'Informatique, Institut Teknologi Sepuluh Nopember",
            "ITS Surabaya");
        for (String a : affiliations) {
            System.out.println("'" + normText(a) + "'");
        }
        r = spellings(affiliations, Normalize::normText);
        System.out.printf(
            "affiliations: %d raw spellings, %d after normalization%n",
            r[0], r[1]);
    }
}
```

Keduanya mencetak:

```
Heather Piwowar      -> 'heather piwowar'
Piwowar, Heather     -> 'heather piwowar'
Heather, Piwowar     -> 'heather piwowar'
Heather A. Piwowar   -> 'a heather piwowar'
Heather A Piwowar    -> 'a heather piwowar'
H. Piwowar           -> 'h piwowar'
names: 6 raw spellings, 3 after normalization
'institut teknologi sepuluh nopember'
'institut teknologi sepuluh nopember surabaya'
'institut teknologi sepuluh nopember'
'institut teknologi sepuluh nopember'
'departement d informatique institut teknologi sepuluh nopember'
'its surabaya'
affiliations: 6 raw spellings, 4 after normalization
```

Enam bentuk nama menyusut menjadi tiga kunci: “heather piwowar”, “a heather piwowar” (dengan inisial tengah), dan “h piwowar” (hanya dengan inisial). Enam string afiliasi menyusut menjadi empat. Normalisasi menggabungkan versi berhuruf kapital, bertanda baca, dan polos dari nama universitas itu; normalisasi dengan tepat mempertahankan “…, Surabaya” sebagai bentuk tersendiri, karena kata tambahan itu adalah informasi nyata; dan normalisasi tidak mungkin tahu bahwa “ITS Surabaya” adalah singkatan institusi yang sama. Kasus terakhir itulah batas jujur normalisasi string: sebagian varian membutuhkan pengetahuan, bukan aturan, dan itulah sebabnya OpenAlex memetakan string afiliasi ke pengenal institusi, dan mengapa pemeriksa melaporkan string mentah *dan* institusi berdampingan.

13.8.5 Langkah 2: pemeriksanya

Kode 13.3 adalah pemeriksa dalam Python. Pemeriksa ini memiliki dua sumber di balik satu antarmuka: `HttpSource` berbicara dengan API (atau dengan server mana pun yang menjawab tiga panggilan yang sama), dan `SampleSource` menjawab dari berkas tersimpan, meniru pencarian nama yang longgar dengan mengembalikan setiap rekaman tersimpan yang berbagi setidaknya satu kata dengan kueri. Semua yang berada setelah sumber, dari penjaga sampai laporan, adalah kode yang sama, sumber mana pun yang dipakai.

Kode 13.3 `researcher_check.py`: pemeriksa rekaman peneliti OpenAlex, dengan penjaga nama persis, laju menurut Persamaan (13.6), dan kegagalan jaringan yang jujur.

```
"""Lab G9: an OpenAlex researcher-record checker (a teaching tool, not a ranking).

usage: python researcher_check.py [--sample | --base URL] "Full Name"
  --sample    read the saved responses in ../samples instead of the network
  --base URL  another API root, e.g. http://127.0.0.1:7130 (sample_server.py)
Set OPENALEX_API_KEY to use a free OpenAlex key (optional).
"""
import json
import os
import sys
import urllib.error
import urllib.parse
import urllib.request

from normalize import norm_name, norm_text

API = "https://api.openalex.org"
SAMPLES = os.path.join(os.path.dirname(os.path.abspath(__file__)),
                       "..", "samples")


class SampleSource:
    """Answers from saved JSON files, imitating a fuzzy name search."""
    label = "saved sample (fetched from OpenAlex on 2026-09-24)"

    def _authors(self):
        for f in sorted(os.listdir(SAMPLES)):
            if f.startswith("author_") and f.endswith(".json"):
                with open(os.path.join(SAMPLES, f), encoding="utf-8") as fh:
                    yield json.load(fh)

    def search(self, name):
        words = set(norm_text(name).split())
        hits = []
        for a in self._authors():
            known = [a["display_name"]] + a["display_name_alternatives"]
            if any(words & set(norm_text(k).split()) for k in known):
                hits.append(a)
        return hits

    def author(self, short_id):
        with open(os.path.join(SAMPLES, f"author_{short_id}.json"),
                  encoding="utf-8") as fh:
            return json.load(fh)
```

Kode 13.3 (Lanjutan) *researcher_check.py: pemeriksa rekaman peneliti OpenAlex, dengan penjaga nama persis, laju menurut Persamaan (13.6), dan kegagalan jaringan yang jujur.*

```
    def works(self, short_id):
        with open(os.path.join(SAMPLES, f"works_{short_id}.json"),
                  encoding="utf-8") as fh:
            return json.load(fh)["results"]


class HttpSource:
    """Answers from the OpenAlex REST API (or a local stand-in)."""

    def __init__(self, base):
        self.base = base.rstrip("/")
        self.label = self.base

    def _get(self, path, params):
        key = os.environ.get("OPENALEX_API_KEY")
        if key:
            params = dict(params, api_key=key)
        url = f"{self.base}{path}?{urllib.parse.urlencode(params)}"
        req = urllib.request.Request(
            url, headers={"User-Agent": "profitina-book-lab-g9/1.0"})
        with urllib.request.urlopen(req, timeout=20) as resp:
            return json.load(resp)

    def search(self, name):
        return self._get("/authors", {"search": name, "per_page": 5})["results"]

    def author(self, short_id):
        return self._get(f"/authors/{short_id}", {})

    def works(self, short_id):
        return self._get("/works", {"filter": f"author.id:{short_id}",
                                    "per_page": 100})["results"]


def pct(part, whole):
    return (200 * part + whole) // (2 * whole) if whole else 0
```

Kode 13.3 (Lanjutan) *researcher_check.py: pemeriksa rekaman peneliti OpenAlex, dengan penjaga nama persis, laju menurut Persamaan (13.6), dan kegagalan jaringan yang jujur.*

```
def exact_match(name, candidates):
    """Negative control: accept only a candidate whose name really matches."""
    want = norm_name(name)
    for a in candidates:
        known = [a["display_name"]] + a.get("display_name_alternatives", [])
        if want in {norm_name(k) for k in known}:
            return a
    return None


def check(source, name):
    print("OpenAlex researcher-record check (a teaching tool, not a ranking)")
    print(f"query : {name}")
    print(f"source: {source.label}")
    candidates = source.search(name)
    found = exact_match(name, candidates)
    if found is None:
        print(f"result: NOT FOUND ({len(candidates)} candidate(s), none with"
              " this exact name). No score is given.")
        return 1
    short_id = found["id"].rsplit("/", 1)[-1]
    a = source.author(short_id)
    works = source.works(short_id)
    print(f"match : {a['display_name']}  {a['id']}")

    def line(ok, text):
        print(f"  [{'ok ' if ok else 'gap'}] {text}")

    line(bool(a.get("orcid")), f"ORCID linked: {a.get('orcid') or 'none'}")
    line(a["works_count"] > 0, f"works indexed: {a['works_count']}")
    names = [a["display_name"]] + a["display_name_alternatives"]
    n_keys = len({norm_name(n) for n in names})
    line(n_keys <= 2, f"name spellings: {len(set(names))} raw,"
                      f" {n_keys} after normalization")
    n = len(works)
    with_doi = sum(1 for w in works if w.get("doi"))
    with_oa = sum(1 for w in works if w["open_access"]["is_oa"])
    print(f"  works examined: {n}")
    line(pct(with_doi, n) >= 90, f"DOI present: {with_doi}/{n}"
                                 f" ({pct(with_doi, n)}%)")
    line(pct(with_oa, n) >= 50, f"open access: {with_oa}/{n}"
                                f" ({pct(with_oa, n)}%)")
    raw, insts = [], set()
    for w in works:
        for au in w["authorships"]:
            if au["author"]["id"] == a["id"]:
```

Kode 13.3 (Lanjutan) *researcher_check.py: pemeriksa rekaman peneliti OpenAlex, dengan penjaga nama persis, laju menurut Persamaan (13.6), dan kegagalan jaringan yang jujur.*

```
                raw += au["raw_affiliation_strings"]
                insts |= {i["display_name"] for i in au["institutions"]}
    n_norm = len({norm_text(r) for r in raw})
    line(n_norm <= len(insts), f"affiliation strings: {len(raw)} raw,"
         f" {n_norm} after normalization, for {len(insts)} institutions")
    return 0


def main(argv):
    source, args = HttpSource(API), list(argv)
    if args and args[0] == "--sample":
        source, args = SampleSource(), args[1:]
    elif len(args) > 1 and args[0] == "--base":
        source, args = HttpSource(args[1]), args[2:]
    if len(args) != 1:
        print(__doc__.strip())
        return 2
    try:
        return check(source, args[0])
    except (urllib.error.URLError, OSError) as err:
        print(f"network unavailable: {err}")
        print("Nothing was scored; rerun with --sample to use the saved"
              " responses.")
        return 2


if __name__ == "__main__":
    sys.exit(main(sys.argv[1:]))
```

Versi Java membutuhkan pembaca JSON, karena pustaka standar Java tidak memilikinya. **Kode 13.4** adalah pengurai *recursive descent* kecil: satu metode untuk setiap jenis nilai JSON, masing-masing memakan tepat karakter-karakter nilai itu dan memanggil metode lain untuk nilai bersarang. Pengurai ini membaca objek ke `LinkedHashMap`, larik ke `ArrayList`, bilangan bulat ke `Long`, dan bilangan lain ke `Double`, serta dapat menulis nilai kembali, yang dibutuhkan server pengganti di Langkah 3.

Kode 13.4 `Json.java`: pembaca dan penyusun JSON *recursive descent* yang kecil, cukup untuk lab ini.

```java
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

/**
 * A small JSON reader and writer, enough for the OpenAlex fields of Lab G9.
 * Objects become LinkedHashMap, arrays ArrayList, numbers Long or Double.
 */
public final class Json {
    private final String s;
    private int i;

    private Json(String s) {
        this.s = s;
    }

    public static Object parse(String text) {
        Json p = new Json(text);
        Object v = p.value();
        p.ws();
        if (p.i != p.s.length()) throw p.error("trailing characters");
        return v;
    }

    private RuntimeException error(String what) {
        return new IllegalArgumentException(what + " at offset " + i);
    }

    private void ws() {
        while (i < s.length() && Character.isWhitespace(s.charAt(i))) i++;
    }

    private Object value() {
        ws();
        if (i >= s.length()) throw error("unexpected end");
        char c = s.charAt(i);
        if (c == '{') return object();
        if (c == '[') return array();
        if (c == '"') return string();
        if (s.startsWith("true", i)) { i += 4; return Boolean.TRUE; }
        if (s.startsWith("false", i)) { i += 5; return Boolean.FALSE; }
        if (s.startsWith("null", i)) { i += 4; return null; }
        return number();
    }
```

Kode 13.4 (Lanjutan) *Json.java: pembaca dan penyusun JSON recursive descent yang kecil, cukup untuk lab ini.*

```
    private Map<String, Object> object() {
        Map<String, Object> m = new LinkedHashMap<>();
        i++;                                          // skip '{'
        ws();
        if (s.charAt(i) == '}') { i++; return m; }
        while (true) {
            ws();
            String key = string();
            ws();
            if (s.charAt(i++) != ':') throw error("expected ':'");
            m.put(key, value());
            ws();
            char c = s.charAt(i++);
            if (c == '}') return m;
            if (c != ',') throw error("expected ',' or '}'");
        }
    }

    private List<Object> array() {
        List<Object> a = new ArrayList<>();
        i++;                                          // skip '['
        ws();
        if (s.charAt(i) == ']') { i++; return a; }
        while (true) {
            a.add(value());
            ws();
            char c = s.charAt(i++);
            if (c == ']') return a;
            if (c != ',') throw error("expected ',' or ']'");
        }
    }

    private String string() {
        if (s.charAt(i++) != '"') throw error("expected a string");
        StringBuilder b = new StringBuilder();
        while (true) {
            char c = s.charAt(i++);
            if (c == '"') return b.toString();
            if (c != '\\') { b.append(c); continue; }
            char e = s.charAt(i++);
            switch (e) {
                case 'n' -> b.append('\n');
                case 't' -> b.append('\t');
                case 'r' -> b.append('\r');
                case 'b' -> b.append('\b');
                case 'f' -> b.append('\f');
```

Kode 13.4 (Lanjutan) *Json.java: pembaca dan penyusun JSON recursive descent yang kecil, cukup untuk lab ini.*

```
                case 'u' -> {
                    b.append((char) Integer.parseInt(s.substring(i, i + 4), 16));
                    i += 4;
                }
                default -> b.append(e);             // \" \\ \/
            }
        }
    }

    private Object number() {
        int start = i;
        while (i < s.length() && "+-0123456789.eE".indexOf(s.charAt(i)) >= 0) i++;
        String t = s.substring(start, i);
        if (t.isEmpty()) throw error("unexpected character");
        if (t.matches("-?\\d+")) return Long.parseLong(t);
        return Double.parseDouble(t);
    }

    /** Write a value back as compact JSON text. */
    public static String write(Object v) {
        if (v == null) return "null";
        if (v instanceof String str) {
            StringBuilder b = new StringBuilder("\"");
            for (char c : str.toCharArray()) {
                if (c == '"' || c == '\\') b.append('\\').append(c);
                else if (c < 0x20) b.append(String.format("\\u%04x", (int) c));
                else b.append(c);
            }
            return b.append('"').toString();
        }
        if (v instanceof Map<?, ?> m) {
            List<String> parts = new ArrayList<>();
            for (Map.Entry<?, ?> e : m.entrySet()) {
                parts.add(write(e.getKey().toString()) + ":" + write(e.getValue()));
            }
            return "{" + String.join(",", parts) + "}";
        }
        if (v instanceof List<?> l) {
            List<String> parts = new ArrayList<>();
            for (Object o : l) parts.add(write(o));
            return "[" + String.join(",", parts) + "]";
        }
        return v.toString();                        // numbers and booleans
    }
}
```

Kode 13.5 adalah pemeriksa itu sendiri dalam Java, dibangun di atas `java.net.http.HttpClient` [13] dan pengurai di atas.

Kode 13.5 `ResearcherCheck.java`: pemeriksa yang sama dalam Java, dengan antarmuka `Source` dan dua implementasi.

```java
import java.io.IOException;
import java.net.URI;
import java.net.URLEncoder;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.time.Duration;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.stream.Stream;

/**
 * Lab G9: an OpenAlex researcher-record checker (a teaching tool, not a ranking).
 * usage: java ResearcherCheck [--sample | --base URL] "Full Name"
 * Run it from the java folder so that ../samples is found.
 */
public class ResearcherCheck {
    static final String API = "https://api.openalex.org";
    static final Path SAMPLES = Path.of("..", "samples");

    interface Source {
        String label();
        List<Map<String, Object>> search(String name) throws IOException;
        Map<String, Object> author(String shortId) throws IOException;
        List<Map<String, Object>> works(String shortId) throws IOException;
    }

    @SuppressWarnings("unchecked")
    static Map<String, Object> obj(Object o) {
        return (Map<String, Object>) o;
    }

    @SuppressWarnings("unchecked")
    static List<Object> list(Object o) {
        return (List<Object>) o;
    }
```

Kode 13.5 (Lanjutan) *ResearcherCheck.java: pemeriksa yang sama dalam Java, dengan antarmuka Source dan dua implementasi.*

```
    static List<Map<String, Object>> objects(Object o) {
        List<Map<String, Object>> out = new ArrayList<>();
        for (Object x : list(o)) out.add(obj(x));
        return out;
    }

    static List<String> knownNames(Map<String, Object> a) {
        List<String> known = new ArrayList<>();
        known.add((String) a.get("display_name"));
        Object alt = a.get("display_name_alternatives");
        if (alt != null) for (Object x : list(alt)) known.add((String) x);
        return known;
    }

    /** Answers from saved JSON files, imitating a fuzzy name search. */
    static class SampleSource implements Source {
        public String label() {
            return "saved sample (fetched from OpenAlex on 2026-09-24)";
        }

        static Map<String, Object> read(Path p) throws IOException {
            return obj(Json.parse(Files.readString(p, StandardCharsets.UTF_8)));
        }

        public List<Map<String, Object>> search(String name) throws IOException {
            Set<String> words = Set.of(Normalize.normText(name).split(" "));
            List<Path> files;
            try (Stream<Path> st = Files.list(SAMPLES)) {
                files = st.filter(p -> p.getFileName().toString()
                                        .matches("author_.*\\.json"))
                          .sorted().toList();
            }
            List<Map<String, Object>> hits = new ArrayList<>();
            for (Path f : files) {
                Map<String, Object> a = read(f);
                boolean hit = false;
                for (String k : knownNames(a)) {
                    for (String w : Normalize.normText(k).split(" ")) {
                        hit |= words.contains(w);
                    }
                }
                if (hit) hits.add(a);
            }
            return hits;
        }
```

Kode 13.5 (Lanjutan) *ResearcherCheck.java: pemeriksa yang sama dalam Java, dengan antarmuka Source dan dua implementasi.*

```
        public Map<String, Object> author(String shortId) throws IOException {
            return read(SAMPLES.resolve("author_" + shortId + ".json"));
        }

        public List<Map<String, Object>> works(String shortId) throws IOException {
            return objects(read(SAMPLES.resolve("works_" + shortId + ".json"))
                           .get("results"));
        }
    }

    /** Answers from the OpenAlex REST API (or a local stand-in). */
    static class HttpSource implements Source {
        final String base;
        final HttpClient client = HttpClient.newBuilder()
            .connectTimeout(Duration.ofSeconds(20)).build();

        HttpSource(String base) {
            this.base = base.replaceAll("/+$", "");
        }

        public String label() {
            return base;
        }

        Map<String, Object> get(String path, String query) throws IOException {
            String key = System.getenv("OPENALEX_API_KEY");
            if (key != null && !key.isEmpty()) query += "&api_key=" + enc(key);
            HttpRequest req = HttpRequest.newBuilder(URI.create(
                    base + path + "?" + query))
                .header("User-Agent", "profitina-book-lab-g9/1.0")
                .timeout(Duration.ofSeconds(20)).build();
            try {
                HttpResponse<String> resp = client.send(req,
                    HttpResponse.BodyHandlers.ofString(StandardCharsets.UTF_8));
                if (resp.statusCode() != 200) {
                    throw new IOException("HTTP Error " + resp.statusCode());
                }
                return obj(Json.parse(resp.body()));
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new IOException("interrupted");
            }
        }
```

Kode 13.5 (Lanjutan) *ResearcherCheck.java: pemeriksa yang sama dalam Java, dengan antarmuka Source dan dua implementasi.*

```
        static String enc(String s) {
            return URLEncoder.encode(s, StandardCharsets.UTF_8);
        }

        public List<Map<String, Object>> search(String name) throws IOException {
            return objects(get("/authors", "search=" + enc(name) + "&per_page=5")
                           .get("results"));
        }

        public Map<String, Object> author(String shortId) throws IOException {
            return get("/authors/" + shortId, "");
        }

        public List<Map<String, Object>> works(String shortId) throws IOException {
            return objects(get("/works", "filter=" + enc("author.id:" + shortId)
                               + "&per_page=100").get("results"));
        }
    }

    static long pct(long part, long whole) {
        return whole == 0 ? 0 : (200 * part + whole) / (2 * whole);
    }

    /** Negative control: accept only a candidate whose name really matches. */
    static Map<String, Object> exactMatch(String name,
                                          List<Map<String, Object>> cands) {
        String want = Normalize.normName(name);
        for (Map<String, Object> a : cands) {
            for (String k : knownNames(a)) {
                if (Normalize.normName(k).equals(want)) return a;
            }
        }
        return null;
    }

    static void line(boolean ok, String text) {
        System.out.println("  [" + (ok ? "ok " : "gap") + "] " + text);
    }
```

Kode 13.5 (Lanjutan) *ResearcherCheck.java: pemeriksa yang sama dalam Java, dengan antarmuka Source dan dua implementasi.*

```
    static int check(Source source, String name) throws IOException {
        System.out.println(
            "OpenAlex researcher-record check (a teaching tool, not a ranking)");
        System.out.println("query : " + name);
        System.out.println("source: " + source.label());
        List<Map<String, Object>> candidates = source.search(name);
        Map<String, Object> found = exactMatch(name, candidates);
        if (found == null) {
            System.out.println("result: NOT FOUND (" + candidates.size()
                + " candidate(s), none with this exact name). No score is given.");
            return 1;
        }
        String id = (String) found.get("id");
        String shortId = id.substring(id.lastIndexOf('/') + 1);
        Map<String, Object> a = source.author(shortId);
        List<Map<String, Object>> works = source.works(shortId);
        System.out.println("match : " + a.get("display_name") + "  " + a.get("id"));

        Object orcid = a.get("orcid");
        line(orcid != null, "ORCID linked: " + (orcid == null ? "none" : orcid));
        long worksCount = (Long) a.get("works_count");
        line(worksCount > 0, "works indexed: " + worksCount);
        List<String> names = knownNames(a);
        Set<String> nameKeys = new HashSet<>();
        for (String n : names) nameKeys.add(Normalize.normName(n));
        line(nameKeys.size() <= 2, "name spellings: " + new HashSet<>(names).size()
             + " raw, " + nameKeys.size() + " after normalization");

        int n = works.size(), withDoi = 0, withOa = 0;
        for (Map<String, Object> w : works) {
            if (w.get("doi") != null) withDoi++;
            if ((Boolean) obj(w.get("open_access")).get("is_oa")) withOa++;
        }
        System.out.println("  works examined: " + n);
        line(pct(withDoi, n) >= 90, "DOI present: " + withDoi + "/" + n
             + " (" + pct(withDoi, n) + "%)");
        line(pct(withOa, n) >= 50, "open access: " + withOa + "/" + n
             + " (" + pct(withOa, n) + "%)");
```

Kode 13.5 (Lanjutan) *ResearcherCheck.java: pemeriksa yang sama dalam Java, dengan antarmuka Source dan dua implementasi.*

```
        List<String> raw = new ArrayList<>();
        Set<String> insts = new HashSet<>();
        for (Map<String, Object> w : works) {
            for (Map<String, Object> au : objects(w.get("authorships"))) {
                if (!obj(au.get("author")).get("id").equals(a.get("id"))) continue;
                for (Object r : list(au.get("raw_affiliation_strings"))) {
                    raw.add((String) r);
                }
                for (Map<String, Object> i : objects(au.get("institutions"))) {
                    insts.add((String) i.get("display_name"));
                }
            }
        }
        Set<String> normed = new HashSet<>();
        for (String r : raw) normed.add(Normalize.normText(r));
        line(normed.size() <= insts.size(), "affiliation strings: " + raw.size()
             + " raw, " + normed.size() + " after normalization, for "
             + insts.size() + " institutions");
        return 0;
    }

    public static void main(String[] args) {
        Source source = new HttpSource(API);
        List<String> rest = new ArrayList<>(List.of(args));
        if (!rest.isEmpty() && rest.get(0).equals("--sample")) {
            source = new SampleSource();
            rest.remove(0);
        } else if (rest.size() > 1 && rest.get(0).equals("--base")) {
            source = new HttpSource(rest.get(1));
            rest = rest.subList(2, rest.size());
        }
        if (rest.size() != 1) {
            System.out.println("usage: java ResearcherCheck [--sample | --base URL]"
                + " \"Full Name\"");
            System.exit(2);
        }
        try {
            System.exit(check(source, rest.get(0)));
        } catch (IOException err) {
            System.out.println("network unavailable: " + err);
            System.out.println("Nothing was scored; rerun with --sample to use the"
                + " saved responses.");
            System.exit(2);
        }
    }
}
```

13.8.6 Menjalankannya, dan apa yang terjadi pada jaringan

Program-program ini dijalankan pada 24 September 2026. Mesin yang dipakai untuk menyiapkan bab ini mengakses internet melalui sebuah proksi, dan proksi itu menolak koneksi ke `api.openalex.org`. Pemeriksa melakukan apa yang memang ditulis untuknya. Python mencetak:

```
OpenAlex researcher-record check (a teaching tool, not a ranking)
query : Heather Piwowar
source: https://api.openalex.org
network unavailable: <urlopen error Tunnel connection failed: 403 Forbidden>
Nothing was scored; rerun with --sample to use the saved responses.
```

dan Java, yang klien HTTP-nya merumuskan kegagalan yang sama secara berbeda, mencetak:

```
OpenAlex researcher-record check (a teaching tool, not a ranking)
query : Heather Piwowar
source: https://api.openalex.org
network unavailable: java.io.IOException: Tunnel failed, got: 403
Nothing was scored; rerun with --sample to use the saved responses.
```

Tidak ada yang diberi skor, tidak ada yang dikarang, dan pesannya menyatakan dengan tepat apa yang harus dilakukan selanjutnya. Inilah hasil pertama lab ini, dan dalam arti tertentu hasil terpentingnya.

Karena itu, percobaan-percobaan di bawah memakai **respons tersimpan**, di folder `samples` di samping kode. Peneliti yang dipilih adalah Heather Piwowar, salah seorang pendiri OurResearch dan OpenAlex itu sendiri, yang rekamannya publik dan karyanya tentang data terbuka dan akses terbuka dikenal luas [14]. Berkas `author_A5048491430.json` adalah salinan ringkas rekaman penelitinya sebagaimana dikembalikan oleh `GET /authors/A5048491430` pada 24 September 2026, yang hanya menyimpan kolom yang dibaca pemeriksa. Berkas `works_A5048491430.json` berisi lima karyanya, masing-masing diambil dari API pada hari yang sama berdasarkan DOI-nya dan diringkas menjadi *authorship* miliknya sendiri; alamat surel yang muncul di dalam dua string afiliasi telah dihapus. Titik akhir daftar yang sesungguhnya akan mengembalikan semua karyanya yang terindeks, 99 pada tanggal itu; sampelnya berisi lima, dan laporan mencetak "works examined: 5" agar tidak ada yang mengira sampel itu sebagai keseluruhan. Setiap berkas diawali kolom `_note` yang menyatakan semua ini, sehingga data membawa asal-usulnya sendiri.

Dengan sampel tersimpan, `python researcher_check.py --sample "Heather Piwowar"` dan `java ResearcherCheck --sample "Heather Piwowar"` sama-sama mencetak:

```
OpenAlex researcher-record check (a teaching tool, not a ranking)
query : Heather Piwowar
source: saved sample (fetched from OpenAlex on 2026-09-24)
match : Heather Piwowar  https://openalex.org/A5048491430
  [ok ] ORCID linked: https://orcid.org/0000-0003-1613-5981
  [ok ] works indexed: 99
  [gap] name spellings: 7 raw, 3 after normalization
  works examined: 5
  [ok ] DOI present: 5/5 (100%)
  [ok ] open access: 5/5 (100%)
  [gap] affiliation strings: 9 raw, 8 after normalization, for 6 institutions
```

Bacalah laporan itu baris demi baris. ORCID-nya tertaut, dan itulah hal paling berguna yang dapat dilakukan seorang peneliti demi kemudahan dirinya ditemukan. Sembilan puluh sembilan karya terindeks. Rekaman itu membawa tujuh bentuk nama yang menyusut menjadi tiga setelah normalisasi, tiga yang sama dengan yang kita lihat di Langkah 1: sebuah `gap`, tetapi ringan dan sangat umum. Kelima karya sampel memiliki DOI dan kelimanya berakses terbuka, tidak mengejutkan bagi peneliti yang bertahun-tahun menulis tentang akses terbuka. Dan baris terakhirlah yang menarik: di lima karya, afiliasinya muncul sebagai sembilan string mentah, delapan setelah normalisasi, yang dipetakan OpenAlex ke enam institusi. Dua string untuk University of Pittsburgh hanya berbeda pada titik penutup dan tergabung; sisanya memang berbeda, termasuk satu kalimat biografi utuh yang diletakkan penerbit di kolom afiliasi. Manusia yang membaca string-string itu akan melihat empat atau lima tempat nyata; mesin melihat delapan ejaan. Inilah, dalam bentuk mini, masalah yang menghasilkan 466 ejaan dalam 500 karya di satu universitas.

Sekarang kontrol negatifnya. Kueri yang berbagi satu kata dengan rekaman nyata tetapi bukan nama orang itu:

```
OpenAlex researcher-record check (a teaching tool, not a ranking)
query : Piwowar
source: saved sample (fetched from OpenAlex on 2026-09-24)
result: NOT FOUND (1 candidate(s), none with this exact name). No score is given.
```

dan sebuah nama yang dikarang untuk buku ini:

```
OpenAlex researcher-record check (a teaching tool, not a ranking)
query : Sekar Laundrawati
source: saved sample (fetched from OpenAlex on 2026-09-24)
result: NOT FOUND (0 candidate(s), none with this exact name). No score is given.
```

Kueri pertama memiliki satu kandidat, yaitu rekaman tersimpan yang cocok dengan kata "Piwowar", dan penjaga menolaknya karena "piwowar" bukan bentuk ternormalisasi dari nama mana pun yang tercatat untuknya. Kueri kedua sama sekali tidak memiliki kandidat. Keduanya berakhir dengan cara yang sama: NOT FOUND, tanpa skor, dan kode keluar bukan nol yang dapat diuji oleh sebuah skrip.

13.8.7 Langkah 3: pengganti lokal untuk API

Mode sampel tersimpan menguji segala hal kecuali kode HTTP. Untuk menguji bagian itu juga, tanpa internet, lab ini menyertakan server kecil yang menjawab tiga panggilan yang sama dari berkas tersimpan di `127.0.0.1:7130`. **Kode 13.6** adalah versi Python, dibangun di atas `http.server` [15].

Kode 13.6 `sample_server.py`: pengganti lokal untuk tiga panggilan OpenAlex, menjawab dari sampel tersimpan.

```
"""Lab G9, step 3: a local stand-in for the three OpenAlex calls we use.

It answers from the saved files in ../samples, so the HTTP code path of
researcher_check.py can be tried without the internet.
usage: python sample_server.py [port]      (default 7130; Ctrl+C to stop)
"""
import json
import sys
import urllib.parse
from http.server import BaseHTTPRequestHandler, ThreadingHTTPServer

from researcher_check import SampleSource

SOURCE = SampleSource()


class Handler(BaseHTTPRequestHandler):
    def do_GET(self):
        url = urllib.parse.urlsplit(self.path)
        q = urllib.parse.parse_qs(url.query)
        try:
            if url.path == "/authors" and "search" in q:
                hits = SOURCE.search(q["search"][0])
                body = {"meta": {"count": len(hits)}, "results": hits}
            elif url.path.startswith("/authors/"):
                body = SOURCE.author(url.path.split("/")[-1])
            elif url.path == "/works" and "filter" in q:
                short_id = q["filter"][0].split(":", 1)[1]
                results = SOURCE.works(short_id)
                body = {"meta": {"count": len(results)}, "results": results}
            else:
                return self.send_error(404, "unknown route")
        except FileNotFoundError:
            return self.send_error(404, "no saved sample for this id")
        data = json.dumps(body).encode("utf-8")
        self.send_response(200)
        self.send_header("Content-Type", "application/json; charset=utf-8")
        self.send_header("Content-Length", str(len(data)))
        self.end_headers()
        self.wfile.write(data)

    def log_message(self, fmt, *args):
        print("served", self.path.split("?")[0], file=sys.stderr)
```

Kode 13.6 (Lanjutan) *sample_server.py: pengganti lokal untuk tiga panggilan OpenAlex, menjawab dari sampel tersimpan.*

```
def main():
    port = int(sys.argv[1]) if len(sys.argv) > 1 else 7130
    server = ThreadingHTTPServer(("127.0.0.1", port), Handler)
    print(f"sample OpenAlex stand-in on http://127.0.0.1:{port}")
    try:
        server.serve_forever()
    except KeyboardInterrupt:
        server.server_close()


if __name__ == "__main__":
    main()
```

Kode 13.7 adalah versi Java, dibangun di atas paket `com.sun.net.httpserver` dari JDK.

Kode 13.7 `SampleServer.java`: server pengganti yang sama dalam Java.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.nio.file.NoSuchFileException;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

/**
 * Lab G9, step 3: a local stand-in for the three OpenAlex calls we use.
 * It answers from ../samples. usage: java SampleServer [port]   (default 7130)
 */
public class SampleServer {
    static final ResearcherCheck.SampleSource SOURCE =
        new ResearcherCheck.SampleSource();

    static Map<String, String> query(String raw) {
        Map<String, String> q = new LinkedHashMap<>();
        if (raw == null) return q;
        for (String pair : raw.split("&")) {
            int eq = pair.indexOf('=');
            if (eq > 0) {
                String k = pair.substring(0, eq), v = pair.substring(eq + 1);
                q.put(URLDecoder.decode(k, StandardCharsets.UTF_8),
                      URLDecoder.decode(v, StandardCharsets.UTF_8));
            }
        }
        return q;
    }

    static Map<String, Object> page(List<Map<String, Object>> results) {
        Map<String, Object> body = new LinkedHashMap<>();
        body.put("meta", Map.of("count", results.size()));
        body.put("results", results);
        return body;
    }
```

Kode 13.7 (Lanjutan) *SampleServer.java: server pengganti yang sama dalam Java.*

```
    static void handle(HttpExchange ex) throws IOException {
        String path = ex.getRequestURI().getPath();
        Map<String, String> q = query(ex.getRequestURI().getRawQuery());
        System.err.println("served " + path);
        Object body;
        int status = 200;
        try {
            if (path.equals("/authors") && q.containsKey("search")) {
                body = page(SOURCE.search(q.get("search")));
            } else if (path.startsWith("/authors/")) {
                body = SOURCE.author(path.substring(path.lastIndexOf('/') + 1));
            } else if (path.equals("/works") && q.containsKey("filter")) {
                String shortId = q.get("filter").split(":", 2)[1];
                body = page(SOURCE.works(shortId));
            } else {
                status = 404;
                body = Map.of("error", "unknown route");
            }
        } catch (NoSuchFileException e) {
            status = 404;
            body = Map.of("error", "no saved sample for this id");
        }
        byte[] data = Json.write(body).getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type",
                                    "application/json; charset=utf-8");
        ex.sendResponseHeaders(status, data.length);
        try (OutputStream out = ex.getResponseBody()) {
            out.write(data);
        }
    }

    public static void main(String[] args) throws IOException {
        int port = args.length > 0 ? Integer.parseInt(args[0]) : 7130;
        HttpServer server = HttpServer.create(
            new InetSocketAddress("127.0.0.1", port), 0);
        server.createContext("/", SampleServer::handle);
        server.start();
        System.out.println("sample OpenAlex stand-in on http://127.0.0.1:" + port);
    }
}
```

Jalankan salah satu server di satu terminal (`python sample_server.py` atau `java SampleServer`), lalu di terminal kedua jalankan pemeriksa terhadapnya dengan `--base http://127.0.0.1:7130`. Pemeriksa mencetak:

```
OpenAlex researcher-record check (a teaching tool, not a ranking)
query : Heather Piwowar
source: http://127.0.0.1:7130
match : Heather Piwowar  https://openalex.org/A5048491430
  [ok ] ORCID linked: https://orcid.org/0000-0003-1613-5981
  [ok ] works indexed: 99
  [gap] name spellings: 7 raw, 3 after normalization
  works examined: 5
  [ok ] DOI present: 5/5 (100%)
  [ok ] open access: 5/5 (100%)
  [gap] affiliation strings: 9 raw, 8 after normalization, for 6 institutions
```

dan server mencatat tiga permintaan yang dijawabnya:

```
sample OpenAlex stand-in on http://127.0.0.1:7130
served /authors
served /authors/A5048491430
served /works
```

Laporannya identik dengan laporan sampel tersimpan kecuali baris `source`, yang membuktikan bahwa jalur HTTP, penguraian JSON, dan pengodean kueri semuanya bekerja. Kedua bahasa juga dapat saling menggantikan di sini: pemeriksa Python terhadap server Java, dan pemeriksa Java terhadap server Python, menghasilkan laporan yang sama. Hentikan server dengan Ctrl+C setelah selesai. Di mesin yang dapat menjangkau OpenAlex, menjalankan pemeriksa tanpa opsi apa pun akan menanyai katalog yang sesungguhnya, dan angkanya adalah angka pada hari Anda menjalankannya.

> **PERINGATAN.** Pakailah pemeriksa ini pada rekaman Anda sendiri, atau pada ilmuwan publik untuk belajar, dan jangan pernah untuk membandingkan atau memeringkat orang. Celah pada sebuah rekaman mengatakan sesuatu tentang metadata, bukan tentang mutu riset seseorang. Batasi jumlah permintaan, kenali program Anda dengan User-Agent yang jelas, dan pakailah kunci API untuk apa pun yang lebih dari segelintir pemeriksaan.

13.8.8 Mengembangkan pemeriksa lebih jauh

Pemeriksa ini sengaja dibuat kecil, dan setiap keterbatasannya adalah sebuah undangan. Lima pengembangan layak dikerjakan, kira-kira dalam urutan berikut, dan masing-masing mengajarkan sesuatu yang telah dipersiapkan bab-bab sebelumnya.

Bedakan dengan ORCID. Terima argumen opsional `--orcid`, dan bila argumen itu diberikan, biarkan penjaga hanya menerima kandidat yang kolom `orcid`-nya cocok. Dua orang nyata yang bernama sama pun tidak lagi menjadi masalah (Soal Latihan 7 menanyakan mengapa sekarang hal itu masih menjadi masalah). Ini langkah yang sama dengan yang diambil Bab 11 untuk data pribadi: kenali seseorang dengan pengenal stabil yang ia kendalikan, bukan dengan string yang juga disandang orang lain.

Baca setiap halaman. Titik akhir karya yang sesungguhnya mengembalikan hasil per halaman. Peneliti dengan 400 karya membutuhkan beberapa permintaan, dan pemeriksa harus mengikuti petunjuk halaman dalam respons sampai tidak ada lagi halaman berikutnya. Gelungnya pendek, tetapi gelung itu harus berhenti sendiri, menghitung apa yang telah dilihatnya, dan menyatakannya dalam laporan, sehingga "works examined" selalu benar.

Tetap santun saat ditekan. Ketika OpenAlex menjawab `429 Too Many Requests`, tanggapan yang tepat bukanlah mencoba lagi seketika. Tunggu, lalu coba lagi, dan gandakan waktu tunggu setiap kali sampai suatu batas: *exponential backoff* (penundaan eksponensial). Setelah beberapa kegagalan, berhenti dan laporkan, persis seperti pemutus sirkuit di Subbab 13.2.2. Tambahkan pula *cache* kecil di cakram, sehingga memeriksa orang yang sama dua kali dalam sehari tidak memerlukan permintaan kedua; inilah kesantunan yang sama dengan yang diminta Bab 3 dari pemeriksa kesiapan.

Rincilah lajunya. Akses terbuka memiliki beberapa warna (*gold*, *green*, *hybrid*, *bronze*), dan peneliti yang tidak dapat berbuat apa-apa terhadap kebijakan penerbit sering kali masih dapat menyimpan manuskrip yang telah diterima di sebuah repositori. Mencetak rinciannya mengubah satu angka menjadi saran yang dapat dilaksanakan seseorang. Hal yang sama berlaku untuk DOI: mendaftar judul karya yang belum memiliki DOI lebih berguna daripada sebuah persentase.

Agregasikan, jangan memeringkat. Organization Visibility bekerja pada tingkat institusi: ia menghitung ejaan dan pengenal yang hilang di seluruh penelitinya lalu melaporkan totalnya. Pembaca yang mengembangkan lab ke arah itu sebaiknya memegang garis yang sama: jumlahkan celahnya, tunjukkan berapa yang dapat ditutup, dan jangan pernah mencetak daftar orang yang diurutkan berdasarkan apa pun. Laporan yang menemukan 466 ejaan dalam 500 karya

membantu satu universitas utuh merapikan metadatanya justru karena laporan itu menunjuk sebuah kebiasaan bersama, bukan perorangan.

Setiap pengembangan tetap memegang tiga prinsip. Pemeriksa tetap program lokal tanpa layanan tersembunyi; ia menyarankan, dan peneliti yang memutuskan apa yang diperbaiki; dan setiap pengembangan layak mendapat tes yang mengunci perilakunya, dimulai dari yang terpenting, yaitu bahwa nama yang tidak ada tetap menghasilkan NOT FOUND.

13.9 Apa Selanjutnya

Gambar 13.4 menempatkan jalan ke depan sesudah jalan yang telah dilalui, dengan perubahan garis yang jelas di tempat fakta berakhir dan rencana dimulai.

Gambar 13.4 Jalan yang telah dilalui (garis utuh) dan jalan ke depan (garis putus-putus). Segala hal di sebelah kiri September 2026 telah terjadi; segala hal di sebelah kanannya direncanakan dan bergerbang.

13.9.1 Jalan yang telah dilalui

Tanggal-tanggal pada garis utuh adalah yang telah diceritakan buku ini: *commit* pertama basis kode sekarang pada 3 Juli 2026; peluncuran produksi pada 17 Juli, dengan basis data diatur ulang ke nol (“launch-fresh”); token PROFIT di devnet Solana pada akhir Juli; kohort universitas pertama sekitar 250 mahasiswa pada 31 Agustus; dan, pada September, whitepaper publik v24 serta rilis 0.14.5-testnet, yang di-*commit* pada 24 September 2026, hari ketika bab ini ditulis. Di antara tanggal-tanggal itu terbentang fitur-fitur Bab 3 sampai 12, masing-masing dikirim, diukur, dan, bila kurang memadai, diperbaiki.

13.9.2 Jalan ke depan

Fase 3: mainnet PROFIT dan Ultralight DEX. Whitepaper memberi label K4 2026 pada fase ini dan melekatkan gerbangnya dalam kalimat yang sama: audit token dan perizinan di bawah regulator Indonesia [7]. Bab 12 menjelaskan mengapa gerbang lebih penting daripada label. Ketika syarat-syaratnya terpenuhi, PROFIT pindah ke mainnet, *mint authority* pindah ke multisig, dan penukaran nonkustodian melalui agregator Jupiter menjadi mungkin. Sampai saat itu, penjualan tetap dinonaktifkan.

Fase 4: skala regional dan sebuah indeks. Setelah memimpin di dalam negeri, whitepaper merencanakan perluasan ke kawasan dan sebuah “Profitina AI Visibility Index” [7]. Urutannya disengaja: keunggulan yang diklaim whitepaper (bahasa Indonesia, kepatuhan UU PDP, QRIS) bersifat lokal, dan harus nyata di dalam negeri sebelum diekspor.

Hukum privasi mulai berlaku. Peraturan pelaksana UU PDP, PP No. 33 Tahun 2026, mulai berlaku pada 16 Januari 2027 (Bab 11). Produk yang dibangun di atas prinsip-prinsip buku ini semestinya memenuhinya tanpa drama, dan gerbang rilis adalah tempat hal itu akan diperiksa.

Siklus model. Alur umpan balik di Bab 7 sudah berjalan, dan kelak dapat menghidupi sebuah model lokal yang disetel ulang (*fine-tuned*). Langkah itu tetap berada di peta jalan; ketika tiba, ia akan dirilis dengan cara yang sama seperti hal lainnya: diukur terhadap model yang ada, diputuskan oleh manusia, dan dikirim melalui gerbang.

Tempat berlatih. Bahan Profitina Laundry yang direncanakan (kasus penggunaan, diagram hubungan entitas, transaksi contoh, dan README per bab) tumbuh bersama buku-buku yang memakainya.

13.9.3 Yang tidak akan berubah

Rencana bergeser. Tanggal mundur, fase bergabung, fitur dihapus ketika pengukuran menyatakan demikian (Bab 8 menceritakan bagaimana paket Agency dihapus). Yang tidak semestinya bergeser adalah hal-hal yang telah coba ditunjukkan buku ini sebagai prinsip, bukan selera: model berjalan secara lokal; aplikasi menyarankan dan pengguna memutuskan; tidak ada yang dikirim kecuali tes hijau dan pemeriksaan kesehatan menyatakannya; produk mengatakan apa yang sebenarnya terjadi; hak-hak di Bab 11 gratis selamanya; dan tidak ada harga berbayar yang pernah ditulis di samping janji tentang sesuatu yang belum dibangun. Pembaca yang kembali ke Profitina setahun lagi semestinya mendapati daftar fitur yang berbeda dan prinsip yang sama.

13.9.4 Lab-lab, sekali lagi

Jalan ke depan bagi seorang pembaca lebih pendek dan lebih pasti daripada jalan produknya: jalan itu melewati lab-lab. **Tabel 13.1** mendaftar setiap lab di buku ini beserta gagasan yang diajarkannya dan prinsip di Subbab 13.7 yang paling dilayaninya. Masing-masing cukup kecil untuk dibaca dalam satu malam dan cukup lengkap untuk dikembangkan selama satu semester.

Tabel 13.1 Lab-lab dalam buku ini, gagasan yang diajarkan masing-masing, dan prinsip yang paling dilayaninya.

Bab	Lab	Gagasannya	Prinsip yang paling dilayani
2	G6: generator JSON-LD, robots.txt, dan llms.txt	Fakta yang dapat dibaca mesin tentang sebuah usaha	Aplikasi menyarankan; pengguna menerbitkan
3	G1: pemeriksa kesiapan AI mini	Sinyal yang dapat diamati, dijelaskan satu per satu	Katakan yang sebenarnya: setiap pemeriksaan menampilkan buktinya
4	G2: umpan langsung melalui Server-Sent Events	Dorongan satu arah, invalidasi lawan pengambilan ulang	Tes sebagai gerbang untuk perilaku waktu nyata
5	G3: penyaring tautan berbasis aturan	Lantai aturan deterministik, diukur pada korpus berlabel	Ukur sebelum percaya
6	G5: model lokal melalui API HTTP Ollama	Cadangan yang jujur ketika tidak ada model	AI *local-first*
7	Alur umpan balik	Kirim, kurasi, dataset, ukur, rilis	Pengguna memutuskan apa yang dikirim
8	Paket, kuota, dan mesin keadaan langganan	Hari dihitung dalam WIB; tanpa kredit ganda	Tes sebagai gerbang
9	G4: penjadwal dan pemeriksaan kesehatan	Antrean prioritas tugas yang jatuh tempo; berhasil hanya setelah sehat	Tes sebagai gerbang
10	Pengaudit header cache dan robots	Verifikasi URL yang hidup, melewati cache	Katakan yang sebenarnya tentang apa yang benar-benar disajikan
11	G7: log persetujuan, ekspor, dan hapus	Hak sebagai tombol; tidak ada data pribadi yang tersisa	Pengguna memutuskan tentang datanya sendiri
12	G8: simulator emisi dan pembakaran	Hukum pangkat, deret geometri, buku besar bilangan bulat	Katakan yang sebenarnya: ilustratif, tidak pernah harga
13	G9: pemeriksa rekaman peneliti	Kontrol negatif dan ejaan yang dinormalisasi	Katakan yang sebenarnya: tidak ditemukan berarti tanpa skor

Bacalah kolom terakhir dari atas ke bawah, dan tiga prinsip itu, dengan prinsip keempat di bawahnya, muncul berulang-ulang. Pengulangan itulah intinya. Sebuah prinsip dipelajari dengan cara yang sama seperti sebuah program: dengan menjalankannya berkali-kali pada masukan yang berbeda sampai perilakunya menjadi akrab.

13.10 Profitina dalam Praktik

Pada rilis September 2026, bab ini berpadanan dengan produk-produk nyata sebagai berikut.

- **Kelompoknya.** Satu repositori payung dengan enam lapisan bersama (bahasa antarmuka, Conductor, pembaruan mandiri, mesin gambar *cloud* dan lokal, ledger penyedia yang beristirahat), disalin ke bawah ke produk-produk dan tidak pernah ke atas; dua mesin dengan rutinitas tarik-sebelum, sinkron-sesudah; pemindai rahasia pada setiap *commit*.
- **Produk-produknya.** Profitina (platform GEO di app.profitina.com, rilis 0.14.5-testnet); Profitina-Deploy (tema profitina.com); Profitina Capital 3.8 (perkakas kuantitatif privat dengan meja risiko); Profitina TTM 2.8.1 (teks ke film pendek, pelukisan hibrida, gagal dengan lantang); Profitina AI 1.7.2 (pendamping AI *local-first*, tidak pernah berhenti diam-diam). Profitina tidak memakai Profitina AI maupun AI *cloud* apa pun.
- **Tempat berlatih.** Profitina Laundry, fiktif, dalam Java dan Python, berbagi gagasan dan tidak pernah kode atau data.

- **Prinsip-prinsipnya.** AI *local-first*; aplikasi menyarankan dan pengguna memutuskan; tes sebagai gerbang (1.577 lulus dan 1 dilewati pada rangkaian SQLite lengkap, 440 lulus pada subset PostgreSQL 16, pada rilis 0.14.5); katakan apa yang sebenarnya terjadi.
- **Researcher Visibility Report dan Organization Visibility.** Dibangun di atas OpenAlex, dengan kontrol negatif permanen (8 dari 8 nama fiktif bernilai 0,0) dan laporan organisasi yang hanya agregat (466 ejaan dalam 500 karya; 244 dari 607 tanpa ORCID di satu universitas) [7].
- **Jalan ke depan.** Fase 3 (mainnet dan Ultralight DEX) bergerbang audit dan perizinan; Fase 4 (skala regional dan sebuah indeks) setelah memimpin di dalam negeri.

Lab G9 tidak mereproduksi kode, bobot, maupun ambang Researcher Visibility Report. Lab ini memakai tiga panggilan API publik, aturan yang dapat diperiksa setiap pembaca, dan sampel tersimpan yang asal-usulnya tertulis di dalamnya.

13.11 Kesalahpahaman yang Sering Terjadi

KESALAHPAHAMAN 1: "Profitina memakai Profitina AI, jadi Profitina memakai AI *cloud*." Profitina AI adalah produk saudara. Profitina, platform GEO, menjalankan model lokalnya sendiri melalui Ollama dan sama sekali tidak memakai AI *cloud*.

KESALAHPAHAMAN 2: "*Local-first* berarti hanya lokal." Di Profitina AI artinya lokal secara bawaan, *cloud* bila pengguna memilihnya. Hanya platform GEO yang sepenuhnya lokal.

KESALAHPAHAMAN 3: "Profitina Laundry adalah peragaan kode Profitina." Profitina Laundry adalah aplikasi pengajaran fiktif tanpa hubungan dengan kode atau data produk yang sebenarnya. Ia hanya berbagi gagasan dan pola.

KESALAHPAHAMAN 4: "Profitina Capital memberi nasihat investasi." Profitina Capital adalah perkakas privat yang menghitung statistik risiko. Statistik menggambarkan masa lalu; contoh-contoh di bab ini dikarang dan tidak menyarankan apa pun.

KESALAHPAHAMAN 5: "Jika pencarian mengembalikan seseorang, pemeriksa harus memberinya skor." Mesin pencari mengembalikan kecocokan terdekat bahkan untuk nama yang tidak ada. Penjaga nama persis itulah yang mencegah pemeriksa memberi skor kepada orang asing; tidak ditemukan harus berarti tanpa skor.

KESALAHPAHAMAN 6: "Peneliti dengan lebih banyak ejaan afiliasi adalah peneliti yang lebih buruk." Ejaan mengukur kerapian metadata, yang sebagian besar ditentukan penerbit dan pengindeks, bukan mutu riset. Pemeriksa melaporkan celah untuk diperbaiki, tidak pernah peringkat.

KESALAHPAHAMAN 7: "Ketika API tidak terjangkau, mengembalikan nol tidak berbahaya." Nol akan tampak seperti hasil nyata yang buruk. Pemeriksa mencetak galat jaringan dan tidak memberi skor apa pun, dan sampel tersimpan diberi label sebagai sampel.

KESALAHPAHAMAN 8: "K4 2026 adalah tanggal mainnet." Itu label yang dilekati gerbang: audit dan perizinan. Gerbanglah yang menentukan; label mengikuti.

13.12 Ringkasan Bab

- Profitina Group adalah payung produk-produk bersaudara yang berbagi enam lapisan, disalin ke bawah dan tidak pernah ke atas (**Gambar 13.1**); ledger penyedia yang beristirahat adalah pemutus sirkuit bersama.
- Profitina Capital 3.8 adalah meja kuantitatif privat dengan meja risiko. Pada sepuluh hari rekaan kita menghitung *value at risk* (Persamaan (13.1)), *expected shortfall* (Persamaan (13.2)),

drawdown maksimum (Persamaan (13.3)), rasio Sharpe (Persamaan (13.4)), dan indeks Herfindahl–Hirschman (Persamaan (13.5)).

- Profitina TTM 2.8.1 mengubah teks menjadi film pendek melalui perencana, sutradara dengan enam ketukan cerita, pelukisan hibrida dengan mesin difusi lokal, suara, bunyi prosedural, dan ffmpeg, serta gagal dengan lantang.
- Profitina AI 1.7.2 adalah pendamping AI *local-first* dengan Conductor, perkakas, dan Studio; ia tidak pernah berhenti diam-diam. Ia bukan bagian dari app.profitina.com, yang tidak memakai AI *cloud*.
- Profitina Laundry adalah tempat berlatih yang fiktif: aman, kecil, dan hanya berbagi gagasan.
- Tiga prinsip mengikat kelompok ini (**Gambar 13.2**): AI *local-first*; aplikasi menyarankan dan pengguna memutuskan; tes sebagai gerbang. Di bawahnya mengalir prinsip keempat: katakan apa yang sebenarnya terjadi.
- Lab G9 memeriksa rekaman peneliti di OpenAlex: pencarian, penjaga nama persis sebagai kontrol negatif, rekaman, karya, ejaan ternormalisasi yang dihitung dengan himpunan, dan laporan `ok` dan `gap` dengan laju menurut Persamaan (13.6) (**Gambar 13.3**). Ketika jaringan diblokir, pemeriksa mengatakannya dan tidak memberi skor apa pun; sampel tersimpan yang berlabel dan server pengganti lokal menguji setiap jalur.
- Lab-lab buku ini, masing-masing dipetakan ke prinsip yang paling dilayaninya, dihimpun dalam **Tabel 13.1**.
- Jalan ke depan (**Gambar 13.4**): Fase 3 (mainnet dan Ultralight DEX) setelah audit dan perizinan; Fase 4 (skala regional dan sebuah indeks) setelah memimpin di dalam negeri; PP 33/2026 mulai 16 Januari 2027. Rencana bergeser; prinsip semestinya tidak.

13.13 Soal Latihan

1. **Konsentrasi.** Hitung indeks Herfindahl–Hirschman pada Persamaan (13.5) untuk bobot 0.4,0.4,0.2 beserta kebalikannya. Kemudian hitung untuk lima bobot yang sama. Portofolio mana yang lebih terkonsentrasi, dan seberapa besar?

2. **VaR dan expected shortfall.** Untuk sepuluh imbal hasil harian di Subbab 13.3.2, hitung $\mathrm{VaR}_{90\%}$ dan $\mathrm{CVaR}_{90\%}$ menurut Persamaan (13.1) dan Persamaan (13.2), lalu $\mathrm{VaR}_{70\%}$ dan $\mathrm{CVaR}_{70\%}$.

3. **Drawdown maksimum dalam satu penelusuran.** Uraikan algoritma yang menghitung Persamaan (13.3) untuk deret n nilai dalam waktu $O(n)$ dan memori tambahan $O(1)$, lalu terapkan pada nilai-nilai 100,110,99,105,90,120,108.

4. **Normalisasi secara manual.** Terapkan `norm_text` dari **Kode 13.1** pada empat string `"Universitas Indonesia"`, `"UNIVERSITAS INDONESIA, Depok"`, `"Universitas  Indonesia."` (dengan dua spasi), dan `"Univ. Indonesia"`. Berapa ejaan mentahnya, dan berapa setelah normalisasi?

5. **Di mana normalisasi berhenti.** Nama “Soekarno” dan “Sukarno” merujuk pada orang yang sama dalam dua sistem ejaan bahasa Indonesia. Apa hasil `norm_name` untuk keduanya, dan informasi jenis apa, alih-alih aturan, yang dibutuhkan untuk mencocokkannya?

6. **Membulatkan laju.** Pakai Persamaan (13.6) untuk menghitung pct(1,8), pct(1,3), dan pct(2,3) secara tepat dengan bilangan bulat. Apa yang dikembalikan program ketika $w = 0$, dan mengapa di situ diperlukan penjaga?

7. **Batas penjaga.** Penjaga nama persis mencegah pemeriksa memberi skor kepada orang asing ketika nama itu tidak ada. Apa yang keliru ketika dua peneliti nyata yang berbeda memiliki nama yang persis sama? Sarankan satu kolom rekaman peneliti yang akan membantu, dan jelaskan caranya.

8. **Sebutkan prinsipnya.** Untuk setiap praktik, sebutkan prinsip di Subbab 13.7 yang dilayaninya: (a) mesin difusi lokal TTM ikut bekerja hanya bila setidaknya sama baik dan tidak lebih lambat; (b) sebuah rilis dianggap selesai hanya ketika `/api/health` melaporkan versi baru; (c) Content Engine menghasilkan draf yang disunting dan diterbitkan pengguna; (d) Profitina AI mengembalikan keluaran parsial ketika sebuah tugas tidak dapat selesai; (e) deteksi kategori pada pindaian berkata "belum yakin".

9. **Ambang.** Pemeriksa menyebut cakupan DOI di bawah 90% sebagai celah. Berargumenlah untuk ambang yang berbeda bagi peneliti di bidang yang banyak karyanya terbit dalam buku dan jurnal nasional tanpa DOI, dan katakan apa yang sebaiknya dicetak laporan agar pembaca memahami pilihan itu.

10. **Urutan kata.** Jalankan pemeriksa dengan sampel tersimpan dan kueri `"piwowar, heather"`. Apakah ditemukan? Fungsi mana yang membuat hal itu terjadi?

13.14 Penyelesaian Soal Latihan

Penyelesaian 1. $0.4^2 + 0.4^2 + 0.2^2 = 0.16 + 0.16 + 0.04 = 0.36$, kebalikannya $1/0.36 \approx 2.78$. Lima bobot sama sebesar 0.2: $5 \times 0.04 = 0.20$, kebalikannya 5. Portofolio pertama lebih terkonsentrasi: ia berperilaku seperti kurang dari tiga aset berbobot sama, sedangkan yang kedua seperti tepat lima; indeksnya $0.36/0.20 = 1.8$ kali lebih tinggi.

Penyelesaian 2. Imbal hasil terurut: $-2.1, -1.4, -0.8, -0.6, -0.3, 0.5, 0.7, 0.9, 1.2, 1.5$. Pada 90%, 10% terburuk adalah satu hari, sehingga $q_{10\%} = -2.1$: $\mathrm{VaR}_{90\%} = 2.1\%$, dan rata-rata kerugian di atau melampauinya adalah hari tunggal yang sama, $\mathrm{CVaR}_{90\%} = 2.1\%$. Pada 70%, 30% terburuk adalah tiga hari: $q_{30\%} = -0.8$, sehingga $\mathrm{VaR}_{70\%} = 0.8\%$ dan $\mathrm{CVaR}_{70\%} = (2.1 + 1.4 + 0.8)/3 \approx 1.43\%$. Dengan hari sesedikit ini, kedua ukuran berimpit pada 90%; dengan data lebih banyak keduanya terpisah, itulah sebabnya meja sungguhan membutuhkan riwayat panjang.

Penyelesaian 3. Simpan dua variabel: puncak berjalan P (nilai terbesar yang terlihat sejauh ini) dan *drawdown* terbaik D yang ditemukan sejauh ini, mula-mula $P = V_0$ dan $D = 0$. Untuk setiap nilai berikutnya V: tetapkan $P = \max(P, V)$, lalu $D = \max(D, (P - V)/P)$. Setiap nilai dikunjungi sekali dan hanya dua bilangan yang disimpan, sehingga waktunya $O(n)$ dan memori tambahannya $O(1)$. Pada 100,110,99,105,90,120,108: puncak menjadi 110, *drawdown* pada 99 adalah 10%, pada 105 adalah 4.5%, pada 90 adalah $20/110 \approx 18.18\%$; puncak lalu menjadi 120 dan *drawdown* pada 108 adalah 10%. Jadi MDD $\approx 18.18\%$.

Penyelesaian 4. Kunci-kuncinya adalah `'universitas indonesia'`, `'universitas indonesia depok'`, `'universitas indonesia'`, dan `'univ indonesia'`. Empat ejaan mentah menjadi tiga setelah normalisasi: huruf kapital, titik penutup, dan spasi ganda hilang; kata tambahan "Depok" dan singkatan "Univ." tetap, yang pertama memang semestinya dan yang kedua tidak terhindarkan.

Penyelesaian 5. `norm_name` mengembalikan `'soekarno'` dan `'sukarno'`: dua kunci berbeda, karena perbedaannya terletak pada huruf-hurufnya sendiri, bukan pada huruf besar-kecil, aksen, tanda baca, atau urutan kata. Mencocokkannya membutuhkan pengetahuan: daftar nama alternatif yang tercatat (`display_name_alternatives` di OpenAlex adalah daftar semacam itu

bagi peneliti), pengenal persisten seperti ORCID yang mengikat kedua ejaan pada satu orang, atau aturan khusus ejaan lama bahasa Indonesia (oe menjadi u), yang merupakan pengetahuan yang ditulis sebagai aturan dan harus dipakai hati-hati agar tidak menggabungkan nama yang memang berbeda.

Penyelesaian 6. pct(1,8) = ⌊(200 + 8)/16⌋ = ⌊13⌋ = 13 (12,5% dibulatkan ke atas menjadi 13). pct(1,3) = ⌊(200 + 3)/6⌋ = ⌊33.83⌋ = 33. pct(2,3) = ⌊(400 + 3)/6⌋ = ⌊67.17⌋ = 67. Ketika $w = 0$ rumus itu akan membagi dengan nol; kedua program mengembalikan 0 dalam kasus itu, dan laporan tetap mencetak pecahannya (misalnya `0/0`) agar pembaca melihat bahwa tidak ada yang dapat dihitung, bukan 0% yang sungguhan.

Penyelesaian 7. Jika dua peneliti nyata memiliki nama yang persis sama, keduanya lolos penjaga, dan pemeriksa mengambil kandidat pertama, yang mungkin orang yang salah. Penjaga itu melindungi dari mengarang seseorang, bukan dari tertukarnya dua orang nyata. Kolom ORCID membantu: jika pengguna memberikan ORCID-nya, pemeriksa dapat menerima hanya kandidat yang `orcid`-nya cocok, yang mengenali satu orang secara unik. Institusi terakhir yang diketahui adalah sinyal kedua yang lebih lemah tetapi berguna.

Penyelesaian 8. (a) AI *local-first*, dijaga kejujurannya dengan pengukuran. (b) Tes sebagai gerbang. (c) Aplikasi menyarankan; pengguna memutuskan. (d) Katakan apa yang sebenarnya terjadi ("tidak pernah berhenti diam-diam"). (e) Katakan apa yang sebenarnya terjadi, dan juga aplikasi menyarankan dan pengguna memutuskan, karena pengguna diminta memilih kategorinya.

Penyelesaian 9. Di bidang yang buku, bab buku, dan jurnal nasionalnya sering tanpa DOI, ambang 90% akan menandai hampir semua orang dan tidak mengajarkan apa pun. Ambang seperti 60%, atau laporan yang menampilkan bagiannya tanpa label `ok`/`gap` untuk bidang semacam itu, lebih berguna. Apa pun pilihannya, laporan sebaiknya mencetak ambangnya di samping hasil, misalnya `DOI present: 12/20 (60%) - threshold 60% for this field`, agar pembaca melihat bahwa label itu sebuah pilihan dan dapat tidak menyetujuinya.

Penyelesaian 10. Program mencetak:

```
OpenAlex researcher-record check (a teaching tool, not a ranking)
query : piwowar, heather
source: saved sample (fetched from OpenAlex on 2026-09-24)
match : Heather Piwowar  https://openalex.org/A5048491430
  [ok ] ORCID linked: https://orcid.org/0000-0003-1613-5981
  [ok ] works indexed: 99
  [gap] name spellings: 7 raw, 3 after normalization
  works examined: 5
  [ok ] DOI present: 5/5 (100%)
  [ok ] open access: 5/5 (100%)
  [gap] affiliation strings: 9 raw, 8 after normalization, for 6 institutions
```

Rekaman itu ditemukan. Sumber sampel mengembalikan rekaman karena kuerinya berbagi kata dengannya, dan penjaga menerimanya karena `norm_name` mengurutkan kata-kata: `"piwowar, heather"` dan `"Heather Piwowar"` sama-sama menjadi `'heather piwowar'`.

13.15 Sepatah Kata Penutup

Buku ini bermula di ruang kelas dan berakhir di ruang kelas pula. Bab pertamanya menceritakan bagaimana sebuah platform tumbuh dari puluhan tahun mengajar; lab terakhirnya bertanya kepada katalog publik apakah seorang peneliti dapat ditemukan, dan menolak menjawab ketika tidak tahu. Di antara keduanya, buku ini mencoba memperlihatkan satu hal berkali-kali: bahwa

produk nyata dibangun dari gagasan yang dapat dipahami seorang mahasiswa, sebuah deret geometri di sini, sehimpunan string yang dinormalisasi di sana, sebuah mesin keadaan, sebuah antrean prioritas, sebuah log persetujuan, sebuah gerbang rilis, dan dari disiplin untuk memakainya dengan jujur.

Profitina akan berubah setelah buku ini dicetak. Versinya akan melampaui 0.14.5, whitepaper-nya melampaui v24, jumlah tesnya melampaui 1.577, dan sebagian rencana di Subbab 13.9 akan datang terlambat atau dalam bentuk lain. Memang begitulah semestinya. Yang diharapkan buku ini bertahan adalah cara bekerja di balik angka-angka itu: ukur sebelum percaya, katakan kepada pengguna apa yang sebenarnya terjadi, biarkan manusia memutuskan, dan jangan kirim apa pun sampai tes mengizinkannya.

Jika Anda telah membangun lab-labnya, kini Anda memiliki versi kecil yang bekerja dari gagasan-gagasan di balik sebuah platform yang hidup, dalam dua bahasa, dengan keluaran yang Anda periksa sendiri. Bawalah lebih jauh. Arahkan pemeriksa kesiapan ke situs Anda sendiri, penyaring ke pesan-pesan Anda sendiri, simulator ke skema loyalitas Profitina Laundry, pemeriksa peneliti ke nama Anda sendiri. Ketika sesuatu di dalamnya mengejutkan Anda, atau ketika Anda menemukan kekeliruan di halaman-halaman ini, tulislah ke info@profitina.com. *Bersama, berkembang, berbagi*: slogan itu meminta ketiganya, dan yang terakhir kini menjadi bagian Anda.

Daftar Pustaka

[1] M. T. Nygard, *Release It! Design and Deploy Production-Ready Software*, 2nd ed. Raleigh, NC, USA: Pragmatic Bookshelf, 2018, ch. 5, “Stability Patterns.”

[2] R. T. Rockafellar and S. Uryasev, “Optimization of conditional value-at-risk,” *Journal of Risk*, vol. 2, no. 3, pp. 21–41, 2000.

[3] W. F. Sharpe, “The Sharpe ratio,” *The Journal of Portfolio Management*, vol. 21, no. 1, pp. 49–58, 1994.

[4] FFmpeg Developers, “FFmpeg Documentation.” [Online]. Available: https://ffmpeg.org/documentation.html

[5] S. Lin, A. Wang, and X. Yang, “SDXL-Lightning: Progressive adversarial diffusion distillation,” arXiv:2402.13929, 2024.

[6] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, “Depth Anything V2,” arXiv:2406.09414, 2024.

[7] Profitina, *Profitina Whitepaper — Public Version v24* (ID/EN), Sep. 2026. [Online]. Available: https://app.profitina.com/static/whitepaper/

[8] J. Priem, H. Piwowar, and R. Orr, “OpenAlex: A fully-open index of scholarly works, authors, venues, institutions, and concepts,” arXiv:2205.01833, 2022.

[9] OpenAlex, “API keys required starting Feb 13 (and some new endpoints!),” *openalex-users* mailing list, Jan. 14, 2026. [Online]. Available: https://groups.google.com/g/openalex-users/c/rI1GIAySpVQ

[10] OpenAlex, “Authentication,” *OpenAlex API Reference*. [Online]. Available: https://developers.openalex.org/api-reference/authentication (accessed Sep. 24, 2026).

[11] Unicode Consortium, “Unicode Normalization Forms,” Unicode Standard Annex #15. [Online]. Available: https://www.unicode.org/reports/tr15/

[12] Oracle, “Class Normalizer,” *Java SE 21 & JDK 21 API Specification*, 2023. [Online]. Available: https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/text/Normalizer.html

[13] Oracle, “Class HttpClient,” *Java SE 21 & JDK 21 API Specification*, 2023. [Online]. Available: https://docs.oracle.com/en/java/javase/21/docs/api/java.net.http/java/net/http/HttpClient.html

[14] H. Piwowar, J. Priem, V. Larivière, J. P. Alperin, L. Matthias, B. Norlander, A. Farley, J. West, and S. Haustein, "The state of OA: a large-scale analysis of the prevalence and impact of Open Access articles," *PeerJ*, vol. 6, p. e4375, 2018, doi: 10.7717/peerj.4375.

[15] Python Software Foundation, "http.server — HTTP servers," *The Python Standard Library*. [Online]. Available: https://docs.python.org/3/library/http.server.html