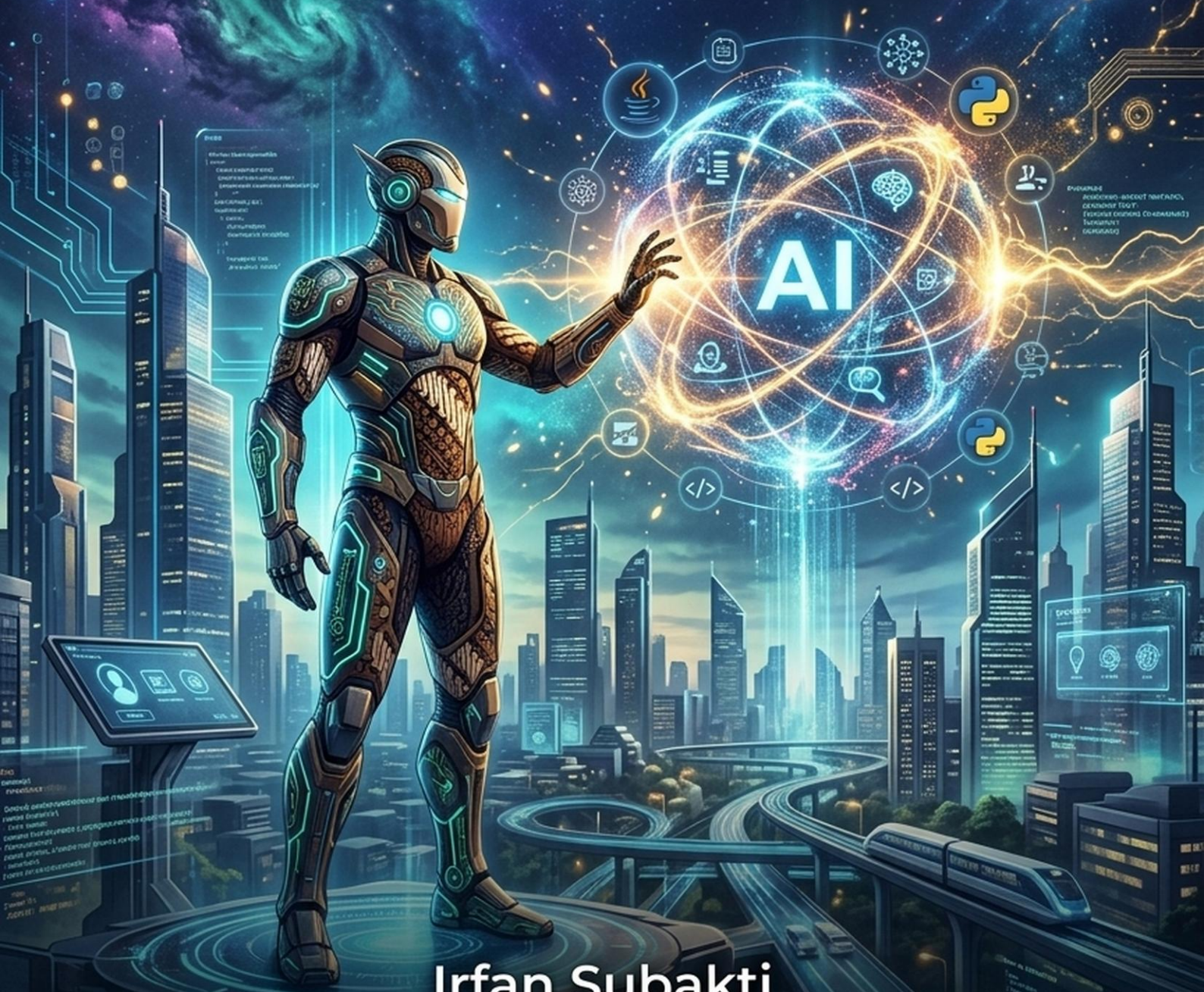
INTERAKSI MANUSIA DAN KOMPUTER

Merancang Antarmuka yang Mudah Dipakai, Aksesibel, dan Siap AI, dengan Java dan Python



Irfan Subakti

Profitina

profitina.com

Interaksi Manusia dan Komputer

Merancang Antarmuka yang Mudah Dipakai, Aksesibel, dan Siap AI, dengan Java dan Python

Irfan Subakti

Profitina

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 ..... **xxii**
Untuk siapa buku ini ditulis xxii
Satu gagasan, dua bahasa xxiii
Isi buku ini xxiii
Cara membacanya xxiv
Tentang penulis xxiv
Ucapan terima kasih xxv

BAB 1 Pengantar IMK: Mengapa Antarmuka Penting **1**
1.1 Warung Bu Ratna Masuk Daring 1
1.2 Apa Itu Interaksi Manusia dan Komputer 2
1.2.1 Sebuah definisi, dan tiga kata 2
1.2.2 Dua jurang 3
1.2.3 Apa yang bukan IMK 4
1.3 Dari Kartu Berlubang hingga Agen: Sejarah Singkat dan Masa Kini 4
1.4 Mengapa Antarmuka Penting di Indonesia pada 2026 6
1.4.1 Siapa yang daring, dan dengan apa 6
1.4.2 Hukum kini menuntut antarmuka yang baik 6
1.4.3 Cara kerja buku ini 7
1.5 Meja Kerja Anda: Apa yang Dipasang dan Mengapa 7
1.6 Memasang di Windows 10 dan 11 8
1.6.1 JDK 8
1.6.2 Python 9
1.6.3 VS Code 9
1.7 Memasang di macOS 9
1.7.1 JDK 9
1.7.2 Python 10
1.7.3 VS Code 10
1.8 Memasang di Linux 10
1.9 VS Code: Dua Ekstensi dan Sebuah Tur 11
1.10 Program-Program Pertama Anda 12
1.10.1 Halo, dalam Java 12
1.10.2 Halo, dalam Python 13
1.10.3 Memeriksa seluruh meja kerja 13
1.11 Program IMK Pertama: Menghitung Biaya Klik 16
1.11.1 Keystroke-Level Model 16
1.11.2 Dua halaman, dihitung dengan tangan 16
1.11.3 Programnya 17
1.11.4 Menjalankannya 21
1.11.5 Mengujinya 22
1.12 IMK dalam Praktik: Profitina 23
1.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 24
1.14 Ringkasan Bab 24
1.15 Soal Latihan 25
1.16 Penyelesaian Soal Latihan 25
Daftar Pustaka 27

BAB 2 Faktor Manusia: Persepsi, Kognisi, Memori, Hukum Fitts dan Hick **30**
2.1 Bu Siti di Kios Puskesmas 30
2.2 Gagasannya dengan Kata-Kata Sederhana: Manusia sebagai Pemroses Informasi 31

2.3 Persepsi: Apa yang Sampai ke Pikiran 32
2.3.1 Penglihatan 32
2.3.2 Pengelompokan: prinsip Gestalt 32
2.3.3 Apa yang menonjol: fitur praatensi 33
2.3.4 Pendengaran dan sentuhan 33
2.4 Perhatian: Satu Hal pada Satu Waktu 34
2.5 Memori: Apa yang Bertahan, dan Berapa Lama 34
2.5.1 Memori kerja 34
2.5.2 Mengenali, bukan mengingat 34
2.5.3 Model mental 35
2.6 Hukum Fitts: Berapa Lama Mencapai Sasaran 35
2.6.1 Hukumnya 35
2.6.2 Contoh yang dikerjakan 36
2.6.3 Arti hukum ini bagi desain 36
2.7 Hukum Hick: Berapa Lama Memutuskan 37
2.7.1 Hukumnya 37
2.7.2 Kapan hukum ini berlaku, dan kapan tidak 37
2.8 Membangun Kalkulatornya Langkah demi Langkah 38
2.8.1 Berkas datanya 38
2.8.2 Mencocokkan garis dengan kuadrat terkecil 39
2.8.3 Waktu harapan sebuah tata letak 40
2.8.4 Programnya 40
2.9 Menjalankannya 46
2.10 Mengujinya 47
2.11 Memakainya Sungguh-sungguh 48
2.12 IMK dalam Praktik: Profitina 49
2.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 50
2.14 Ringkasan Bab 50
2.15 Soal Latihan 51
2.16 Penyelesaian Soal Latihan 51
Daftar Pustaka 52

BAB 3 Prinsip dan Heuristik Desain 54
3.1 Laundry Mbak Dewi Meninggalkan WhatsApp 54
3.2 Gagasannya dengan Kata-Kata Sederhana: Prinsip, Pedoman, dan Heuristik 55
3.3 Prinsip-Prinsip Norman 56
3.3.1 Afordans dan penanda 56
3.3.2 Pemetaan 56
3.3.3 Batasan dan fungsi pemaksa 57
3.3.4 Umpan balik dan model konseptual 57
3.4 Tiga Keluarga Aturan 58
3.5 Prinsip Bertemu Bahasa Desain 2025–2026 59
3.6 Evaluasi Heuristik Langkah demi Langkah 60
3.6.1 Prosedurnya 60
3.6.2 Berapa banyak evaluator? 61
3.6.3 Evaluasi heuristik dan kerabatnya 62
3.7 Membangun Pembuat Laporan 62
3.7.1 Datanya 63
3.7.2 Programnya 64
3.8 Menjalankannya 68
3.9 Mengujinya 69
3.10 Memakainya Sungguh-sungguh 71
3.11 IMK dalam Praktik: Profitina 71
3.12 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 72

3.13 Ringkasan Bab 72
3.14 Soal Latihan 73
3.15 Penyelesaian Soal Latihan 73
Daftar Pustaka 75

BAB 4 Proses Berpusat pada Pengguna dan Design Thinking 76
4.1 Pangkalan Ojek Ingin Punya Aplikasi 76
4.2 Gagasannya dengan Kata-Kata Sederhana 77
4.3 Proses Desain Berpusat Manusia menurut ISO 9241-210 77
4.4 Double Diamond dan Design Thinking 79
4.4.1 Double Diamond 79
4.4.2 Design thinking 79
4.4.3 Lean UX dan tim *agile* 80
4.5 Alat-Alat Kerja, dengan Templat 80
4.5.1 Diagram afinitas 80
4.5.2 Persona 81
4.5.3 Peta perjalanan 81
4.5.4 Peta empati dan *jobs to be done* 82
4.5.5 Pernyataan masalah dan pertanyaan "bagaimana kita bisa" 82
4.5.6 Ideasi dan matriks dampak-upaya 82
4.5.7 Kebutuhan pengguna yang terukur 83
4.6 AI dalam Proses, 2026 83
4.7 Membangun Alat Sintesis 83
4.7.1 Datanya 84
4.7.2 Programnya 85
4.8 Menjalankannya 91
4.9 Mengujinya 93
4.10 Memakainya Sungguh-sungguh 94
4.11 IMK dalam Praktik: Profitina 95
4.12 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 95
4.13 Ringkasan Bab 96
4.14 Soal Latihan 96
4.15 Penyelesaian Soal Latihan 97
Daftar Pustaka 98

BAB 5 Metode Riset Pengguna 100
5.1 Pelanggan Bu Rina yang Menghilang 100
5.2 Gagasannya dalam Bahasa Sederhana 101
5.2.1 Mencari tahu sebelum membangun 101
5.2.2 Apa yang dikatakan dan apa yang dilakukan 101
5.2.3 Empat pertanyaan untuk memilih metode 102
5.3 Merencanakan Studi 103
5.3.1 Rencana riset satu halaman 103
5.3.2 Siapa yang diajak bicara, dan berapa banyak 104
5.3.3 Persetujuan dan data pribadi 105
5.4 Wawancara dan Inkuiri Kontekstual 105
5.4.1 Wawancara semi-terstruktur 105
5.4.2 Bertanya tanpa menggiring 106
5.4.3 Inkuiri kontekstual: guru dan murid magang 106
5.4.4 Catatan lapangan dan studi buku harian 107
5.5 Survei dan Kuesioner 107
5.5.1 Kapan survei menjadi alat yang tepat 107
5.5.2 Menulis butir yang baik 107
5.5.3 Seberapa teliti sebuah survei? 108
5.6 Membangun Perangkum Survei 109

5.6.1 Datanya 109
5.6.2 Programnya dalam Python 109
5.6.3 Programnya dalam Java 111
5.6.4 Menjalankannya 112
5.6.5 Membaca hasilnya 113
5.7 Memaknai Data Kualitatif 113
5.7.1 Dari catatan ke kode 113
5.7.2 Diagram afinitas 114
5.7.3 Kapan wawancara sudah cukup? 114
5.7.4 Program saturasi 115
5.8 Menguji Program 120
5.9 Dari Temuan ke Wawasan 121
5.9.1 Triangulasi bukti 121
5.9.2 Menulis pernyataan wawasan 122
5.9.3 Apa yang dilakukan Bu Rina 122
5.10 AI dalam Riset Pengguna, per 2026 122
5.11 Menerapkannya dalam Kenyataan 123
5.12 IMK dalam Praktik: Profitina 124
5.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 125
5.14 Ringkasan Bab 125
5.15 Soal Latihan 126
5.16 Penyelesaian Soal Latihan 126
Daftar Pustaka 128

BAB 6 Prototipe: dari Kertas hingga Interaktif (dan Berbantuan AI) 130
6.1 Antrean Sebelum Fajar 130
6.2 Gagasannya dalam Bahasa Sederhana 131
6.2.1 Prototipe adalah pertanyaan 131
6.2.2 Sketsa dan prototipe; hal yang tepat dan hal yang benar 131
6.2.3 Prototipe horizontal dan vertikal 132
6.3 Prototipe Kertas 132
6.3.1 Bahan dan peran 132
6.3.2 Sebuah sesi di puskesmas 133
6.3.3 Wizard of Oz 133
6.4 Memeriksa Alur Sebelum Membangunnya 133
6.4.1 Alur sebagai berkas teks 133
6.4.2 Pemeriksanya 134
6.4.3 Menjalankannya pada kedua versi 138
6.5 Wireframe dan Mockup yang Dapat Diklik 140
6.5.1 Wireframe 140
6.5.2 Mockup yang dapat diklik dan alat-alat tahun 2026 140
6.6 Membangun Prototipe Berkode 140
6.6.1 Apa yang harus dilakukannya 140
6.6.2 Layar-layarnya 141
6.6.3 Servernya dalam Python 143
6.6.4 Prototipe yang sama dalam Java 144
6.6.5 Menjalankannya dan mengambil tangkapan layar 149
6.7 Menguji Kode Prototipe 151
6.8 Prototipe Berbantuan AI pada 2026 153
6.8.1 Apa yang dilakukan alat-alat itu 153
6.8.2 Memakai AI tanpa melewatkan pertanyaan 154
6.9 Membuat Prototipe Pesan Laundry 154
6.10 Menerapkannya dalam Kenyataan 155
6.11 IMK dalam Praktik: Profitina 155
6.12 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 156

6.13 Ringkasan Bab 156
6.14 Soal Latihan 157
6.15 Penyelesaian Soal Latihan 157
Daftar Pustaka 159

BAB 7 Evaluasi dan Pengujian Usabilitas 160
7.1 Pesan Dulu di Warung Bu Tini 160
7.2 Gagasannya dalam Bahasa Sederhana 161
7.2.1 Apa arti "mudah dipakai" 161
7.2.2 Dua keluarga evaluasi 161
7.3 Evaluasi Heuristik 161
7.3.1 Metodenya 161
7.3.2 Tingkat keparahan 162
7.4 Uji Usabilitas dengan Pengguna Sungguhan 162
7.4.1 Merencanakan uji 162
7.4.2 Berpikir dengan bersuara 163
7.4.3 Apa yang terjadi 163
7.5 Mengukur Kinerja 164
7.5.1 Keberhasilan tugas dan selang kepercayaannya 164
7.5.2 Waktu pengerjaan dan rata-rata geometrik 164
7.5.3 Kesalahan 165
7.5.4 Hasilnya 165
7.6 Mengukur Kepuasan: SUS dan UMUX-Lite 165
7.6.1 System Usability Scale 165
7.6.2 UMUX-Lite 166
7.6.3 Hasilnya 166
7.7 Membangun Program Metrik 167
7.8 Berapa Banyak Partisipan? 174
7.8.1 Model penemuan masalah 174
7.8.2 Menaksir *p* dari studi 174
7.9 Menguji Program 176
7.10 Uji A/B 177
7.10.1 Dari delapan orang ke ratusan 177
7.10.2 Uji-z dua proporsi 178
7.10.3 Berapa pengunjung yang dibutuhkan? 178
7.10.4 Programnya 179
7.11 Evaluasi pada 2026: Jarak Jauh, Tanpa Pemandu, dan Berbantuan AI 182
7.12 Menerapkannya dalam Kenyataan 182
7.13 IMK dalam Praktik: Profitina 183
7.14 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 184
7.15 Ringkasan Bab 184
7.16 Soal Latihan 185
7.17 Penyelesaian Soal Latihan 185
Daftar Pustaka 186

BAB 8 Desain Visual, Design System, dan Arsitektur Informasi 188
8.1 Aplikasi Sebuah Bank Perkreditan Rakyat 188
8.2 Gagasannya dalam Bahasa Sederhana 189
8.3 Melihat Kelompok: Gestalt dan Hierarki Visual 189
8.3.1 Prinsip-prinsip Gestalt 189
8.3.2 Hierarki visual 190
8.4 Tipografi dan Tata Letak 190
8.4.1 Huruf yang dapat dibaca 190
8.4.2 Skala huruf modular 191
8.4.3 Ruang dan grid 191

8.5 Warna dan Kontras ... 191
8.5.1 Mengapa kontras diukur, bukan dinilai ... 191
8.5.2 Luminans relatif dan rasio kontras ... 192
8.6 Membangun Pemeriksa Kontras ... 192
8.6.1 Warna sebagai *design token* ... 192
8.6.2 Programnya dalam Python ... 193
8.6.3 Programnya dalam Java ... 195
8.6.4 Menjalankannya ... 197
8.7 *Design System* dan Token ... 199
8.7.1 Dari keputusan menjadi sistem ... 199
8.7.2 Bahasa desain tahun 2026 ... 199
8.7.3 Apa yang sebaiknya dilakukan tim kecil ... 200
8.8 Arsitektur Informasi ... 200
8.8.1 Mengorganisasi, memberi label, menavigasi, mencari ... 200
8.8.2 *Card sorting* dan *tree testing* ... 201
8.9 Menganalisis *Card Sorting* ... 201
8.9.1 Kemiripan antarkartu ... 201
8.9.2 Pengelompokan *average linkage* ... 201
8.9.3 Programnya ... 202
8.9.4 Membaca hasilnya ... 208
8.10 Menguji Program ... 209
8.11 Menerapkannya dalam Kenyataan ... 211
8.12 IMK dalam Praktik: Profitina ... 212
8.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya ... 212
8.14 Ringkasan Bab ... 213
8.15 Soal Latihan ... 213
8.16 Penyelesaian Soal Latihan ... 214
Daftar Pustaka ... 215

BAB 9 Gaya Interaksi: GUI, Sentuh, Suara, Gestur ... 217
9.1 Meja Kasir Bu Tini ... 217
9.2 Gagasannya dalam Bahasa Sehari-hari ... 218
9.2.1 Apa itu gaya interaksi ... 218
9.2.2 Peta gaya-gaya interaksi ... 218
9.2.3 Contoh yang dikerjakan: segelas teh, lima cara ... 219
9.3 Teori dan Buktinya ... 219
9.3.1 Dua jurang, dan mengapa manipulasi langsung mempersempitnya ... 219
9.3.2 Peristiwa, keadaan, dan kecepatan umpan balik ... 220
9.3.3 Sentuhan: jari bukan penunjuk tetikus ... 220
9.3.4 Gestur: ampuh, tetapi tak terlihat ... 221
9.3.5 Suara: alami, tetapi ambigu ... 222
9.3.6 Bahasa desain masa kini, dan interaksi di ruang ... 223
9.4 Metode Praktis: Memilih dan Memeriksa Gaya ... 223
9.5 Membangunnya Langkah demi Langkah ... 224
9.5.1 Menu dan inti pesanan ... 225
9.5.2 Baris perintah dan baris suara yang diketik ... 232
9.5.3 Kios sentuh ... 235
9.5.4 Mengukur setiap target di peramban sungguhan ... 241
9.6 Menjalankannya ... 242
9.7 Mengujinya ... 243
9.8 Membuatnya Lebih Baik ... 245
9.8.1 Target yang dapat dikenai semua orang ... 245
9.8.2 Bertanya pada saat yang tepat ... 246
9.9 Memakainya Sungguhan ... 246
9.10 IMK dalam Praktik: Profitina ... 246

9.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 247
9.12 Ringkasan Bab 247
9.13 Soal Latihan 248
9.14 Penyelesaian Soal Latihan 248
Daftar Pustaka 250

BAB 10 UX Seluler dan Responsif 252
10.1 Ponsel Mas Yoga 252
10.2 Gagasannya dalam Bahasa Sehari-hari 253
10.3 Teori dan Buktinya 253
10.3.1 Kenyataan seluler Indonesia, dibaca dengan cermat 253
10.3.2 Piksel, viewport, dan mengapa halaman desktop menciut 254
10.3.3 Tata letak cair dan titik henti yang dipilih konten 254
10.3.4 Tangan dan ibu jari 255
10.3.5 Formulir di ponsel 255
10.3.6 Kecepatan: *vitals*, perjalanan pulang-pergi, dan anggaran 256
10.3.7 Kriteria WCAG yang paling terasa di ponsel 256
10.4 Metode Praktis: Daftar Periksa *Mobile-First* 257
10.5 Membangunnya Langkah demi Langkah 257
10.5.1 Dua templat, satu perangkat kata 257
10.5.2 Mengisi dan melayani halaman 260
10.5.3 Peramban yang berpura-pura menjadi ponsel 263
10.5.4 Perkiraan waktu muat 264
10.6 Menjalankannya 266
10.7 Mengujinya 268
10.8 Membuatnya Lebih Baik 269
10.9 Memakainya Sungguhan 270
10.10 IMK dalam Praktik: Profitina 270
10.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 271
10.12 Ringkasan Bab 271
10.13 Soal Latihan 271
10.14 Penyelesaian Soal Latihan 272
Daftar Pustaka 273

BAB 11 Aksesibilitas dan Desain Inklusif (WCAG 2.2) 274
11.1 Suatu Pagi di Puskesmas Sukamaju 274
11.2 Gagasannya dalam Bahasa Sehari-hari 275
11.3 Teori dan Buktinya 275
11.3.1 Siapa yang terdampak 275
11.3.2 Apa yang diwajibkan hukum 275
11.3.3 WCAG 2.2 dalam satu halaman 276
11.3.4 Kontras, dihitung 277
11.3.5 Cara teknologi bantu membaca halaman 278
11.3.6 Wajah web pada 2026 279
11.4 Metode Praktis: Empat Lapis Pengujian 279
11.5 Membangunnya Langkah demi Langkah 280
11.5.1 Halaman kios pertama 280
11.5.2 Pemeriksa kontras 281
11.5.3 Audit 283
11.5.4 Halaman yang diperbaiki 288
11.6 Menjalankannya 289
11.7 Mengujinya 291
11.8 Membuatnya Lebih Baik 293
11.9 Memakainya Sungguhan 293
11.10 IMK dalam Praktik: Profitina 294

11.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 294
11.12 Ringkasan Bab 295
11.13 Soal Latihan 295
11.14 Penyelesaian Soal Latihan 296
Daftar Pustaka 297

BAB 12 Desain Emosional, Kepercayaan, dan UX yang Etis 298
12.1 Keanggotaan Bu Sri 298
12.2 Gagasannya dalam Bahasa Sehari-hari 299
12.3 Teori dan Buktinya 299
12.3.1 Tiga tingkat desain emosional 299
12.3.2 Bagaimana kepercayaan dibangun dan hilang 300
12.3.3 Katalog *dark pattern* 301
12.3.4 Apa kata hukum 302
12.3.5 Mengukur perasaan 303
12.3.6 Persuasi yang menghormati pengguna 303
12.4 Metode Praktis: Tinjauan Alur yang Adil 303
12.5 Membangunnya Langkah demi Langkah 304
12.5.1 Menguraikan sebuah alur 304
12.5.2 *Linter*-nya 304
12.5.3 Alur kedua 307
12.5.4 Buku besar persetujuan 308
12.6 Menjalankannya 310
12.7 Mengujinya 311
12.8 Membuatnya Lebih Baik 312
12.9 Memakainya Sungguhan 313
12.10 IMK dalam Praktik: Profitina 314
12.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 314
12.12 Ringkasan Bab 315
12.13 Soal Latihan 315
12.14 Penyelesaian Soal Latihan 316
Daftar Pustaka 317

BAB 13 Merancang Antarmuka AI: Chat, Kopilot, dan Agen 319
13.1 Pesan Tengah Malam Bu Wati 319
13.2 Gagasannya dengan Kata-Kata Sederhana 320
13.2.1 Chat, kopilot, dan agen 320
13.2.2 Empat pertanyaan yang harus dijawab setiap antarmuka AI 320
13.2.3 Contoh yang dikerjakan dengan tangan 321
13.3 Teori dan Buktinya 322
13.3.1 Pedoman interaksi manusia-AI 322
13.3.2 Tingkat otomatisasi dan tangga otonomi 322
13.3.3 Kepercayaan, ketergantungan, dan kalibrasi 323
13.3.4 Agen pada 2025 dan 2026 324
13.3.5 Aturan pada 2026: keterbukaan dan persetujuan 324
13.4 Metode: Kanvas Interaksi AI 325
13.5 Membangun Asisten Pesanan 326
13.5.1 Program dalam Python 326
13.5.2 Program dalam Java 329
13.5.3 Menjalankannya 333
13.6 Mengukur Ketergantungan dan Kalibrasi 335
13.7 Mengujinya 339
13.8 Membuatnya Lebih Baik 341
13.9 Memakainya Sungguhan 341
13.10 IMK dalam Praktik: Profitina 342

13.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 342
13.12 Ringkasan Bab 343
13.13 Soal Latihan 343
13.14 Penyelesaian Soal Latihan 344
Daftar Pustaka 345

BAB 14 UX Berbasis Data: Analitik, Eksperimen, dan Metrik 347
14.1 Checkout Bu Tini yang Sepi 347
14.2 Gagasannya dengan Kata-Kata Sederhana 348
14.2.1 Apa yang dikatakan orang dan apa yang dilakukannya 348
14.2.2 Peristiwa, corong, dan metrik 348
14.2.3 Contoh yang dikerjakan dengan tangan 348
14.3 Teori dan Buktinya 349
14.3.1 Memilih metrik: kerangka HEART 349
14.3.2 Di mana penggunanya: Indonesia pada 2026 349
14.3.3 Eksperimen terkendali 350
14.3.4 Eksperimen yang tepercaya: jebakan-jebakannya 351
14.3.5 Data dengan persetujuan 351
14.3.6 Metrik untuk fitur AI 352
14.4 Metode: Rencana Eksperimen 352
14.5 Membangun Analisis Corong 353
14.5.1 Log peristiwa 353
14.5.2 Corong dalam Python dan Java 354
14.5.3 Menjalankannya 356
14.6 Membangun Pemeriksaan A/B 357
14.6.1 Program dalam Python 357
14.6.2 Program dalam Java 359
14.6.3 Menjalankannya pada log yang sehat 361
14.6.4 Menjalankannya pada log yang rusak 361
14.7 Mengujinya 362
14.8 Membuatnya Lebih Baik 363
14.9 Memakainya Sungguhan 364
14.10 IMK dalam Praktik: Profitina 364
14.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 365
14.12 Ringkasan Bab 365
14.13 Soal Latihan 366
14.14 Penyelesaian Soal Latihan 366
Daftar Pustaka 367

BAB 15 Antarmuka Masa Depan: Spasial, Wearable, dan Ambient 369
15.1 Mbah Sumi dan Layar Antrean 369
15.2 Gagasannya dengan Kata-Kata Sederhana 370
15.2.1 Melampaui persegi panjang 370
15.2.2 Ukuran adalah sudut 370
15.2.3 Contoh yang dikerjakan dengan tangan 371
15.3 Teori dan Buktinya 371
15.3.1 Perangkat 2025 dan 2026 371
15.3.2 Masukan di ruang: memandang bukan memilih 372
15.3.3 Wearable: lirik, ketuk, lanjut 373
15.3.4 Teknologi ambient dan tenang 373
15.3.5 Kenyamanan, keselamatan, privasi, dan aksesibilitas 374
15.3.6 Gagasan spasial di layar datar 374
15.4 Metode: Daftar Periksa Konteks Pemakaian 375
15.5 Membangun Kalkulator Ukuran Target 375
15.6 Memilih Waktu Tatap 378

15.6.1 Log pandangan simulasi 378
15.6.2 Analisis dalam Python dan Java 378
15.6.3 Membaca hasilnya 380
15.7 Mengujinya 381
15.8 Membuatnya Lebih Baik 383
15.9 Memakainya Sungguhan 383
15.10 IMK dalam Praktik: Profitina 384
15.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 384
15.12 Ringkasan Bab 385
15.13 Soal Latihan 385
15.14 Penyelesaian Soal Latihan 386
Daftar Pustaka 387

BAB 16 Proyek Akhir: Merancang dan Mengevaluasi Sistem Interaktif Utuh 388
16.1 Kios untuk Puskesmas Sehat Selalu 388
16.2 Gagasannya dengan Kata-Kata Sederhana 389
16.2.1 Sistem interaktif yang utuh 389
16.2.2 Setiap bab sebagai keputusan 389
16.3 Menemukan: Riset di Loket Pendaftaran 390
16.4 Merumuskan: Kebutuhan dan Batasan 391
16.4.1 Kebutuhan 391
16.4.2 Anggaran faktor manusia 391
16.4.3 Aksesibilitas, hukum, dan etika sebagai batasan 392
16.5 Mengembangkan: dari Kertas ke Kios yang Berjalan 393
16.5.1 Prototipe kertas 393
16.5.2 Keputusan desain 393
16.6 Membangun Kios 394
16.6.1 Kios dalam Python 394
16.6.2 Kios dalam Java 396
16.6.3 Menjalankannya 400
16.7 Mengevaluasi Kios 402
16.7.1 Uji rintis 402
16.7.2 Ukuran-ukurannya 403
16.7.3 Program evaluasi 403
16.7.4 Hasilnya 408
16.8 Iterasi: dari Temuan ke Perubahan 409
16.9 Menguji Kode 410
16.10 Menyajikan Keputusan Desain 411
16.11 Memakainya Sungguhan 412
16.12 IMK dalam Praktik: Profitina 412
16.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya 412
16.14 Ringkasan Bab 413
16.15 Soal Latihan 414
16.16 Penyelesaian Soal Latihan 414
Daftar Pustaka 415

Daftar Gambar

Gambar 1.1 Dua jurang Norman: eksekusi (bagaimana caranya?) dan evaluasi (berhasilkah?), serta mutu desain yang menjembatani masing-masing. 3
Gambar 1.2 Bagaimana program Java dan program Python dijalankan. Java dikompilasi sekali oleh `javac` menjadi *bytecode*, yang dijalankan oleh Java Virtual Machine; Python dibaca dan dijalankan oleh interpreter setiap kali. 8
Gambar 1.3 Bagian-bagian Visual Studio Code yang dipakai di buku ini: (1) Activity Bar, (2) Explorer, (3) editor, (4) tombol Run, (5) terminal terpadu, (6) Status Bar. 11
Gambar 1.4 Prediksi waktu satu pesanan pada kedua halaman, dipecah per operator: menunjuk (P) mendominasi desain A. 17
Gambar 2.1 Model Human Processor: prosesor persepsi, kognitif, dan motorik dengan waktu siklus yang khas, penyimpan sensorik yang singkat, dan dua jenis memori. 31
Gambar 2.2 Empat prinsip Gestalt: kedekatan (dua kelompok karena jarak), kemiripan (kolom karena warna), wilayah bersama (pasangan karena bingkai), dan kesinambungan (dua garis mulus, bukan zig-zag) 33
Gambar 2.3 Layar pertama kios antrean yang lama dan yang baru, sesuai skala, dengan porsi pasien setiap layanan dan titik diam tangan. 38
Gambar 2.4 Log latihan dengan garis hasil pencocokan: hukum Fitts (kiri) dan hukum Hick (kanan). 47
Gambar 3.1 Halaman pesanan Profitina Laundry sebelum dan sesudah evaluasi heuristik, dirender dari halaman pengajaran kita sendiri; penanda bernomor menunjuk masalah P01, P02, P03, P07, dan P14 pada **Kode 3.1**. 55
Gambar 3.2 Pemetaan pada kompor: kenop yang berjajar memberi pemetaan sembarang, kenop dalam persegi yang sama dengan tungkunya memberi pemetaan alami. 57
Gambar 3.3 Bagian masalah yang ditemukan oleh i evaluator, dari Persamaan (3.1), untuk $\lambda = 0.464$ (evaluasi ini) dan $\lambda = 0.31$ (rata-rata Nielsen dan Landauer) 62
Gambar 4.1 Aktivitas desain berpusat manusia dalam ISO 9241-210: rencanakan, lalu berputar melalui konteks, kebutuhan, desain, dan evaluasi hingga solusinya memenuhi kebutuhan pengguna. 78
Gambar 4.2 Double Diamond: discover dan define di ruang masalah, develop dan deliver di ruang solusi; setiap berlian mula-mula melebar, lalu menyempit. 79
Gambar 4.3 Garis emosi peta perjalanan: rata-rata keluhan per tahap dari 37 catatan (sumbunya dibalik, sehingga kurva turun di tempat yang menyakitkan; pita berarsir menandai keluhan berat). 92
Gambar 4.4 Kesepuluh gagasan dalam matriks dampak-upaya, dinomori sesuai urutan pada **Kode 4.3**; ukuran setiap lingkaran menunjukkan banyaknya suara. 92
Gambar 5.1 Lanskap metode riset pengguna: metode perilaku (atas) mengamati apa yang dilakukan orang, metode sikap (bawah) mencatat apa yang dikatakan orang; metode kualitatif (kiri) menjelaskan mengapa, metode kuantitatif (kanan) menghitung berapa banyak. 102
Gambar 5.2 Enam butir survei sebagai diagram batang bertumpuk divergen (60 responden); angka menunjukkan persentase responden, dan butir Q2 paling condong ke sisi tidak setuju. 113
Gambar 5.3 Kode baru per wawancara (batang) dan kode berbeda yang ditemukan sejauh ini (garis) dalam studi laundry; aturan Persamaan (5.4) menyatakan saturasi pada wawancara 7. 119
Gambar 6.1 Spektrum fidelitas: dari sketsa yang bertanya apakah ini hal yang tepat sampai prototipe berkode yang bertanya apakah hal itu sudah benar; alat berbantuan AI mencapai fidelitas tinggi dengan cepat, sehingga pertanyaan di sisi kiri mudah terlewat. 132
Gambar 6.2 Alur kios sebelum dan sesudah uji kertas: versi 1 memiliki layar tak terjangkau dan jalan buntu serta membutuhkan empat atau lima ketukan menuju tiket; versi 2 membutuhkan tiga, dan setiap layar mempunyai jalan kembali. 139
Gambar 6.3 Tiga layar prototipe kios yang sedang berjalan, ditangkap oleh `shoot.py`: layar sambutan, pilihan layanan, dan tiket dengan banyaknya orang di depan serta perkiraan waktu tunggu. 150

Gambar 7.1 Skor SUS kedelapan partisipan terhadap pita nilai berkurva; rata-ratanya (70,3) sedikit di atas produk rata-rata (68), tetapi kedua partisipan lanjut usia berada di pita F. 167
Gambar 7.2 Sembilan masalah dalam uji warung dan partisipan yang mengalami masing-masing; titik berwarna yang lebih besar menandai kejadian, diwarnai menurut keparahan, dan P3 serta P6 mengalami masalah terbanyak. 175
Gambar 7.3 Bagian masalah yang diharapkan ditemukan seiring bertambahnya partisipan, dari $1-(1-p)n$: dengan p sekitar 0,3, lima partisipan menemukan sekitar 85%; dengan masalah yang lebih jarang ($p = 0.15$), lima partisipan hanya menemukan sekitar 56%. 175
Gambar 8.1 Empat prinsip Gestalt yang dipakai dalam desain antarmuka: kedekatan, kesamaan, wilayah bersama, dan ketertutupan. 189
Gambar 8.2 Sepuluh pasangan warna aplikasi bank sebagai contoh warna, dengan pemakaian, rasio kontras, dan hasil WCAG 2.2 AA. 198
Gambar 8.3 *Design token* dalam *design system*: token primitif memuat nilai, token semantik menamai tujuan, komponen memakai token semantik, dan alat *build* mengubah satu berkas token menjadi kode untuk setiap platform. 199
Gambar 8.4 Dendrogram *card sorting* terbuka (20 partisipan, 15 kartu, *average linkage*); memotong pada kemiripan 0,5 menghasilkan empat kelompok, dan angsuran pinjaman bergabung dengan tagihan. 208
Gambar 9.1 Satu inti pesanan dengan tiga antarmuka: gaya perintah, gaya suara yang diketik beserta aturan konfirmasinya, dan kios manipulasi langsung semuanya memakai menu, keranjang, dan total yang sama. 224
Gambar 9.2 Kios akhir pada lebar 360 piksel CSS setelah mengetuk "Es teh" dua kali dan "Nasi goreng" sekali: dua kolom tombol menu setinggi 64 piksel, tombol minus 48 piksel pada setiap baris pesanan, dan tautan bahasa dengan bidang sentuh 48 piksel. 245
Gambar 10.1 Kedua halaman pemesanan di ponsel 360 piksel: (a) halaman desktop, ditata pada 980 piksel dan diciutkan agar muat, dan (b) halaman *mobile-first* dengan teks yang terbaca, ruas berlabel setinggi 48 piksel, serta bilah Telepon dan WhatsApp dalam jangkauan ibu jari. 267
Gambar 10.2 Perkiraan waktu tiba halaman pemesanan versi pertama (2.492 kB) dan versi ramping (76 kB) pada tiga jaringan, dari Persamaan (10.4) dengan empat perjalanan pulang-pergi; garis putus-putus adalah anggaran 2,5 detik untuk LCP yang baik. Sumbu waktu berskala logaritmis. 268
Gambar 11.1 Tujuh pasangan warna kios beserta rasio kontrasnya menurut Persamaan (11.3): tiga gagal memenuhi ambang WCAG 2.2 AA (4,5:1 untuk teks, 3:1 untuk teks besar dan bagian kontrol). 290
Gambar 11.2 Halaman kios yang diperbaiki pada 360 piksel CSS setelah tiga kali menekan Tab: tombol sungguhan dengan teks yang terlihat, ruas kartu berlabel dengan petunjuk dan bingkai 3:1, serta cincin fokus amber 4 piksel pada tombol yang terfokus. 291
Gambar 12.1 Tiga tingkat desain emosional Norman yang diterapkan pada Laundry Wangi: kesan pertama yang viseral, pengalaman behavioral saat memesan dan melacak, serta penilaian reflektif tentang kejujuran yang menentukan apakah pelanggan merekomendasikan laundry itu. 300
Gambar 12.2 Kedua layar persetujuan di ponsel: (a) versi pertama, dengan spanduk merah, hitung mundur yang mulai lagi, satu kotak tercentang untuk empat tujuan, tombol "setuju" yang besar, dan tautan kecil yang mempermalukan; (b) versi kedua, dengan satu pilihan per tujuan, tujuan opsional mati, penjelasan yang lugas, dan dua tombol setara. 311
Gambar 13.1 Tangga otonomi: risiko dan keterbalikan tiap tindakan menentukan seberapa banyak AI boleh berbuat tanpa bertanya, dengan contoh dari laundry untuk setiap anak tangga. 323
Gambar 13.2 Kalibrasi model label perawatan hasil simulasi: titik di bawah diagonal berarti model mengklaim keyakinan melebihi yang pantas; garis merah adalah selisih yang dirata-ratakan ECE. 339
Gambar 14.1 Corong checkout pada eksperimen simulasi: proporsi pengguna pembuka menu yang mencapai setiap langkah, untuk checkout lama (A) dan baru (B). 357
Gambar 15.1 Sudut pandang θ benda berukuran s pada jarak d: sudut yang sama dapat dihasilkan benda kecil yang dekat atau benda besar yang jauh. 370
Gambar 15.2 Tawar-menawar waktu tatap pada log pandangan simulasi: aktivasi keliru turun dan pilihan terlewat naik seiring bertambahnya waktu tatap; biaya terendah menurut Persamaan (15.4) ada pada 700 ms. 381

Gambar 16.1 Jalur proyek akhir: empat fase, bab-bab yang metodenya diterapkan di setiap fase, dan putaran balik yang mengubah satu desain menjadi proses iteratif. 389
Gambar 16.2 Kios yang berjalan, ditangkap dengan Playwright: layar utama dan layar nomor. Penanda: 1 bahasa dan ukuran huruf, 2 catatan bantuan darurat, 3 keempat layanan, 4 nomor dan jumlah orang di depan dalam wilayah status. 393
Gambar 16.3 Mode huruf besar setelah satu kali menekan tombol Tab: tautan lompat muncul dengan cincin fokus biru tua, dan teks mengalir ulang. 402
Gambar 16.4 Skor SUS dalam uji rintis, diwarnai menurut kelompok usia, dengan rerata (72,5) dan rerata acuan yang lazim, 68; ketiga partisipan berusia 60 tahun ke atas berada di bawah acuan. 409

Daftar Tabel

Tabel 1.1 Era-era interaksi manusia dan komputer dan apa yang dituntut masing-masing dari penggunanya. 5
Tabel 1.2 Alat-alat di meja kerja dan versi yang dipakai di seluruh buku. 7
Tabel 1.3 Operator Keystroke-Level Model dan waktu satuan bakunya. 16
Tabel 2.1 Ukuran minimum sasaran sentuh dalam pedoman terkini. 37
Tabel 2.2 Daftar periksa faktor manusia untuk meninjau sebuah layar. 49
Tabel 3.1 Sepuluh heuristik usability Nielsen, dengan kata-kata sederhana, beserta contoh dari halaman pesanan pertama Profitina Laundry. 59
Tabel 3.2 Skala keparahan Nielsen untuk masalah usability. 61
Tabel 3.3 Templat pencatatan untuk evaluasi heuristik. 71
Tabel 4.1 Enam prinsip desain berpusat manusia dalam ISO 9241-210:2019, beserta artinya di pangkalan ojek. 78
Tabel 4.2 Siklus yang sama dalam tiga kosakata, beserta metode yang lazim untuk setiap langkah. 80
Tabel 4.3 Persona yang dibangun dari catatan riset, beserta implikasi desain yang disarankannya. 81
Tabel 4.4 Rencana lokakarya sintesis satu hari beserta keluarannya. 94
Tabel 5.1 Memilih metode pertama berdasarkan pertanyaan riset, dengan ukuran sampel dan durasi yang lazim untuk tim kecil. 103
Tabel 5.2 Rencana riset satu halaman untuk studi Laundry Bersih. 104
Tabel 5.3 Jebakan wawancara dan pertanyaan yang lebih baik untuk studi laundry. 106
Tabel 7.1 Skala keparahan masalah usabilitas, dengan contoh dari halaman pemesanan. 162
Tabel 9.1 Lima gaya interaksi utama, apa yang dituntut masing-masing dari pengguna, dan di mana masing-masing muncul di meja kasir Bu Tini. 219
Tabel 9.2 Ukuran minimum target sentuh dan penunjuk per September 2026 (KK = kriteria keberhasilan). 221
Tabel 9.3 Lembar kerja untuk memilih dan memeriksa gaya interaksi, diisi untuk kios warung. 224
Tabel 10.1 Daftar periksa *mobile-first*, dengan hasil untuk versi pertama halaman pemesanan. 257
Tabel 11.1 Sembilan kriteria keberhasilan yang ditambahkan pada WCAG 2.2. 277
Tabel 11.2 Enam kegagalan paling umum dalam WebAIM Million 2026, kriteria WCAG 2.2 di baliknya, dan pemeriksaan oleh program-program bab ini. 280
Tabel 12.1 *Dark pattern* pada alur pertama Laundry Wangi, mengapa masing-masing tidak adil, dan alternatifnya yang adil. 301
Tabel 12.2 Tinjauan alur yang adil: pertanyaan-pertanyaan yang harus dijawab "ya" sebelum merilis alur pendaftaran, langganan, atau persetujuan. 304
Tabel 13.1 Empat pertanyaan pengguna tentang sistem AI mana pun, dan cara asisten pesanan menjawab masing-masing. 321
Tabel 13.2 Sebagian dari 18 pedoman interaksi manusia-AI dan cara masing-masing membentuk asisten pesanan. 322
Tabel 13.3 Kanvas interaksi AI, diisi untuk asisten pesanan laundry. 325

Tabel 14.1 Proses Goals–Signals–Metrics dari HEART diterapkan pada situs pemesanan warung; keterlibatan sengaja ditinggalkan, karena waktu yang lebih lama di halaman pemesanan bukanlah tujuan. 349
Tabel 14.2 Rencana eksperimen yang diisi untuk checkout warung; setiap baris diputuskan sebelum pengunjung pertama ditempatkan. 353
Tabel 15.1 Perangkat yang membentuk desain antarmuka spasial dan wearable pada 2026, beserta masukan dan keluaran yang harus direncanakan perancang; harga adalah harga peluncuran di Amerika Serikat. 372
Tabel 15.2 Daftar periksa konteks pemakaian untuk antarmuka masa depan, dijawab untuk ruang tunggu puskesmas. 375
Tabel 16.1 Metode Bab 2–15 dan keputusan yang disumbangkan masing-masing bagi kios antrean. 390
Tabel 16.2 Kriteria sukses WCAG 2.2 yang diperiksa untuk kios antrean dan cara masing-masing dipenuhi 392
Tabel 16.3 Tiga tugas uji rintis dan kriteria keberhasilannya. 402
Tabel 16.4 Temuan uji rintis dengan penilaian keparahan dan perubahan yang direncanakan untuk iterasi kedua. 409

Daftar Kode

Kode 1.1 `Hello.java`: program Java pertama, yang juga melaporkan versi Java dan sistem operasinya. 12
Kode 1.2 `hello.py`: program Python pertama. 13
Kode 1.3 `check_setup.py`: memeriksa versi Python dan mencari Java. 13
Kode 1.4 `CheckSetup.java`: memeriksa versi Java dan mencari Python. 14
Kode 1.5 `data/orders.txt`: kedua desain yang ditulis sebagai operator Keystroke-Level Model. 17
Kode 1.6 `klm.py`: membaca desain sebagai daftar operator dan memprediksi waktunya. 18
Kode 1.7 `Klm.java`: kalkulator Keystroke-Level Model yang sama dalam Java. 19
Kode 1.8 `test_klm.py`: uji kalkulator Keystroke-Level Model. 22
Kode 1.9 `KlmTest.java`: pemeriksaan yang sama dalam Java. 22
Kode 2.1 `data/kiosk_old.csv`: tata letak lama, dua belas tombol dalam satu layar. 38
Kode 2.2 `data/kiosk_new.csv`: tata letak baru, enam tombol besar dan *Other* di layar 1, enam lagi di layar 2. 39
Kode 2.3 `humanfactors.py`: mencocokkan hukum Fitts dan Hick pada log dan menghitung waktu harapan setiap tata letak kios. 40
Kode 2.4 `HumanFactors.java`: pencocokan dan perbandingan tata letak yang sama dalam Java. 43
Kode 2.5 `test_humanfactors.py`: uji kedua hukum dan pencocokan kuadrat terkecil 47
Kode 2.6 `HumanFactorsTest.java`: pemeriksaan yang sama dalam Java. 48
Kode 3.1 `data/problems.csv`: empat belas masalah berbeda pada halaman pesanan pertama. 63
Kode 3.2 `data/findings.csv`: 26 temuan dari empat evaluator, masing-masing dengan keparahan 0 hingga 4. 63
Kode 3.3 `heuristics.py`: menggabungkan temuan, mengurutkan masalah, dan memperkirakan banyaknya evaluator yang diperlukan. 64
Kode 3.4 `Heuristics.java`: laporan yang sama dalam Java 65
Kode 3.5 `test_heuristics.py`: uji model dan pengurutan. 70
Kode 3.6 `HeuristicsTest.java`: pemeriksaan yang sama dalam Java, ditambah pemecahan CSV. 70
Kode 4.1 `data/themes.csv`: tema-tema diagram afinitas beserta frasa “bagaimana kita bisa” 84
Kode 4.2 `data/notes.csv`: 37 catatan dari enam penumpang dan empat pengemudi, masing-masing dengan tahap, tema, dan skor keluhan. 84
Kode 4.3 `data/ideas.csv`: sepuluh gagasan dengan dampak, upaya, dan suara. 85

Kode 4.4 `synthesis.py`: tema beserta jangkauannya, keluhan per tahap perjalanan, pertanyaan "bagaimana kita bisa", dan gagasan yang diprioritaskan. 85
Kode 4.5 `Synthesis.java`: sintesis yang sama dalam Java. 87
Kode 4.6 `test_synthesis.py`: uji kuadran, peringkat tema, dan rata-rata perjalanan. 93
Kode 4.7 `SynthesisTest.java`: pemeriksaan yang sama dalam Java. 93
Kode 5.1 `survey_stats.py`: merangkum survei Likert dengan rata-rata, selang kepercayaan 95%, dan proporsi *top-2-box*. 109
Kode 5.2 `SurveyStats.java`: ringkasan survei yang sama dalam Java. 111
Kode 5.3 `saturation.py`: frekuensi kode, kode baru per wawancara, dan aturan saturasi Guest, Namey, dan Chen. 115
Kode 5.4 `Saturation.java`: analisis yang sama dalam Java. 116
Kode 5.5 `test_research.py`: pengujian statistik survei dan aturan saturasi. 120
Kode 5.6 `ResearchTest.java`: pemeriksaan yang sama dalam Java. 121
Kode 6.1 `flow_v1.txt`: alur kios pada uji kertas, satu baris per tombol. 134
Kode 6.2 `flow_check.py`: menemukan layar tak terjangkau, jalan buntu, dan ketukan paling sedikit menuju setiap tujuan. 134
Kode 6.3 `FlowCheck.java`: pemeriksa yang sama dalam Java. 136
Kode 6.4 `flow_v2.txt`: alur setelah uji kertas. 139
Kode 6.5 `kiosk_pages.py`: teks, layanan, gaya, dan enam layar prototipe kios. 141
Kode 6.6 `kiosk_proto.py`: prototipe kios yang dilayani Flask pada porta 4060. 143
Kode 6.7 `KioskPages.java`: layar-layar yang sama dalam Java. 144
Kode 6.8 `KioskProto.java`: prototipe kios yang dilayani server HTTP bawaan JDK pada porta 4061. 147
Kode 6.9 `shoot.py`: menelusuri prototipe yang sedang berjalan dengan Playwright dan menyimpan tangkapan layar. 149
Kode 6.10 `test_proto.py`: pengujian pemeriksa alur dan prototipe. 151
Kode 6.11 `ProtoTest.java`: pemeriksaan yang sama dalam Java. 152
Kode 7.1 `usability_metrics.py`: keberhasilan tugas dengan selang *adjusted-Wald*, waktu rata-rata geometrik, SUS beserta nilainya, UMUX-Lite, dan penemuan masalah. 167
Kode 7.2 `UsabilityMetrics.java`: metrik yang sama dalam Java. 170
Kode 7.3 `test_eval.py`: pengujian metrik usabilitas dan uji A/B. 176
Kode 7.4 `EvalTest.java`: pemeriksaan yang sama dalam Java. 176
Kode 7.5 `ab_test.py`: uji-z dua proporsi, selang kepercayaan selisih, dan ukuran sampel. 179
Kode 7.6 `AbTest.java`: analisis A/B yang sama dalam Java. 180
Kode 8.1 `contrast.py`: memeriksa pasangan warna token terhadap batas minimum kontras WCAG 2.2 dan menyarankan warna yang memenuhi syarat. 193
Kode 8.2 `Contrast.java`: pemeriksa yang sama dalam Java. 195
Kode 8.3 `cardsort.py`: kemiripan kemunculan bersama dan pengelompokan *average linkage* untuk *card sorting* terbuka. 202
Kode 8.4 `CardSort.java`: analisis yang sama dalam Java. 205
Kode 8.5 `test_design.py`: pengujian pemeriksa kontras dan analisis *card sorting*. 209
Kode 8.6 `DesignTest.java`: pemeriksaan yang sama dalam Java. 210
Kode 9.1 `menu.csv`: lima menu, harganya, dan kata-kata yang dipakai pelanggan untuk menyebutnya. 225
Kode 9.2 `order_core.py`: menu, keranjang, dan dua pengurai untuk gaya perintah dan gaya suara. 225
Kode 9.3 `OrderCore.java`: menu, keranjang, dan pengurai yang sama dalam Java. 228
Kode 9.4 `cli.py`: gaya perintah dan gaya suara yang diketik, dengan konfirmasi eksplisit di bawah ambang keyakinan. 232
Kode 9.5 `Cli.java`: antarmuka perintah dan suara yang sama dalam Java. 233
Kode 9.6 `session.txt`: sesi tertulis berisi perintah, ucapan, dan tiga masalah yang disengaja. 235
Kode 9.7 `kiosk_v1.html`: versi pertama templat halaman kios, dengan tempat-tempat isian yang diisi oleh peladen. 235

Kode 9.8 `kiosk.py`: kios manipulasi langsung yang dilayani Flask pada porta 4090. 236
Kode 9.9 `Kiosk.java`: kios yang sama dengan peladen HTTP bawaan JDK pada porta 4091. 238
Kode 9.10 `audit_targets.py`: mengukur setiap elemen kios yang sedang berjalan yang dapat diketuk pada lebar ponsel dan membandingkannya dengan tiga pedoman. 241
Kode 9.11 `test_order_core.py`: pengujian inti pesanan untuk pytest. 243
Kode 9.12 `OrderCoreTest.java`: pemeriksaan yang sama dalam Java biasa. 243
Kode 10.1 `naive.html`: versi pertama halaman pemesanan yang hanya untuk desktop. 257
Kode 10.2 `responsive.html`: tulisan ulang *mobile-first* dengan baris *viewport*, tata letak cair, ruas berlabel, dan bilah tindakan dalam jangkauan ibu jari. 258
Kode 10.3 `strings.csv`: kata-kata halaman dalam bahasa Inggris dan Indonesia. 259
Kode 10.4 `pages.py`: mengisi templat mana pun dalam bahasa mana pun. 260
Kode 10.5 `app.py`: melayani kedua halaman dengan Flask pada porta 4100. 260
Kode 10.6 `Pages.java`: pengisian templat yang sama dalam Java. 261
Kode 10.7 `Server.java`: melayani kedua halaman dengan peladen HTTP JDK pada porta 4101. 262
Kode 10.8 `check_responsive.py`: mengukur lebar tata letak, luapan, ukuran teks dan target efektif, serta ruas telepon pada empat lebar. 263
Kode 10.9 `resources_v1.csv`: sumber daya desain pertama yang sudah jadi (desain ramping ada di `resources_v2.csv`). 265
Kode 10.10 `networks.csv`: tiga profil jaringan yang dipakai untuk perkiraan. 265
Kode 10.11 `budget.py`: memperkirakan waktu tiba halaman pada setiap jaringan dan membandingkannya dengan anggaran LCP. 265
Kode 10.12 `Budget.java`: perkiraan yang sama dalam Java. 265
Kode 10.13 `test_ch10.py`: pengujian perkiraan dan perenderan halaman. 268
Kode 10.14 `Ch10Test.java`: pemeriksaan yang sama dalam Java biasa. 269
Kode 11.1 `queue_v1.html`: halaman kios pertama, dengan kegagalan aksesibilitas yang paling umum. 280
Kode 11.2 `colors.csv`: pasangan-pasangan warna kios dan jenis konten yang diwarnai masing-masing. 281
Kode 11.3 `contrast.py`: rasio kontras WCAG dan warna lebih gelap yang lulus. 281
Kode 11.4 `Contrast.java`: pemeriksa kontras yang sama dalam Java. 282
Kode 11.5 `a11y_audit.py`: menemukan bahasa, teks alternatif, label, dan nama yang hilang, elemen yang dapat diklik tetapi tak terjangkau papan ketik, lompatan judul, dan id ganda. 283
Kode 11.6 `A11yAudit.java`: audit yang sama dalam Java dengan `java.util.regex`. 285
Kode 11.7 `queue_v2.html`: halaman kios yang diperbaiki, memakai elemen HTML bawaan, label, teks alternatif, wilayah *live*, dan cincin fokus yang terlihat. 288
Kode 11.8 `test_ch11.py`: pengujian pemeriksa kontras dan audit. 291
Kode 11.9 `Ch11Test.java`: pemeriksaan yang sama dalam Java biasa. 292
Kode 12.1 `flow_v1.txt`: alur pendaftaran dan keanggotaan pertama Laundry Wangi, diuraikan baris demi baris. 304
Kode 12.2 `consent_lint.py`: memeriksa uraian alur dari persetujuan yang sudah tercentang dan dibundel, campur tangan antarmuka, *confirmshaming*, desakan waktu palsu, dan jalan keluar yang sulit. 305
Kode 12.3 `ConsentLint.java`: *linter* yang sama dalam Java. 306
Kode 12.4 `flow_v2.txt`: alur yang diperbaiki, dengan tujuan mati secara bawaan, tombol setara, tanpa hitung mundur, dan jalan keluar semudah jalan masuk. 308
Kode 12.5 `consent_log.csv`: peristiwa persetujuan beserta waktu, pengguna, tujuan, tindakan, dan versi teks persetujuan yang ditampilkan. 308
Kode 12.6 `consent_ledger.py`: memutar ulang peristiwa persetujuan menjadi keadaan kini dan menjawab apakah suatu tujuan diizinkan. 308
Kode 12.7 `ConsentLedger.java`: buku besar yang sama dalam Java 309
Kode 12.8 `test_ch12.py`: pengujian *linter* dan buku besar. 311
Kode 12.9 `Ch12Test.java`: pemeriksaan yang sama dalam Java biasa. 312

Kode 13.1 `laundry_assistant.py`: asisten pesanan, dengan kata keyakinan, pertanyaan untuk slot yang kosong, konfirmasi, pembatalan, dan jalan ke manusia. 326
Kode 13.2 `LaundryAssistant.java`: asisten yang sama dalam Java. 329
Kode 13.3 `make_log.py`: menulis log simulasi dengan seed berisi 200 kasus yang ditinjau untuk setiap versi layar. 335
Kode 13.4 `reliance.py`: ketergantungan berlebih, ketergantungan kurang, keputusan akhir yang benar, dan galat kalibrasi harapan. 335
Kode 13.5 `Reliance.java`: analisis yang sama dalam Java. 336
Kode 13.6 `test_assistant.py`: uji penguraian, harga, dan jendela pembatalan. 339
Kode 13.7 `AssistantTest.java`: pemeriksaan yang sama dalam Java. 340
Kode 14.1 `make_events.py`: menulis dua log peristiwa simulasi dengan seed, satu sehat dan satu dengan bug pencatatan di varian B. 353
Kode 14.2 `funnel.py`: pengguna per langkah checkout dan per varian, dengan konversi langkah dan konversi keseluruhan. 354
Kode 14.3 `Funnel.java`: corong yang sama dalam Java. 355
Kode 14.4 `abtest.py`: pemeriksaan rasio sampel, selisih dengan selang kepercayaan 95%, uji-*z*, dan ukuran sampel. 357
Kode 14.5 `AbTest.java`: pemeriksaan yang sama dalam Java. 359
Kode 14.6 `test_abtest.py`: uji fungsi galat, pemeriksaan rasio, uji-*z*, dan ukuran sampel. 362
Kode 14.7 `AbTestTest.java`: pemeriksaan yang sama dalam Java. 362
Kode 15.1 `visual_angle.py`: ukuran fisik untuk suatu sudut pandang pada empat jarak, dan sudut tombol 1 cm. 375
Kode 15.2 `VisualAngle.java`: kalkulator yang sama dalam Java. 376
Kode 15.3 `make_gaze.py`: menulis log simulasi dengan seed berisi 1.000 lirikan, dengan tanda mana yang merupakan pilihan disengaja. 378
Kode 15.4 `dwell.py`: aktivasi keliru dan pilihan terlewat untuk setiap waktu tatap, serta waktu tatap dengan biaya terendah. 379
Kode 15.5 `Dwell.java`: analisis yang sama dalam Java. 379
Kode 15.6 `test_spatial.py`: uji geometri dan arah tawar-menawar waktu tatap. 382
Kode 15.7 `SpatialTest.java`: pemeriksaan yang sama dalam Java. 382
Kode 16.1 `kiosk_app.py`: kios antrean dalam Python dengan Flask, dalam dua bahasa dan dengan mode huruf besar. 394
Kode 16.2 `KioskServer.java`: kios yang sama dengan server HTTP bawaan JDK. 396
Kode 16.3 `shoot.py`: tangkapan layar Playwright atas kios kita sendiri yang sedang berjalan, dalam kedua bahasa. 401
Kode 16.4 `evaluate.py`: SUS, UMUX-Lite, penyelesaian tugas dengan selang Wald yang disesuaikan, dan kontras warna WCAG. 404
Kode 16.5 `Evaluate.java`: evaluasi yang sama dalam Java. 405
Kode 16.6 `test_evaluate.py`: uji penskoran SUS, kontras, dan selang Wald yang disesuaikan. 410
Kode 16.7 `EvaluateTest.java`: pemeriksaan yang sama dalam Java. 410

Daftar Persamaan

(1.1) 16
(2.1) 35
(2.2) 35
(2.3) 36
(2.4) 37
(2.5) 37
(2.6) 39

(2.7) 40
(3.1) 61
(3.2) 61
(3.3) 61
(4.1) 81
(4.2) 82
(4.3) 82
(5.1) 108
(5.2) 108
(5.3) 109
(5.4) 114
(7.1) 164
(7.2) 165
(7.3) 166
(7.4) 166
(7.5) 174
(7.6) 178
(7.7) 178
(8.1) 191
(8.2) 192
(8.3) 192
(8.4) 192
(8.5) 201
(8.6) 202
(9.1) 220
(9.2) 221
(9.3) 223
(10.1)....254
(10.2)....254
(10.3)....254
(10.4)....256
(11.1)....277
(11.2)....277
(11.3)....277
(12.1)....301
(12.2)....301
(13.1)....321
(13.2)....323
(13.3)....323
(13.4)....324
(13.5)....324
(14.1)....348
(14.2)....348
(14.3)....350
(14.4)....350
(14.5)....351
(14.6)....351
(15.1)....370
(15.2)....370
(15.3)....371
(15.4)....379

(16.1)....391
(16.2)....391
(16.3)....403
(16.4)....403
(16.5)....403
(16.6)....403

Kata Pengantar

Warung Bu Ratna menerima pesanan yang sama — dua es teh, satu nasi goreng, satu ayam geprek — pada dua layar yang berbeda, dan pada salah satunya pelanggan menyerah di tengah jalan. Seorang pasien lanjut usia berdiri di depan kios antrean puskesmas dan tidak menemukan tombol yang ia perlukan. Pelanggan sebuah binatu berhenti bertanya lewat WhatsApp lalu berhenti datang. Sebuah asisten *chat* menyebutkan harga tengah malam, dan tidak ada yang tahu apakah harga itu benar. Tidak satu pun masalah itu adalah *bug* dalam kode; semuanya adalah masalah dalam pertemuan antara manusia dan mesin. Interaksi manusia dan komputer adalah disiplin yang mempelajari pertemuan itu, dan ia memiliki metode, ukuran, dan bukti untuk membuat pertemuan itu berjalan baik.

Buku ini tumbuh dari mata kuliah Interaksi Manusia dan Komputer di ITS, tempat materi ini telah diajarkan kepada mahasiswa informatika selama bertahun-tahun, dan telah ditulis ulang dari dasar untuk tahun 2026. Fondasinya adalah yang bertahan lama — persepsi dan memori manusia, hukum Fitts dan Hick, prinsip Norman dan heuristik Nielsen, desain yang berpusat pada pengguna, riset pengguna, prototipe, dan pengujian usabilitas — karena semua itu tidak berubah ketika perangkat lunaknya berubah. Segala yang dibangun di atasnya telah dimutakhirkan: WCAG 2.2 dan undang-undang aksesibilitas yang kini berlaku, dari *European Accessibility Act* sampai aturan Indonesia sendiri tentang disabilitas dan data pribadi; bahasa desain yang diperbarui pada 2025, *Liquid Glass* dari Apple, *Material 3 Expressive*, dan *Fluent 2*; *passkey*, regulasi *dark pattern*, dan persetujuan yang jujur; serta antarmuka baru AI generatif — *chat*, kopilot, dan agen — bersama perangkat spasial, *wearable*, dan *ambient* yang mulai meninggalkan persegi panjang layar.

Untuk siapa buku ini ditulis

Buku ini lengkap dengan sendirinya, dan Anda dianggap seorang pemula. Jika Anda belum pernah menjalankan uji usabilitas, belum pernah menulis sebaris kode, dan belum pernah mendengar istilah heuristik, Andalah pembaca yang dibayangkan saat buku ini ditulis. Setiap istilah diperkenalkan dengan kata-kata sederhana sebelum dipakai, setiap metode lebih dulu ditunjukkan pada situasi kecil yang nyata, teorinya diberikan bersama bukti di baliknya dan sumber tempat ia dapat diperiksa, dan setiap program dalam buku ini sudah dijalankan sebelum keluarannya dicetak di sini. Bab 1 dibuka dengan satu pesanan pada dua desain dan ditutup dengan meja kerja yang siap untuk semua bab sesudahnya.

Pembaca yang sudah membangun antarmuka, dan mahasiswa yang memerlukan ketelitian sebuah mata kuliah universitas, akan menemukan separuh lain buku ini: riset di balik setiap prinsip, kuesioner dan metrik baku beserta rumus dan selang kepercayaannya, eksperimen yang dianalisis dengan benar, templat dan daftar periksa yang siap dipakai dalam proyek nyata, serta program yang utuh dan teruji, siap disesuaikan untuk pekerjaan mereka sendiri. Prinsip itu membuat buku ini tebal. Buku yang lebih tipis akan menganggap Anda sudah menguasai bagian-bagian yang sulit, dan buku semacam itu tidak berguna bagi orang yang belum menguasainya.

Satu gagasan, dua bahasa

Interaksi manusia dan komputer bukan sekadar soal pendapat dan sketsa; sebagian besar darinya dapat diukur, dan mengukur adalah tugas program. Karena itu, di mana pun sebuah program membantu, buku ini menuliskannya dua kali, dalam **Java** dan dalam **Python**, dan kedua versi itu membaca data yang sama dan mencetak keluaran yang identik. Java adalah bahasa sistem besar yang berumur panjang, bahasa ponsel Android, dan bahasa banyak layanan bank serta pemerintahan; Python adalah bahasa percobaan cepat dan pengolahan data. Program-programnya sengaja dibuat kecil: kalkulator hukum Fitts dan Hick, pembuat laporan evaluasi heuristik, perangkum survei, penghitung skor SUS dan UMUX-Lite, pemeriksa kontras, *linter* alur persetujuan, analisis uji A/B, serta halaman web kecil yang aksesibel, dilayani oleh `com.sun.net.httpserver` bawaan Java dan oleh **Flask**, lalu diperiksa di peramban sungguhan dengan **Playwright**. Jika Anda belum mengenal keduanya, mulailah dengan Python, lalu baca versi Java-nya sesudah itu.

Semuanya ditulis dan dijalankan di **Visual Studio Code**, editor gratis yang bekerja dengan cara yang sama di Windows, macOS, dan Linux; Bab 1 memasangnya, bersama JDK 25 dan Python 3.14, langkah demi langkah pada ketiga sistem itu. Pekerjaan desainnya sendiri tidak memerlukan apa-apa selain kertas, pena, dan orang-orang yang akan memakai hasil rancangan Anda.

Isi buku ini

Bab 1 menjelaskan mengapa antarmuka penting, dengan mengikuti warung Bu Ratna yang mulai berjualan daring: apa itu IMK, bagaimana ia tumbuh dari kartu berlubang sampai agen AI, mengapa ia penting di Indonesia pada 2026, dan bab ini menyiapkan meja kerja.

Bab 2 dan 3 meletakkan fondasi. Bab 2 mengamati Bu Siti di kios puskesmas untuk menjelaskan faktor manusia yang tidak dapat diubah oleh pembaruan perangkat lunak mana pun — persepsi, perhatian, dan memori — dan mengubah hukum Fitts dan Hick menjadi kalkulator. Bab 3 mengikuti binatu Mbak Dewi yang meninggalkan WhatsApp, menyajikan prinsip Norman dan keluarga heuristik desain, menemui bahasa desain 2025–2026, dan membangun pembuat laporan evaluasi heuristik.

Bab 4 sampai 7 mengajarkan proses merancang hal yang tepat. Bab 4 membantu sebuah pangkalan ojek yang menginginkan aplikasi dengan proses desain yang berpusat pada manusia menurut ISO 9241-210, *Double Diamond*, dan *design thinking*, serta melihat peran AI dalam proses desain 2026. Bab 5 mencari tahu mengapa binatu Bu Rina kehilangan pelanggan dengan wawancara, *contextual inquiry*, survei, dan pengodean data kualitatif. Bab 6 membuat prototipe kios antrean puskesmas, dari kertas ke *wireframe*, *mockup* yang dapat diklik, dan prototipe berkode. Bab 7 mengevaluasi prapesan di Warung Bu Tini dengan evaluasi heuristik, uji usabilitas, ukuran kinerja, serta kuesioner SUS dan UMUX-Lite.

Bab 8 sampai 10 membahas antarmuka itu sendiri. Bab 8 merancang ulang aplikasi sebuah bank perdesaan dengan Gestalt, hierarki visual, tipografi, warna dan kontras, *design system*, serta arsitektur informasi. Bab 9 membandingkan cara-cara memesan di meja Bu Tini — GUI, sentuh, suara, dan gestur — dengan satu inti pesanan dan tiga antarmuka depan. Bab 10 merancang untuk ponsel murah Mas Yoga, mengukur halaman pemesanan *mobile-first* di peramban sungguhan pada empat lebar layar dan tiga jaringan.

Bab 11 dan 12 membahas orang-orang yang terlalu sering ditinggalkan atau disesatkan. Bab 11 membuat situs web puskesmas aksesibel dengan WCAG 2.2 dan undang-undang yang kini

mewajibkannya, lalu mengujinya dalam empat lapis. Bab 12 menelaah alur keanggotaan binatu Bu Sri untuk menunjukkan desain emosional, kepercayaan, *dark pattern*, dan persetujuan yang jujur, serta memeriksa alur persetujuan secara otomatis.

Bab 13 sampai 15 melihat antarmuka yang kini sedang tiba. Bab 13 merancang asisten pesanan tengah malam Bu Wati — *chat*, kopilot, dan agen — di sekitar konfirmasi, pembatalan, dan kepercayaan yang terkalibrasi. Bab 14 menyelesaikan perdebatan tentang halaman *checkout* Bu Tini yang sepi dengan analitik, *funnel*, eksperimen A/B, dan metrik. Bab 15 mengikuti Mbah Sumi dan sebuah layar antrean menuju komputasi spasial, *wearable*, dan *ambient*, dengan ukuran target dan waktu *dwell* yang dihitung, bukan ditebak.

Bab 16, proyek akhir, merancang dan mengevaluasi sebuah sistem interaktif utuh: kios antrean swalayan untuk Puskesmas Sehat Selalu, dari riset di meja pendaftaran sampai kios dwibahasa yang berjalan dan bukti bahwa kios itu membantu para pasiennya.

Setiap bab menempuh jalan yang sama: sebuah situasi dari kehidupan sehari-hari di Indonesia, gagasannya dalam kata-kata sederhana, teorinya beserta bukti, metode praktis dengan templat dan daftar periksa, program pendamping kecil dalam Java dan Python bila membantu, menjalankan dan mengujinya, IMK dalam praktik di Profitina, kesalahan yang sering terjadi, ringkasan, soal latihan dengan penyelesaian lengkap, dan daftar pustaka.

Setiap program dalam buku ini sudah dijalankan sebelum keluarannya dicetak. Program-program itu dikumpulkan, bab demi bab, dalam folder `Code` yang menyertai buku ini, dari `Chapter01` sampai `Chapter16`, masing-masing dengan folder `java` dan `python`, data yang diperlukannya, dan README yang menjelaskan cara menjalankannya di terminal dan di VS Code, pada Windows, macOS, dan Linux.

Cara membacanya

Dengan pena, beberapa lembar kertas, dan VS Code terbuka. Bab 1 didahulukan bagi semua pembaca, karena bab itulah yang menjelaskan mengapa antarmuka penting dan membangun meja kerja; setelah itu bab-bab bergerak dari manusia, melalui proses, menuju antarmuka dan jenis-jenis antarmuka yang baru, dan setiap bab menyebutkan bab-bab sebelumnya yang menjadi pijakannya. Kerjakan metodenya, jangan hanya membacanya: gambarlah prototipe kertas, wawancarai dua orang, jalankan uji usabilitas lima orang pada sesuatu yang Anda pakai setiap hari, lalu hitung skornya. Setelah sebuah program berjalan, ubahlah — satu ukuran target, satu warna, satu jawaban kuesioner — lalu perhatikan bagaimana hasilnya berubah. Semua soal latihan di akhir tiap bab dilengkapi penyelesaian lengkap — gunakan sesudah usaha Anda sendiri, bukan sebagai penggantinya.

Gambar, tabel, kode, pseudokode, dan persamaan yang bernomor semuanya adalah tautan hidup: pada edisi elektronik, klik “Kode 5.3” 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 ITS sejak 1998. Ia membimbing dan menguji tugas akhir mahasiswa sarjana, mengajar MLOps dalam program IBM SkillsBuild, dan membangun produk: ia mendirikan Profitina, platform visibilitas AI, yang keputusan desainnya sehari-hari menutup setiap bab dalam buku ini. 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 setiap bab dalam buku ini dibuka dengan seseorang dan masalahnya, bukan dengan sebuah definisi.

Ucapan terima kasih

Kepada para mahasiswa yang mempelajari versi-versi awal materi ini dalam mata kuliah Interaksi Manusia dan Komputer dan mengajukan pertanyaan yang menyingkapkan di mana sebuah penjelasan masih tipis, serta kepada rekan-rekan di ITS yang bersama-sama mengajarkan dan menyempurnakan materi ini dari tahun ke tahun. Kepada para pemilik warung, pemilik binatu, pengemudi, tenaga kesehatan, dan pelanggan lanjut usia yang perjumpaan sehari-harinya dengan layar memberi setiap bab kisahnya. Standar buku ini juga berutang pada standar dan pedoman yang diterbitkan W3C, ISO, dan para pembuat platform, pada dokumentasi resmi Java, Python, Flask, Playwright, dan VS Code, serta pada buku teks klasik dan literatur riset interaksi manusia dan komputer, yang dikutip di setiap tempat penggunaannya.

Surabaya, September 2026



Irfan Subakti · yifana@gmail.com

Profitina

profitina.com

Bab 1

Pengantar IMK: Mengapa Antarmuka Penting

Setiap hari jutaan orang di Indonesia memesan makanan, membayar parkir, mengantre di klinik, dan mengirim uang melalui layar yang tidak mereka rancang dan tidak dapat mereka ubah. Ketika layarnya jelas, mereka hampir tidak menyadarinya; ketika tidak, mereka kehilangan waktu, uang, dan kadang-kadang harga diri. Bab ini menjelaskan apa itu interaksi manusia dan komputer (IMK) dan mengapa ia penting, menyiapkan meja kerja yang dipakai di sepanjang buku, lalu ditutup dengan sebuah program kecil pertama yang mengubah perdebatan desain menjadi angka. Di akhir bab ini Anda akan mampu mendefinisikan *usability* (kebergunaan), *user experience* (pengalaman pengguna), dan aksesibilitas menurut standar internasional; menjelaskan dua jurang yang memisahkan tujuan seseorang dari tanggapan sistem; menjelaskan mengapa mutu antarmuka menjadi urusan publik di Indonesia pada 2026, lengkap dengan angka dan aturan hukum terkini; memasang dan memeriksa JDK 25, Python 3.14, dan VS Code di Windows, macOS, atau Linux; serta memprediksi, dengan *Keystroke-Level Model*, berapa detik dan berapa jam kerja yang dihemat oleh layar yang lebih baik.

1.1 Warung Bu Ratna Masuk Daring

Bu Ratna mengelola warung kecil di dekat sebuah kampus di Surabaya: nasi ayam goreng, ayam geprek, nasi goreng, es teh, dan kopi. Sejak 2024 sebagian besar pesanannya datang lewat WhatsApp, dan keponakannya, Dimas, seorang mahasiswa, membuatkan halaman web sederhana di laptop lama di meja kasir agar ia dapat mengetik setiap pesanan, melihat antrean, dan mencetak slip untuk dapur. Dimas bangga dengan buatannya. Halaman itu punya daftar *dropdown* berisi keempat puluh dua menu, sebuah kotak jumlah, tombol *Add*, dan di bagian akhir tombol *Save*.

Pada hari Sabtu pertama, Bu Ratna berhenti memakainya pukul sebelas siang. Bukan karena halamannya rusak; setiap pesanan yang masuk tersimpan dengan benar. Masalahnya, setiap pesanan memakan waktu lama. Ia harus mengeklik *dropdown*, menggulir untuk mencari *Es Teh*, mengekliknya, mengeklik kotak jumlah, memindahkan tangan ke keyboard untuk mengetik angka 2, memindahkan tangan kembali ke mouse, mengeklik *Add*, lalu mengulang semuanya untuk menu berikutnya. Ketika dua puluh mahasiswa datang bersamaan selepas salat zuhur, ia kembali ke buku catatan dan pulpennya.

Reaksi pertama Dimas adalah reaksi hampir semua pemrogram: *nanti juga terbiasa.* Reaksi keduanya, seminggu kemudian, lebih menarik. Ia mengamati bibinya selama satu jam, menghitung langkah-langkahnya, lalu menggambar ulang halaman itu sebagai kisi ubin besar, satu ubin per menu, dengan menu yang paling sering dipesan di bagian atas. Satu klik pada ubin menambah satu porsi; klik kedua menambah satu lagi. Pesanan yang sama, yang semula memakan lebih dari dua puluh detik, kini hanya sekitar sepuluh detik. Bu Ratna memakai halaman baru itu tanpa perlu disuruh.

Tidak ada yang berubah pada basis data, jaringan, atau bahasa pemrogramannya di antara kedua versi. Yang berubah adalah *antarmuka*, bagian sistem yang dilihat, didengar, dan disentuh orang, dan melalui bagian itulah orang memberi tahu sistem apa yang harus dilakukan. Cerita ini berulang di mana-mana: di kios antrean puskesmas, tempat pasien lansia tidak menemukan tombol poli gigi di antara dua belas label kecil; di ATM bank yang menelan kartu siapa pun yang pergi setelah mengambil uang; di aplikasi ojek yang titik jemputnya jatuh di seberang sungai; di pemesanan laundry lewat WhatsApp, tempat tidak ada pelanggan yang ingat pilihan bernomor mana yang berarti "setrika saja". Subbab 1.11 bab ini mengukur kedua halaman Bu Ratna dengan sebuah model yang memprediksi lamanya langkah-langkah itu. Bab-bab berikutnya mengajarkan cara melihat masalah semacam ini sebelum pengguna mengalaminya, cara merancang yang lebih baik, dan cara membuktikan bahwa hasilnya memang lebih baik.

1.2 Apa Itu Interaksi Manusia dan Komputer

1.2.1 Sebuah definisi, dan tiga kata

Definisi klasiknya ditulis oleh komite kurikulum ACM SIGCHI, kelompok minat khusus asosiasi itu di bidang interaksi manusia dan komputer: *interaksi manusia dan komputer adalah disiplin yang berkaitan dengan perancangan, evaluasi, dan implementasi sistem komputasi interaktif untuk digunakan manusia, serta dengan kajian fenomena utama di sekitarnya* [1]. Empat kata dalam kalimat itu memikul seluruh isi buku ini. *Perancangan* berarti memutuskan seperti apa sistem itu sebelum dibangun. *Evaluasi* berarti mencari tahu, dengan orang sungguhan dan tugas sungguhan, apakah sistem itu bekerja bagi mereka. *Implementasi* berarti membangunnya, dan itulah sebabnya buku ini berisi program. Adapun *digunakan manusia* berarti ukuran keberhasilannya bukan apakah kodenya berjalan, melainkan apakah seseorang dapat mencapai tujuannya dengan sistem itu.

IMK menimba dari ilmu komputer untuk sisi mesin, dari psikologi dan ilmu kognitif untuk sisi manusia, serta dari desain, sosiologi, ergonomi, dan linguistik untuk segala sesuatu di antaranya. Bidang ini dipraktikkan dengan banyak nama: *usability engineering*, desain *user experience* (UX), desain interaksi, faktor manusia, rekayasa aksesibilitas. Dalam bahasa Inggris bidang ini disebut *Human-Computer Interaction* (HCI).

Tiga kata dipakai terus-menerus, dan standar internasional memberinya makna yang tepat.

> **DEFINISI.** *Usability* (kebergunaan) adalah sejauh mana suatu sistem, produk, atau layanan dapat digunakan oleh pengguna tertentu untuk mencapai tujuan tertentu dengan *efektivitas* (ketepatan dan kelengkapan), *efisiensi* (sumber daya yang dipakai, seperti waktu dan tenaga), dan *kepuasan* (tanggapan fisik, kognitif, dan emosional pengguna) dalam konteks penggunaan tertentu (ISO 9241-11:2018) [2]. *User experience* (pengalaman pengguna) adalah persepsi dan tanggapan seseorang yang timbul dari penggunaan atau antisipasi penggunaan suatu sistem, sebelum, selama, dan sesudah penggunaan (ISO 9241-210:2019) [3]. *Aksesibilitas* adalah sejauh mana sistem dapat digunakan oleh orang dengan rentang kebutuhan, karakteristik, dan kemampuan yang seluas-luasnya, termasuk penyandang disabilitas [2].

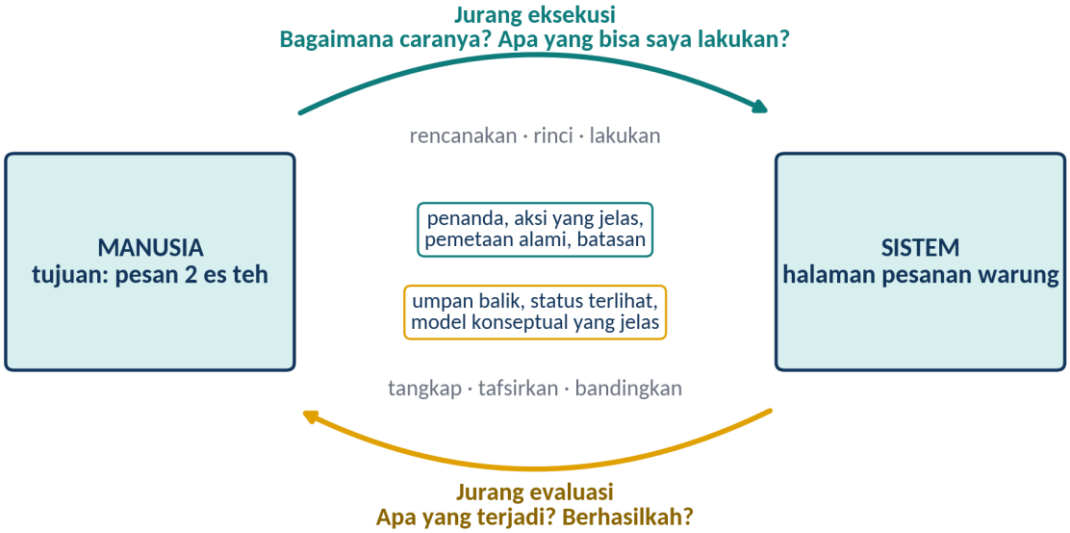
Perhatikan kata *tertentu* yang muncul tiga kali dalam definisi *usability*. Tidak ada yang namanya "mudah dipakai" secara umum. Halaman pertama Dimas sangat mudah dipakai oleh Dimas di mejanya. Halaman itu gagal bagi pengguna tertentu (Bu Ratna, lima puluh empat tahun, juru masak yang cepat tetapi lambat memakai mouse), tujuan tertentu (memasukkan pesanan tiga menu), dan konteks penggunaan tertentu (dua puluh mahasiswa lapar di depan meja kasir). Ubah

salah satu dari ketiganya dan penilaiannya bisa berubah. Karena itulah setiap metode di buku ini dimulai dengan bertanya *siapa*, *untuk apa*, dan *di mana*.

Ketiga mutu *usability* juga dapat diukur, dan itulah yang membuat definisi ini berguna. Efektivitas diukur, misalnya, sebagai persentase pesanan yang dimasukkan tanpa kesalahan; efisiensi sebagai detik per pesanan; kepuasan dengan kuesioner singkat seperti *System Usability Scale* di Bab 7. Pengalaman pengguna lebih luas lagi: di dalamnya ada kecemasan Bu Ratna pada Jumat malam bahwa esok hari akan selambat Sabtu lalu, dan rasa senang kecilnya seminggu kemudian ketika antrean sudah kosong sebelum gelombang berikutnya datang.

1.2.2 Dua jurang

Mengapa satu antarmuka sulit dan yang lain mudah? Jawaban tunggal yang paling berguna datang dari ilmuwan kognitif Don Norman, yang bukunya *The Design of Everyday Things* masih merupakan bacaan pertama terbaik di bidang ini [4]. Norman menggambarkan setiap tindakan sebagai sebuah siklus kecil. Seseorang punya tujuan (memesan dua es teh); ia merencanakan tindakan, merinci langkah-langkahnya, dan melakukannya (sisi *eksekusi*); lalu ia menangkap apa yang ditampilkan sistem, menafsirkannya, dan membandingkannya dengan tujuannya (sisi *evaluasi*). Di antara manusia dan sistem terbentang dua jurang, seperti digambarkan pada **Gambar 1.1**.



Gambar 1.1 Dua jurang Norman: eksekusi (bagaimana caranya?) dan evaluasi (berhasilkah?), serta mutu desain yang menjembatani masing-masing.

Jurang eksekusi (*gulf of execution*) adalah jarak antara apa yang ingin dilakukan seseorang dan apa yang diizinkan sistem, sebagaimana ia memahaminya. Pada halaman pertama Dimas jurang ini lebar: tujuan "dua es teh" harus diterjemahkan ke lima kontrol yang terpisah. *Jurang evaluasi* (*gulf of evaluation*) adalah jarak antara apa yang ditampilkan sistem dan apa yang perlu diketahui seseorang untuk menilai apakah tujuannya tercapai. Ketika Bu Ratna mengeklik *Add* dan tidak ada perubahan yang terlihat selain satu baris kecil di bawah daftar panjang, ia mengeklik lagi, dan dapur membuat empat es teh.

Desain yang baik menjembatani kedua jurang. Penanda (*signifier*) yang menunjukkan apa yang bisa dilakukan, pemetaan (*mapping*) yang alami antara kontrol dan akibatnya, serta batasan (*constraint*) yang membuat tindakan keliru mustahil dilakukan akan mempersempit jurang eksekusi. Umpan balik yang segera, status yang terlihat, dan model konseptual yang jelas akan

mempersempit jurang evaluasi. Bab 3 membahas setiap prinsip ini secara rinci; untuk sekarang cukup kita perhatikan bahwa setiap masalah di Subbab 1.1 adalah jurang yang terlalu lebar di salah satu sisi. Kios puskesmas punya jurang eksekusi yang lebar (label mana dari dua belas itu yang berarti "dokter gigi"?); ATM yang menelan kartu yang terlupa dulu punya jurang evaluasi yang lebar (tujuan orang itu, yaitu uang tunai, sudah tercapai, sehingga kartunya terlupakan), dan bank menjembataninya dengan mengembalikan kartu *sebelum* uang keluar.

> **WAWASAN KUNCI:** Ketika seseorang berbuat salah saat memakai sebuah antarmuka, pertanyaan pertamanya bukan "apa yang salah dengan orang ini?", melainkan "jurang mana yang terlalu lebar?". Menyalahkan pengguna itu mudah dan tidak mengubah apa pun; mempersempit jurang mengubah perilaku setiap pengguna berikutnya.

1.2.3 Apa yang bukan IMK

Tiga salah paham cukup sering muncul sehingga layak disebut. IMK bukan *membuat tampilan cantik*: desain visual memang penting (Bab 8), tetapi halaman indah yang menyembunyikan aksi utamanya tetaplah gagal. IMK bukan *menanyakan kepada pengguna apa yang mereka mau*: orang tidak pandai meramalkan perilakunya sendiri, dan Bab 5 menunjukkan mengapa pengamatan mengalahkan pendapat. IMK juga bukan *akal sehat*: kalau memang akal sehat, halaman pertama Dimas sudah menjadi halaman terakhirnya. Apa yang tampak jelas bagi orang yang membangun sistem justru adalah hal yang tidak lagi terlihat olehnya, karena ia sudah tahu cara kerjanya. Disiplin ini ada untuk mengganti titik buta itu dengan bukti.

1.3 Dari Kartu Berlubang hingga Agen: Sejarah Singkat dan Masa Kini

Sejarah IMK adalah sejarah tentang siapa penggunanya. Pada 1950-an dan 1960-an pengguna adalah seorang spesialis yang menyiapkan kartu berlubang lalu menunggu berjam-jam untuk hasil cetaknya; antarmukanya adalah antrean. Sketchpad karya Ivan Sutherland (1963) memungkinkan orang menggambar langsung di layar dengan pena cahaya, dan demonstrasi Douglas Engelbart pada 1968 memperlihatkan mouse, jendela, dan penyuntingan bersama. Di Xerox PARC pada 1970-an, komputer Alto menggabungkan semuanya menjadi antarmuka pengguna grafis, yang kemudian dibawa Apple Macintosh (1984) dan Microsoft Windows ke kantor dan rumah. Web pada 1990-an menghadirkan antarmuka di depan orang yang belum pernah dilatih memakai komputer apa pun; iPhone (2007) dan Android menjadikan layar sentuh antarmuka yang paling umum di bumi. Sejarah bidang ini yang ditulis Jonathan Grudin menelusuri bagaimana IMK dalam kurun yang sama bergeser dari alat bagi para ahli menjadi mitra bagi semua orang [5].

Tabel 1.1 merangkum era-era itu beserta tuntutan masing-masing kepada pengguna.

Tabel 1.1 Era-era interaksi manusia dan komputer dan apa yang dituntut masing-masing dari penggunanya.

Era	Antarmuka yang khas	Siapa penggunanya	Apa yang harus dipelajari pengguna
1950-an–1970-an	Pekerjaan *batch*, kartu berlubang, baris perintah	Spesialis terlatih	Sintaksis perintah yang persis
1980-an–1990-an	Antarmuka grafis: jendela, ikon, menu, penunjuk	Pekerja kantor, lalu pengguna rumahan	Metafora seperti desktop, folder, tempat sampah
1990-an–2000-an	Web	Siapa saja yang punya peramban	Tautan, formulir, pencarian
2007–2020-an	Layar sentuh, aplikasi seluler	Hampir semua orang, termasuk pengguna pemula	Gestur, konvensi aplikasi
2020-an	Suara, AI percakapan dan generatif, agen, komputasi spasial	Semua orang, sering tanpa layar sama sekali	Apa yang harus ditanyakan, dan kapan jawabannya layak dipercaya

Di mana kita berada pada September 2026? Ada empat perkembangan yang membentuk apa yang dijumpai mahasiswa IMK hari ini.

Bahasa desain diperbarui. Pada 9 Juni 2025 Apple memperkenalkan Liquid Glass, material tembus pandang yang "memantulkan dan membiaskan sekelilingnya" dan menyesuaikan diri dengan konten di bawahnya, di iOS 26, iPadOS 26, macOS Tahoe 26, watchOS 26, dan tvOS 26 [6]. Material 3 Expressive dari Google, yang dirilis pada musim semi yang sama, didukung oleh 46 studi riset dengan lebih dari 18.000 partisipan; Google melaporkan bahwa orang menemukan elemen antarmuka kunci hingga empat kali lebih cepat pada tata letak ekspresif [7]. Fluent 2 dari Microsoft adalah *design system* bersama untuk Windows dan Microsoft 365 [8]. Semua itu bukan mode untuk ditiru; Bab 3 menunjukkan bagaimana masing-masing menerapkan (dan kadang meregangkan) prinsip-prinsip yang sama.

Komputasi meninggalkan layar datar. Apple Vision Pro mulai dijual pada 2 Februari 2024 dan menjadikan "komputasi spasial" istilah umum [9]; jam tangan, *earbud*, dan mobil membawa antarmukanya masing-masing. Bab 15 membahasnya.

AI generatif menjadi antarmuka. Pada Februari 2026, ChatGPT sendiri melaporkan 900 juta pengguna mingguan [10]. Kotak teks yang menjawab dalam paragraf, kopilot di dalam *spreadsheet*, dan *agen* yang memesankan tiket atas nama pengguna adalah gaya interaksi baru dengan masalah lama: jurang eksekusi menjadi "apa yang harus saya tanyakan?" dan jurang evaluasi menjadi "benarkah jawaban ini, dan apa yang baru saja dilakukan agen itu?". Riset atas pertanyaan-pertanyaan ini sudah dimulai sebelum ledakannya; delapan belas pedoman interaksi manusia-AI dari Amershi dkk. di CHI 2019 tetap menjadi titik awal baku [11]. Bab 13 dikhususkan untuk hal ini.

Komunitas riset lebih besar daripada sebelumnya. Konferensi ACM CHI, forum utama bidang ini, diselenggarakan di Barcelona pada 13–17 April 2026, menerima 1.702 makalah penuh dari 6.730 kiriman (25,3%), dan menyambut 5.254 peserta di lokasi [12]. Alat desain berubah secepat itu pula: di Config 2026 (23–24 Juni 2026), Figma mengumumkan agen AI di dalam kanvas desain, *code layers*, dan desain gerak [13]. Bab 6 memakai alat-alat semacam itu; buku ini tidak pernah membiarkannya menggantikan pengamatan terhadap orang sungguhan.

1.4 Mengapa Antarmuka Penting di Indonesia pada 2026

1.4.1 Siapa yang daring, dan dengan apa

Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) merilis survei 2026-nya pada 22 Mei 2026: 235,2 juta penduduk Indonesia, atau 81,72% dari 287,9 juta jiwa, memakai internet, sekitar enam juta lebih banyak daripada setahun sebelumnya [14]. Penetrasinya berkisar dari 85,95% di Jawa hingga 69,74% di Maluku dan Papua. Di antara orang yang masih luring, 34% mengatakan tidak punya perangkat dan 31,5% mengatakan kurang literasi digital [14]. Angka kedua itu adalah angka IMK: sepertiga orang yang belum daring tertahan bukan oleh kabel atau uang, melainkan oleh antarmuka yang menuntut terlalu banyak dari pengguna pemula.

Perangkatnya sebagian besar ponsel Android. Pada Agustus 2026 StatCounter mencatat 77,07% tayangan halaman web seluler di Indonesia berasal dari Android dan 22,88% dari iOS [15]. Layanan yang sama menghitung 60,59% dari seluruh tayangan halaman di Indonesia berasal dari komputer desktop dan 38,98% dari ponsel pada Agustus 2026 [16]. Angka kedua ini sering mengejutkan, dan ia mengajarkan pelajaran yang patut dipelajari sejak awal: StatCounter menghitung *tayangan halaman di situs web yang memasang kode pelacaknya*, bukan orang, dan pengguna desktop yang intens di kantor menayangkan banyak halaman. Sebagian besar kehidupan daring sehari-hari orang Indonesia berlangsung di dalam aplikasi, yang tidak terlihat oleh penghitung semacam itu. Selalu tanyakan apa yang dihitung sebuah statistik sebelum merancang berdasarkan angka itu.

Penggunanya juga makin tua. Menurut Badan Pusat Statistik (BPS), penduduk berusia enam puluh tahun ke atas mencapai 11,97% dari populasi pada 2025, yang berarti Indonesia telah memasuki fase *ageing population* (penuaan penduduk); proporsinya berkisar dari 17,83% di DI Yogyakarta hingga 6,71% di Papua Tengah [17]. Bab 2 menjelaskan bagaimana penglihatan, pendengaran, memori, dan gerak berubah seiring usia, dan mengapa layar yang dirancang anak dua puluhan untuk anak dua puluhan gagal melayani bagian negeri ini yang terus bertambah.

1.4.2 Hukum kini menuntut antarmuka yang baik

Mutu antarmuka telah menjadi urusan hukum. Undang-Undang Nomor 8 Tahun 2016 tentang Penyandang Disabilitas mengakui hak atas aksesibilitas, termasuk informasi dan komunikasi yang aksesibel [18], dan Peraturan Pemerintah Nomor 42 Tahun 2020 merinci aksesibilitas pelayanan publik bagi penyandang disabilitas [19]. Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi (UU PDP) mensyaratkan, di antara dasar pemrosesan yang sah, persetujuan yang sah secara eksplisit untuk tujuan tertentu; peraturan pelaksananya, Peraturan Pemerintah Nomor 33 Tahun 2026, ditandatangani pada 16 Juli 2026 dan berlaku mulai 16 Januari 2027 [20], [21]. Layar persetujuan yang tidak dapat dipahami orang, dengan demikian, bukan hanya desain yang buruk; ia juga dasar hukum yang lemah. Bab 12 membahas desain persetujuan secara mendalam.

Di luar negeri, *European Accessibility Act* (Directive (EU) 2019/882) berlaku sejak 28 Juni 2025 untuk produk dan layanan seperti perdagangan elektronik, perbankan, buku elektronik, dan tiket transportasi yang dijual di Uni Eropa, termasuk oleh perusahaan di luar Uni Eropa [22]. Tolok ukur teknis di balik sebagian besar aturan ini adalah *Web Content Accessibility Guidelines* (WCAG) dari W3C. WCAG 2.2 adalah *Recommendation* yang berlaku saat ini [23]; penggantinya, WCAG 3.0, masih berupa *working draft* yang terakhir diperbarui untuk ditinjau pada September 2026, dan W3C menyatakan bahwa standar itu belum akan rampung dalam beberapa tahun ke depan [24]. Bab 11 membahas WCAG 2.2 kriteria demi kriteria.

1.4.3 Cara kerja buku ini

Setiap bab menempuh jalan yang sama. Bab dimulai dengan situasi dari kehidupan sehari-hari di Indonesia, menjelaskan gagasannya dengan kata-kata sederhana, menyajikan teori beserta bukti dan sumbernya, mengubahnya menjadi metode praktis lengkap dengan templat dan daftar periksa, dan, bila sebuah program membantu, membangun program pendamping kecil dalam Java dan Python yang keluarannya dicetak persis seperti yang dihasilkan program. Subbab *IMK dalam Praktik: Profitina* menunjukkan di mana gagasan bab itu muncul dalam produk Indonesia yang nyata, disusul kesalahan yang sering terjadi, ringkasan, soal latihan, dan penyelesaian lengkapnya. Programnya sengaja dibuat kecil: tujuannya bukan membangun sistem warung, melainkan membuat IMK dapat diukur. Hukum Fitts dan Hick menjadi kalkulator di Bab 2; evaluasi heuristik menjadi pembuat laporan di Bab 3; catatan riset menjadi papan afinitas di Bab 4.

1.5 Meja Kerja Anda: Apa yang Dipasang dan Mengapa

Pekerjaan pertama seorang tukang kayu di bengkel baru adalah menyiapkan meja kerjanya. Meja kerja kita punya tiga alat utama, yang dirangkum pada **Tabel 1.2**. Versinya adalah versi terkini per September 2026; rilis tambalan yang lebih baru (JDK 25.0.5 atau Python 3.14.8) bekerja persis sama.

Tabel 1.2 Alat-alat di meja kerja dan versi yang dipakai di seluruh buku.

Alat	Apa itu	Versi yang dipakai di buku ini
JDK (Java Development Kit)	Kompilator Java `javac`, *runtime* Java `java`, dan pustaka standarnya	JDK 25 LTS, pembaruan 25.0.4 (Juli 2026) [25], [26]
Python	Interpreter Python `python` dan pustaka standarnya, dengan `pip` dan `venv`	3.14.7 (5 Agustus 2026) [27]
Visual Studio Code	Editor kode gratis untuk Windows, macOS, dan Linux	1.139 (23 September 2026) [28]
Extension Pack for Java	Menambahkan penyuntingan, eksekusi, *debugging*, dan pengujian Java ke VS Code	dari tampilan Extensions [29]
Ekstensi Python	Menambahkan penyuntingan Python (Pylance), eksekusi, dan *debugging* ke VS Code	dari tampilan Extensions [30]

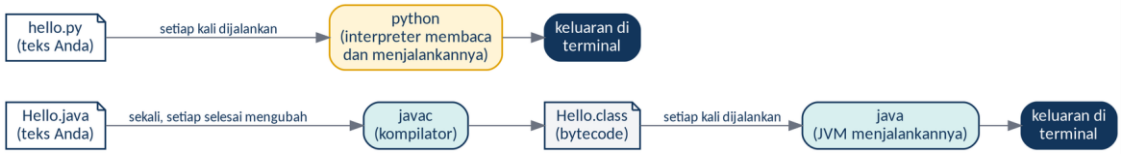
Mengapa JDK 25? Java mendapat versi baru setiap enam bulan, dan setiap dua tahun salah satunya menjadi rilis *long-term support* (LTS) yang menerima pembaruan keamanan selama bertahun-tahun. JDK 25 dirilis pada 16 September 2025 sebagai LTS terkini [26]; Oracle menyediakan pembaruan gratis untuknya setidaknya sampai September 2028, dan pembaruan Juli 2026 adalah 25.0.4 [25]. JDK 27 terbit pada 15 September 2026 [31], tetapi itu rilis jangka pendek, dan untuk buku yang programnya harus tetap berjalan bertahun-tahun, LTS adalah pilihan yang tepat. Setiap program di buku ini hanya memakai fitur yang sudah ada sejak Java 17, jadi JDK 17 dan 21 juga bisa dipakai; JDK 25 hanyalah pilihan terbaik untuk dipasang hari ini.

Ada beberapa *build* dari JDK yang sama, semuanya dibuat dari kode sumber terbuka OpenJDK: milik Oracle sendiri, Eclipse Temurin dari proyek Adoptium, milik Microsoft, Amazon Corretto, dan paket-paket distribusi Linux. Untuk keperluan kita semuanya berperilaku sama. Petunjuk di buku ini memakai **Eclipse Temurin** karena gratis untuk keperluan apa pun, punya pemasang yang sederhana untuk setiap sistem, dan tersedia di pengelola paket ketiga sistem; rilis keamanan September 2026 adalah Temurin 25.0.4.1 [32].

Mengapa Python 3.14? Python 3.14 pertama kali dirilis pada 7 Oktober 2025 dan membawa, antara lain, *template string*, evaluasi anotasi yang ditunda, *build* tanpa *global interpreter lock* yang didukung resmi, dan *shell* interaktif yang lebih ramah dengan warna [33]. Program-program di buku ini tidak memerlukan fitur-fitur itu; semuanya memakai Python yang biasa dan sudah lama mapan, dan juga berjalan di Python 3.11 atau yang lebih baru. Kita memasang 3.14 karena versi itulah yang terkini dan karena bab-bab selanjutnya memakai pustaka yang diuji dengannya.

Mengapa VS Code? Karena gratis, berjalan di ketiga sistem, menangani Java dan Python sama baiknya, dan punya terminal bawaan. Editor lain juga bisa; IntelliJ IDEA dan PyCharm sangat baik. Petunjuk di buku ini memakai VS Code agar satu set gambar dan nama menu cocok untuk semua orang.

Sebelum memasang apa pun, ada baiknya kita tahu apa yang terjadi ketika sebuah program "dijalankan" di masing-masing bahasa, karena perbedaannya menjelaskan sebagian besar pesan galat di hari pertama. **Gambar 1.2** menunjukkan kedua jalurnya.



Gambar 1.2 Bagaimana program Java dan program Python dijalankan. Java dikompilasi sekali oleh `javac` menjadi *bytecode*, yang dijalankan oleh Java Virtual Machine; Python dibaca dan dijalankan oleh interpreter setiap kali.

Program Java adalah teks di dalam berkas seperti `Hello.java`. Kompilator `javac` memeriksanya dan menerjemahkannya menjadi *bytecode*, sekumpulan instruksi ringkas yang disimpan di `Hello.class`. Perintah `java` menyalakan Java Virtual Machine (JVM), yang menjalankan *bytecode* itu. Anda mengompilasi sekali setiap selesai mengubah, dan menjalankannya sesering yang Anda mau. Program Python adalah teks di dalam berkas seperti `hello.py`, dan perintah `python` membaca serta menjalankannya langsung setiap kali; kesalahan yang ditemukan Java saat kompilasi baru ditemukan Python ketika ia mencapai baris yang salah. Sejak Java 11 Anda juga bisa menjalankan satu berkas Java tanpa mengompilasinya lebih dulu, dengan `java Hello.java`: JVM mengompilasinya di memori lalu menjalankannya [34]. Cara ini praktis untuk percobaan kecil; untuk program yang terdiri atas beberapa berkas, kita mengompilasi dengan `javac`.

1.6 Memasang di Windows 10 dan 11

1.6.1 JDK

Cara termudah adalah **winget**, pengelola paket yang sudah ada di Windows 10 dan 11. Buka **PowerShell** (tekan tombol Windows, ketik `powershell`, tekan Enter) lalu ketik:

```
PS C:\Users\you> winget install EclipseAdoptium.Temurin.25.JDK
```

Kalau Anda lebih suka mengeklik, buka `https://adoptium.net`, pilih **Temurin 25 (LTS)**, **Windows**, dan **x64** (atau **aarch64** pada laptop ARM), unduh pemasang `.msi` lalu jalankan. Di halaman **Custom Setup**, pastikan **Add to PATH** dan **Set JAVA_HOME variable** keduanya diatur ke "Will be installed on local hard drive"; yang kedua nonaktif secara bawaan, padahal beberapa alat memerlukannya.

Tutup PowerShell, buka yang baru (jendela yang dibuka sebelum pemasangan tidak mengenal program baru), lalu periksa kedua perintah:

```
PS C:\Users\you> java -version
openjdk version "25.0.4" 2026-07-21 LTS
PS C:\Users\you> javac -version
javac 25.0.4
```

Baris pertama dari `java -version` itulah yang penting; dua baris berikutnya menyebutkan *build*-nya, dan nomor pembaruannya bisa lebih baru daripada 25.0.4. Kalau Windows menjawab bahwa `java` "is not recognized as the name of a cmdlet", pemasang tidak menambahkan Java ke `PATH`, yaitu daftar folder tempat Windows mencari perintah. Jalankan lagi pemasangnya dan pilih **Add to PATH**, atau tambahkan sendiri `C:\Program Files\Eclipse Adoptium\jdk-25...\bin` lewat **Settings › System › About › Advanced system settings › Environment Variables**.

1.6.2 Python

Di Windows, Python sekarang dipasang dengan **Python install manager**, program kecil yang mengunduh, memperbarui, dan berpindah-pindah antarversi Python [35]. Dapatkan dari Microsoft Store (cari "Python Install Manager"), dari `https://www.python.org/downloads/windows/`, atau dengan winget:

```
PS C:\Users\you> winget install 9NQ7512CXL7T -e --accept-package-agreements
```

Lalu buka jendela PowerShell baru, pasang Python 3.14, dan periksa:

```
PS C:\Users\you> py install 3.14
PS C:\Users\you> python --version
Python 3.14.7
```

Setelah ini, `python`, `py`, dan `python3` semuanya menjalankan Python. Kalau `python --version` malah membuka Microsoft Store, tutup semua terminal, buka yang baru, dan coba lagi; pengelola itu juga dapat memperbaiki pengaturannya dengan `py install --configure` [35].

1.6.3 VS Code

Pasang VS Code dengan `winget install Microsoft.VisualStudioCode`, atau unduh **User Installer** dari `https://code.visualstudio.com`. Di pemasangnya, centang **Add to PATH** dan **Add "Open with Code" action to Windows Explorer file context menu**: yang pertama memungkinkan Anda mengetik `code .` di terminal untuk membuka folder saat ini, dan yang kedua memungkinkan Anda mengeklik kanan sebuah folder dan membukanya di VS Code.

1.7 Memasang di macOS

1.7.1 JDK

Cara termudah di Mac adalah paket pemasang Temurin. Buka `https://adoptium.net`, pilih **Temurin 25 (LTS)**, **macOS**, dan **aarch64** untuk Mac dengan cip Apple (M1 dan sesudahnya) atau **x64** untuk Mac Intel yang lebih lama, unduh berkas `.pkg`, lalu klik dua kali. Kalau Anda sudah memakai pengelola paket Homebrew, satu perintah mengerjakan hal yang sama:

```
you@mac ~ % brew install --cask temurin@25
```

Buka **Terminal** (tekan Cmd+Space, ketik "Terminal", tekan Enter) lalu periksa:

```
you@mac ~ % java -version
openjdk version "25.0.4" 2026-07-21 LTS
you@mac ~ % javac -version
javac 25.0.4
```

Kalau ada beberapa JDK terpasang, macOS memilih yang terbaru. Perintah `/usr/libexec/java_home -V` menampilkan semuanya.

1.7.2 Python

macOS mungkin sudah memiliki perintah `python3`, tetapi itu versi lama yang ditujukan untuk perkakas pengembang Apple; jangan membangun di atasnya. Unduh pemasang macOS untuk Python 3.14.7 dari `https://www.python.org/downloads/macos/` lalu klik dua kali; paket itu universal dan berjalan langsung di cip Apple maupun Intel. Di akhir pemasangan terbuka folder **Applications › Python 3.14**: klik dua kali **Install Certificates.command** di sana satu kali, agar Python dapat membuat koneksi internet yang aman. Dengan Homebrew, `brew install python@3.14` juga bisa. Lalu periksa:

```
you@mac ~ % python3 --version
Python 3.14.7
```

Di macOS dan Linux perintahnya `python3`, bukan `python`. Buku ini menulis `python3` dalam perintah untuk macOS dan Linux, dan `python` untuk Windows.

1.7.3 VS Code

Unduh VS Code untuk macOS dari `https://code.visualstudio.com`, buka berkas unduhannya, lalu seret **Visual Studio Code** ke **Applications**. Agar bisa mengetik `code .` di terminal, jalankan VS Code, tekan Cmd+Shift+P untuk membuka Command Palette, lalu jalankan **Shell Command: Install 'code' command in PATH**.

1.8 Memasang di Linux

Sebagian besar distribusi Linux menyediakan versi Java dan Python yang baru di repositori paketnya sendiri, dan itulah pilihan terbaik karena sistem akan terus memperbaruinya. Ubuntu 26.04 LTS memakai OpenJDK 25 sebagai Java bawaannya [36] dan Python 3.14 sebagai `python3` bawaannya, tetapi tidak memasang `javac` maupun modul `venv` Python kecuali Anda memintanya:

```
$ sudo apt update
$ sudo apt install openjdk-25-jdk python3 python3-venv python3-pip
$ java -version
openjdk version "25.0.4" 2026-07-21
$ javac -version
javac 25.0.4
$ python3 --version
Python 3.14.7
```

Di Fedora padanannya adalah `sudo dnf install java-25-openjdk-devel python3 python3-pip`. Pada distribusi lama yang repositorinya tidak punya JDK 25, pasang Temurin dari repositori Adoptium sendiri dengan mengikuti petunjuk di `https://adoptium.net/installation`, atau tetap pakai JDK 21 bawaan distribusi: setiap program di buku ini berjalan di JDK 17 atau yang lebih baru. Hal yang sama berlaku untuk Python: Ubuntu 24.04 membawa Python 3.12, yang menjalankan semua program di buku ini, jadi Anda boleh tetap memakainya. Jangan pernah menghapus atau mengganti Python sistem di Linux; sebagian sistem operasi bergantung padanya.

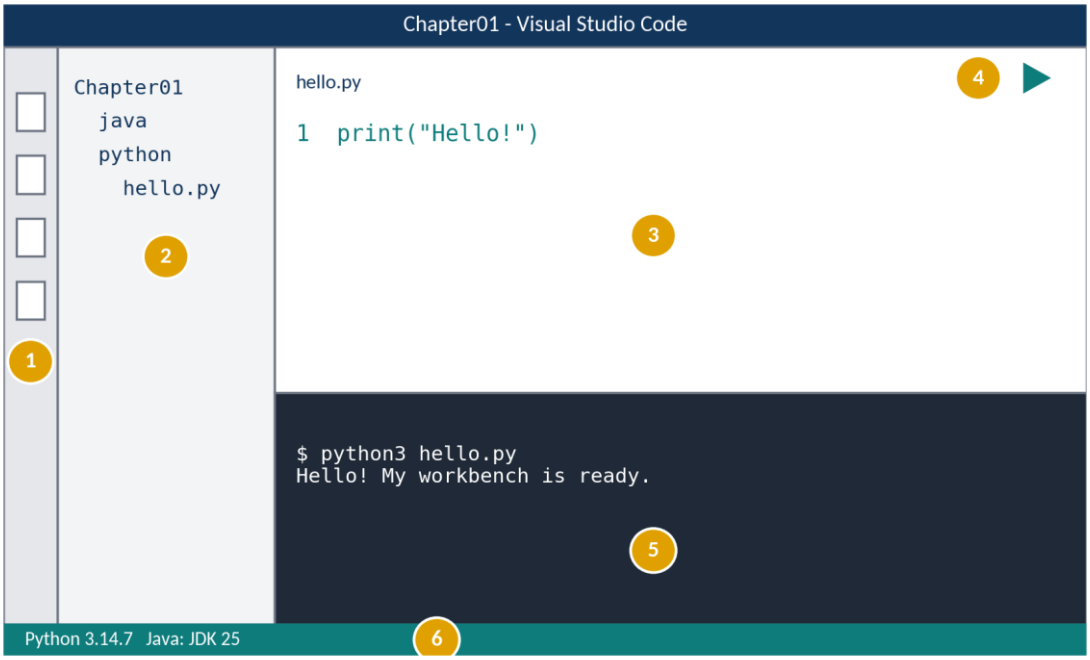
Untuk VS Code, unduh paket `.deb` (Ubuntu, Debian, Mint) atau paket `.rpm` (Fedora, openSUSE) dari `https://code.visualstudio.com` lalu pasang dengan `sudo apt install ./code_*.deb` atau `sudo dnf install ./code-*.rpm`. Paket itu juga menambahkan repositori Microsoft, sehingga VS Code kemudian diperbarui bersama sistem lainnya.

1.9 VS Code: Dua Ekstensi dan Sebuah Tur

Jalankan VS Code. Klik ikon **Extensions** di Activity Bar di sebelah kiri (atau tekan Ctrl+Shift+X, di macOS Cmd+Shift+X) lalu pasang dua ekstensi, keduanya terbitan Microsoft:

- **Extension Pack for Java** (pengenal `vscjava.vscode-java-pack`). Paket ini menghimpun dukungan bahasa Java, *debugger*, *test runner*, dukungan Maven dan Gradle, serta pengelola proyek [29]. Ia menemukan sendiri JDK yang Anda pasang; kalau ada beberapa, perintah **Java: Configure Java Runtime** di Command Palette memungkinkan Anda memilih.
- **Python** (pengenal `ms-python.python`). Ekstensi ini membawa **Pylance**, yang melengkapi nama dan menandai kesalahan selagi Anda mengetik, serta **Python Debugger** [30].

Gambar 1.3 menunjukkan bagian-bagian jendela yang dirujuk buku ini. Activity Bar (1) berpindah antara Explorer, Search, Source Control, Run and Debug, dan Extensions. Explorer (2) menampilkan berkas-berkas di folder yang Anda buka. Editor (3) adalah tempat Anda mengetik. Tombol Run (4), sebuah segitiga di kanan atas editor, menjalankan berkas yang sedang terbuka. Terminal terpadu (5), dibuka dengan **Terminal › New Terminal** atau Ctrl+`, adalah jendela PowerShell, zsh, atau bash biasa yang sudah berada di folder proyek Anda. Status Bar (6) menunjukkan, antara lain, interpreter Python dan *runtime* Java mana yang dipakai VS Code; mengeklik versi Python memungkinkan Anda memilih yang lain.



Gambar 1.3 Bagian-bagian Visual Studio Code yang dipakai di buku ini: (1) Activity Bar, (2) Explorer, (3) editor, (4) tombol Run, (5) terminal terpadu, (6) Status Bar.

Selalu buka sebuah *folder* di VS Code (**File › Open Folder**), bukan satu berkas saja. Folder itu menjadi ruang kerja Anda: Explorer menampilkannya, terminal dimulai di dalamnya, dan ekstensi Java memperlakukan berkas-berkas `.java` di dalamnya sebagai satu proyek. Untuk buku ini, bukalah folder bab yang sedang Anda baca, misalnya `Chapter01`.

Sepatah kata tentang *virtual environment* Python. *Virtual environment* adalah folder pribadi, menurut kebiasaan bernama `.venv`, dengan salinan `python`-nya sendiri dan tempat sendiri untuk paket yang dipasang, sehingga pustaka satu proyek tidak bertabrakan dengan pustaka proyek lain [37]. Bab-bab awal buku ini hanya memakai pustaka standar dan tidak memerlukan *environment*; bab-bab yang memasang pustaka (misalnya Flask untuk antarmuka web kecil di bab-bab

selanjutnya) akan meminta Anda membuatnya dengan `python -m venv .venv` dan mengaktifkannya dengan `.venv\Scripts\Activate.ps1` di Windows atau `source .venv/bin/activate` di macOS dan Linux. Program pemeriksa di Subbab 1.10 melaporkan apakah sebuah *environment* sedang aktif, tetapi untuk Bab 1 sampai 4 hal itu tidak berpengaruh.

1.10 Program-Program Pertama Anda

1.10.1 Halo, dalam Java

Buat folder `Chapter01` dengan subfolder `java` dan `python`, atau buka folder `Chapter01` dari kode buku ini. Di VS Code pilih **File › New File**, ketik program pada **Kode 1.1**, lalu simpan sebagai `java/Hello.java`. Di Java, nama berkas harus sama dengan nama kelas diikuti `.java`, dengan huruf besar yang sama.

Kode 1.1 `Hello.java`: program Java pertama, yang juga melaporkan versi Java dan sistem operasinya.

```java
/** The first program: say hello and show which Java runs it. */
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello! My HCI workbench is ready.");
        System.out.println("Java version: " + System.getProperty("java.version"));
        System.out.println("Running on:   " + System.getProperty("os.name"));
    }
}
```

Baris demi baris: `public class Hello` mendeklarasikan sebuah *kelas* (*class*), satuan tempat Java menyimpan kode, bernama `Hello`. Komentar di atasnya, di antara `/**` dan `*/`, ditujukan untuk manusia dan diabaikan kompilator. `public static void main(String[] args)` adalah *metode main*, tempat setiap aplikasi Java dimulai; kata-kata `public static void` menyatakan bahwa metode ini boleh dipanggil dari luar, tidak memerlukan objek, dan tidak mengembalikan apa pun, sedangkan `String[] args` menerima kata-kata yang diketik sesudah nama program di baris perintah. `System.out.println(...)` mencetak satu baris. `System.getProperty("java.version")` menanyakan versinya kepada JVM yang sedang berjalan, sehingga program memberi tahu Java mana yang sebenarnya menjalankannya; itulah hal pertama yang perlu diperiksa ketika sesuatu berperilaku aneh.

Buka terminal di VS Code (Ctrl+`), masuk ke folder `java`, kompilasi, lalu jalankan:

```
$ cd java
$ javac -d out Hello.java
$ java -cp out Hello
Hello! My HCI workbench is ready.
Java version: 25.0.4
Running on:   Windows 11
```

Opsi `-d out` meminta `javac` menaruh `Hello.class` hasil kompilasi di folder bernama `out`, agar folder sumber tetap rapi, dan `-cp out` (*class path*) memberi tahu `java` di mana menemukannya. Di macOS baris terakhir berbunyi `Mac OS X`, di Linux `Linux`. Anda juga bisa menekan tombol Run (4 pada **Gambar 1.3**); ekstensi Java mengompilasi dan menjalankan berkasnya dan menampilkan keluarannya di terminal. Dan, seperti disebut Subbab 1.5, `java Hello.java` mengompilasi dan menjalankan satu berkas dalam satu langkah.

JDK 25 juga mengizinkan program pertama yang jauh lebih pendek. Sejak JEP 512 menjadi final di JDK 25, sebuah berkas boleh hanya berisi metode `void main()` tanpa kelas yang membungkusnya, dan kelas pembantu baru bernama `IO` mencetak satu baris [38]:

```
void main() {
    IO.println("Hello!");
}
```

Ini penyederhanaan yang menyenangkan bagi pemula. Meski begitu, program-program di buku ini memakai bentuk klasik dengan kelas dan `public static void main`, karena semuanya aplikasi nyata yang terdiri atas beberapa kelas, dan karena bentuk klasik itu juga berjalan di JDK 17 dan 21.

1.10.2 Halo, dalam Python

Kode 1.2 adalah program yang sama dalam Python. Simpan sebagai `python/hello.py`.

Kode 1.2 `hello.py`: program Python pertama.

```python
"""The first program: say hello and show which Python runs it."""
import platform

print("Hello! My HCI workbench is ready.")
print("Python version: " + platform.python_version())
print("Running on:     " + platform.system())
```

Teks di antara tanda kutip tiga di bagian atas adalah *docstring*, keterangan tentang berkas itu. `import platform` memuat modul dari pustaka standar yang mengenal komputer dan interpreternya; `platform.python_version()` mengembalikan versi sebagai teks dan `platform.system()` sistem operasinya. Tidak ada kelas dan tidak ada metode `main`: Python menjalankan berkas itu begitu saja dari atas ke bawah. Jalankan dari terminal, di folder `python`:

```
$ cd ../python
$ python3 hello.py
Hello! My HCI workbench is ready.
Python version: 3.14.7
Running on:     Linux
```

Di Windows ketik `python hello.py`, dan baris terakhir berbunyi `Windows`; di macOS berbunyi `Darwin`, nama inti macOS. Di VS Code, buka `hello.py` lalu tekan tombol Run; kalau VS Code menanyakan interpreter mana yang dipakai, pilih Python 3.14.

1.10.3 Memeriksa seluruh meja kerja

Daripada percaya begitu saja bahwa semuanya berjalan lancar, kita menulis program yang memeriksanya. **Kode 1.3** melaporkan versi Python, apakah sebuah *virtual environment* sedang aktif, dan apakah *runtime* Java dapat ditemukan; **Kode 1.4** mengerjakan hal yang sama dari sisi Java.

Kode 1.3 `check_setup.py`: memeriksa versi Python dan mencari Java.

```
"""Check the workbench: the Python version, the virtual environment and Java."""
import platform
import shutil
import subprocess
import sys

RECOMMENDED = (3, 14)      # the version this book is written for
MINIMUM = (3, 11)          # the oldest version the book's code runs on


def java_version():
    """Ask the 'java' command for its version; None when it is not found."""
    if shutil.which("java") is None:
        return None
    done = subprocess.run(["java", "-version"], capture_output=True, text=True)
    for line in (done.stderr + done.stdout).splitlines():  # java uses stderr
        if " version " in line and '"' in line:
            return line.split('"')[1]         # openjdk version "25.0.4" ...
    return "unknown"


def main():
    version = sys.version_info[:2]
    in_venv = sys.prefix != sys.base_prefix
    print(f"Python version : {platform.python_version()}")
    print(f"System         : {platform.system()} ({platform.machine()})")
    print(f"Virtual env.   : {'yes' if in_venv else 'no (fine for this book)'}")
    java = java_version()
    print(f"Java version   : {java or 'NOT FOUND - install JDK 25'}")
    if version < MINIMUM:
        print("Result         : Python is too old, please install Python 3.14")
    elif java is None:
        print("Result         : Python is fine; now install the JDK")
    elif version < RECOMMENDED:
        print("Result         : works; Python 3.14 is recommended")
    else:
        print("Result         : your workbench is ready")


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

Program ini memakai empat modul standar. `sys.version_info[:2]` adalah versi sebagai pasangan angka seperti `(3, 14)`, dan Python membandingkan pasangan semacam itu elemen demi elemen, sehingga `(3, 12) < (3, 14)` bernilai benar. `sys.prefix` dan `sys.base_prefix` berbeda tepat ketika sebuah *virtual environment* aktif. `shutil.which("java")` mencari perintah `java` di `PATH` dan mengembalikan `None` bila tidak ada. `subprocess.run` menjalankan `java -version` sebagai program tersendiri dan menangkap apa yang dicetaknya; Java mencetak versinya ke aliran *error*, bukan ke keluaran biasa, sebuah kebiasaan historis, dan itu sebabnya fungsi ini membaca `done.stderr`. Fungsi itu kemudian memilih baris yang memuat kata `version` dan tanda kutip, lalu mengembalikan teks di antara pasangan tanda kutip pertama.

Kode 1.4 `CheckSetup.java`: memeriksa versi Java dan mencari Python.

```java
/** Check the workbench: the Java version, and whether Python can be found. */
public class CheckSetup {
    static final int RECOMMENDED = 25;       // the JDK this book is written for
    static final int MINIMUM = 17;           // the oldest JDK the code runs on

    /** Ask a command for its version; null when the command is not found. */
    static String versionOf(String... command) {
        try {
            Process p = new ProcessBuilder(command).redirectErrorStream(true)
                                                   .start();
            String out = new String(p.getInputStream().readAllBytes()).trim();
            p.waitFor();
            return out.isEmpty() ? null : out.lines().findFirst().get();
        } catch (Exception e) {
            return null;                     // not installed, or not on PATH
        }
    }

    public static void main(String[] args) {
        int feature = Runtime.version().feature();
        System.out.println("Java version   : " + Runtime.version());
        System.out.println("Java vendor    : " + System.getProperty("java.vendor"));
        System.out.println("System         : " + System.getProperty("os.name")
                           + " (" + System.getProperty("os.arch") + ")");
        String python = versionOf("python3", "--version");
        if (python == null) {
            python = versionOf("python", "--version");  // the name on Windows
        }
        System.out.println("Python         : "
                           + (python != null ? python : "NOT FOUND"));
        if (feature < MINIMUM) {
            System.out.println("Result         : Java is too old, install JDK 25");
        } else if (python == null) {
            System.out.println("Result         : Java is fine; now install Python");
        } else if (feature < RECOMMENDED) {
            System.out.println("Result         : works; JDK 25 is recommended");
        } else {
            System.out.println("Result         : your workbench is ready");
        }
    }
}
```

`Runtime.version().feature()` adalah nomor versi utama Java yang sedang berjalan, 25 untuk JDK 25. `ProcessBuilder` menjalankan program lain; `redirectErrorStream(true)` menggabungkan aliran *error*-nya ke keluaran biasanya, sehingga versinya tetap ditemukan ke aliran mana pun Python mencetaknya. Kalau perintahnya tidak ada, menjalankannya melempar *exception*, yang diubah oleh `catch` menjadi `null`, artinya “tidak ditemukan”. Karena Python di Windows biasanya bernama `python` dan di macOS serta Linux `python3`, program ini mencoba keduanya.

Pada meja kerja yang sudah siap, kedua program mencetak:

```
$ python3 check_setup.py
Python version : 3.14.7
System         : Linux (x86_64)
Virtual env.   : no (fine for this book)
Java version   : 25.0.4
Result         : your workbench is ready
$ java -cp out CheckSetup
Java version   : 25.0.4+8-LTS
Java vendor    : Eclipse Adoptium
System         : Linux (amd64)
Python         : Python 3.14.7
Result         : your workbench is ready
```

Nomor *build* persis sesudah tanda + berbeda dari komputer ke komputer. Baris terakhirlah yang perlu diperhatikan. Kalau isinya selain "your workbench is ready", baris itu juga menyebutkan apa yang harus dipasang, dan program-program di buku ini tetap berjalan selama isinya "works".

1.11 Program IMK Pertama: Menghitung Biaya Klik

1.11.1 Keystroke-Level Model

Perdebatan Dimas dengan dirinya sendiri di Subbab 1.1 ("nanti juga terbiasa" melawan "halamannya lambat") dapat diselesaikan tanpa menebak. Pada 1980 Stuart Card, Thomas Moran, dan Allen Newell menerbitkan *Keystroke-Level Model* (KLM), yang memprediksi berapa lama seorang pengguna berpengalaman membutuhkan waktu untuk menyelesaikan suatu tugas pada antarmuka tertentu, bahkan sebelum antarmuka itu dibangun [39]. Model ini kemudian menjadi bagian dari keluarga model GOMS yang lebih besar dalam *The Psychology of Human-Computer Interaction* [40] dan masih diajarkan dan dipakai karena sederhana, murah, dan akurat secara mengejutkan untuk tugas rutin.

Gagasannya adalah menuliskan tugas sebagai rangkaian *operator* dasar lalu menjumlahkan waktunya. **Tabel 1.3** mendaftar lima operator yang kita pakai, dengan waktu satuan yang dianjurkan Card, Moran, dan Newell serta diulang dalam panduan praktis Kieras [39], [41].

Tabel 1.3 Operator Keystroke-Level Model dan waktu satuan bakunya.

Operator	Arti	Waktu (s)
K	Menekan satu tombol keyboard (pengetik terampil rata-rata)	0.28
P	Menunjuk sasaran di layar dengan mouse	1.10
B	Menekan atau melepas tombol mouse (satu klik adalah B B)	0.10
H	Memindahkan tangan dari keyboard ke mouse atau sebaliknya	0.40
M	Persiapan mental untuk langkah berikutnya: memutuskan, mencari, mengingat	1.35

Jika sebuah tugas memerlukan n_K ketukan tombol, n_P gerakan menunjuk, dan seterusnya, prediksi waktunya adalah

$$T = n_K t_K + n_P t_P + n_B t_B + n_H t_H + n_M t_M \tag{1.1}$$

dengan $t_K = 0.28$ s, $t_P = 1.10$ s, $t_B = 0.10$ s, $t_H = 0.40$ s, dan $t_M = 1.35$ s sebagai waktu satuan. Persamaan (1.1) tidak lebih dari "hitung tiap jenis langkah lalu kalikan dengan lamanya".

Bagian tersulit adalah menempatkan operator M. Kaidah praktis Kieras adalah menaruh satu M sebelum setiap *bongkahan* (*chunk*) tugas yang harus dipikirkan pengguna, misalnya memutuskan menu apa berikutnya, tetapi tidak di dalam bongkahan yang dikerjakan tangan secara otomatis, seperti dua klik pada klik ganda [41]. Kita menaruh satu M sebelum setiap menu yang dipesan dan satu sebelum *Save*.

1.11.2 Dua halaman, dihitung dengan tangan

Ambil pesanan yang paling sering datang saat makan siang: dua es teh, satu nasi goreng, satu ayam geprek. Pada halaman pertama Dimas (desain A), setiap menu memerlukan: M (memutuskan menu), P B B (membuka *dropdown*), P B B (memilih menu), P B B (mengeklik kotak jumlah), H (tangan ke keyboard), K (mengetik angka), H (tangan kembali ke mouse), P B B (mengeklik *Add*). Jadi 1 M, 4 P, 8 B, 2 H, dan 1 K per menu. Tiga menu ditambah M P B B untuk *Save* memberikan

$$n_M = 4, \quad n_P = 13, \quad n_B = 26, \quad n_H = 6, \quad n_K = 3.$$

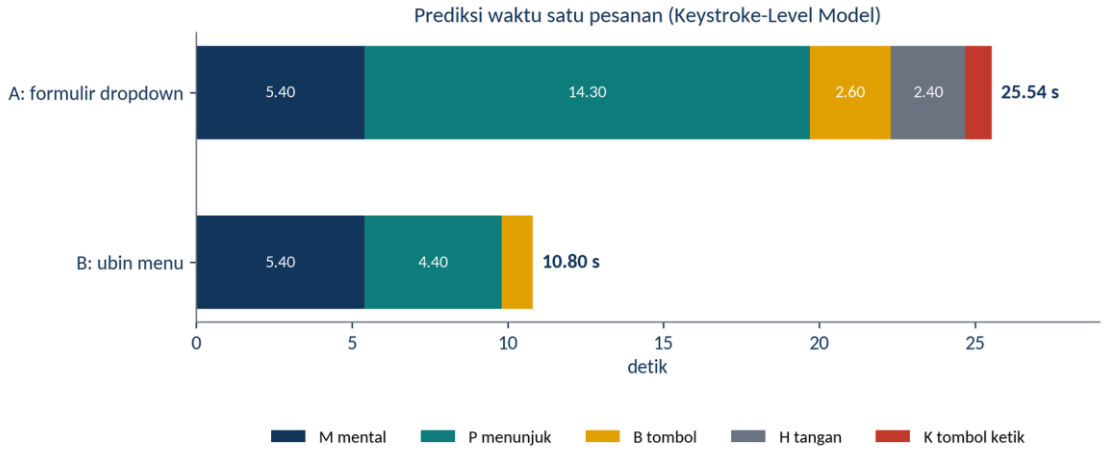
Dengan Persamaan (1.1):

$$T_A = 4(1.35) + 13(1.10) + 26(0.10) + 6(0.40) + 3(0.28)$$
$$= 5.40 + 14.30 + 2.60 + 2.40 + 0.84 = 25.54 \text{ s}.$$

Pada halaman ubin (desain B), es teh adalah M P B B B B (berpikir, menunjuk ubin, mengekliknya dua kali), dua menu lainnya masing-masing M P B B, dan *Save* adalah M P B B:

$$T_B = 4(1.35) + 4(1.10) + 10(0.10) = 5.40 + 4.40 + 1.00 = 10.80 \text{ s}.$$

Ubin menghemat $25.54 - 10.80 = 14.74$ detik, atau 57,7% waktu, pada setiap pesanan seperti itu. **Gambar 1.4** memperlihatkan ke mana waktu itu pergi: empat operator M sama pada kedua desain, karena proses berpikirnya tidak berubah; penghematan hampir seluruhnya datang dari menunjuk, operator yang diperlukan desain A sebanyak tiga belas kali.



Gambar 1.4 Prediksi waktu satu pesanan pada kedua halaman, dipecah per operator: menunjuk (P) mendominasi desain A.

1.11.3 Programnya

Menghitung dengan tangan tidak masalah untuk satu tugas, tetapi melelahkan untuk sepuluh. Program di subbab ini membaca sebuah berkas teks kecil yang menggambarkan setiap desain sebagai daftar operator, lalu mencetak jumlah tiap operator, prediksi waktu, dan penghematan desain yang lebih cepat selama sebulan kerja. **Kode 1.5** adalah berkas untuk pesanan Bu Ratna; "2B" adalah singkatan dari "B B", dan semua yang ada setelah `#` adalah komentar untuk manusia.

Kode 1.5 `data/orders.txt`: kedua desain yang ditulis sebagai operator Keystroke-Level Model.

```
  # Pesanan "2 es teh, 1 nasi goreng, 1 ayam geprek" pada dua desain.
  # K tombol ketik, P menunjuk, B tombol mouse ditekan atau dilepas, H tangan
  # pindah antara keyboard dan mouse, M persiapan mental. "2B" berarti "B B".
  A: dropdown form
    M P 2B P 2B P 2B H K H P 2B     # es teh: daftar menu, jumlah, Add
    M P 2B P 2B P 2B H K H P 2B     # nasi goreng
    M P 2B P 2B P 2B H K H P 2B     # ayam geprek
    M P 2B                          # Save
  B: menu tiles
    M P 4B                          # ubin es teh, diklik dua kali
    M P 2B                          # ubin nasi goreng
    M P 2B                          # ubin ayam geprek
    M P 2B                          # Save
```

Kode 1.6 adalah versi Python-nya.

Kode 1.6 `klm.py`: membaca desain sebagai daftar operator dan memprediksi waktunya.

```python
"""Keystroke-Level Model: berapa lama seorang ahli menyelesaikan tugas?"""
import sys

# Waktu operator dalam detik, dari Card, Moran, dan Newell (1980).
TIMES = {"K": 0.28,   # menekan tombol (pengetik rata-rata)
         "P": 1.10,   # menunjuk sasaran dengan mouse
         "B": 0.10,   # menekan atau melepas tombol mouse
         "H": 0.40,   # memindah tangan antara keyboard-mouse
         "M": 1.35}   # persiapan mental untuk langkah berikut
ORDER = "KPBHM"


def count(ops):
    """Hitung operator dalam teks seperti 'M P 2B H K'."""
    counts = dict.fromkeys(ORDER, 0)
    for token in ops.split():
        times, op = int(token[:-1] or 1), token[-1]   # '2B' -> 2, 'B'
        if op not in TIMES:
            raise ValueError(f"unknown operator {token!r}")
        counts[op] += times
    return counts


def seconds(counts):
    """Prediksi waktu: jumlah tiap operator kali waktu satuannya."""
    return sum(counts[op] * TIMES[op] for op in ORDER)


def read_designs(path):
    """Baca baris 'nama', masing-masing diikuti baris operator menjorok."""
    designs = {}
    name = None
    with open(path, encoding="utf-8") as f:
        for line in f:
            text = line.split("#")[0].rstrip()
            if not text:
                continue
            if not line[0].isspace():
                name = text
                designs[name] = ""
            else:
                designs[name] += " " + text
    return {n: count(ops) for n, ops in designs.items()}
```

Kode 1.6 (Lanjutan) *klm.py: membaca desain sebagai daftar operator dan memprediksi waktunya.*

```
def main():
    path = sys.argv[1] if len(sys.argv) > 1 else "data/orders.txt"
    per_day = int(sys.argv[2]) if len(sys.argv) > 2 else 120
    designs = read_designs(path)
    print("Keystroke-Level Model, times in seconds (Card, Moran and Newell)")
    print(f"{'design':18}" + "".join(f"{op:>4}" for op in ORDER) + "   time")
    for name, counts in designs.items():
        cols = "".join(f"{counts[op]:>4}" for op in ORDER)
        print(f"{name:18}{cols}{seconds(counts):7.2f}")
    ranked = sorted(designs.items(), key=lambda d: seconds(d[1]))
    (fast, b), (_, a) = ranked[0], ranked[-1]        # tercepat, terlambat
    saved = seconds(a) - seconds(b)
    print(f"{fast[0]} saves {saved:.2f} s per order "
          f"({100 * saved / seconds(a):.1f}%).")
    minutes = saved * per_day / 60
    print(f"At {per_day} orders a day: {minutes:.1f} minutes a day, "
          f"{minutes * 26 / 60:.1f} hours a month (26 days).")


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

Dictionary `TIMES` menyimpan waktu satuan dari **Tabel 1.3**. Fungsi `count` memecah teks seperti `M P 2B` menjadi token; karakter terakhir sebuah token adalah operatornya dan apa pun di depannya adalah jumlah pengulangan, sehingga `token[:-1] or 1` mengubah `"2B"` menjadi 2 dan `"B"` menjadi 1 (string kosong dianggap salah di Python, dan `or` lalu memberikan 1). Huruf yang tidak dikenal memicu galat alih-alih diabaikan, karena salah ketik yang diam-diam menghapus satu langkah akan membuat sebuah desain tampak lebih cepat daripada kenyataannya. Fungsi `seconds` adalah Persamaan (1.1) yang ditulis dalam satu baris. Fungsi `read_designs` memperlakukan baris yang dimulai di margin kiri sebagai nama desain baru dan setiap baris menjorok sebagai tambahan operator untuk desain itu, sehingga tugas yang panjang dapat ditulis dalam beberapa baris berkomentar. Terakhir, `main` mencetak tabel, mencari desain tercepat dan terlambat dengan mengurutkan berdasarkan prediksi waktu, lalu menskalakan selisihnya ke satu hari dan ke satu bulan berisi 26 hari kerja. Dua argumen baris perintah opsional memilih berkas lain dan jumlah pesanan per hari yang lain.

Kode 1.7 adalah program yang sama dalam Java.

Kode 1.7 `Klm.java`: kalkulator Keystroke-Level Model yang sama dalam Java.

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.LinkedHashMap;
import java.util.Map;

/** Keystroke-Level Model: berapa lama seorang ahli menyelesaikan tugas? */
public class Klm {
    static final String ORDER = "KPBHM";
    // Waktu operator dalam detik, dari Card, Moran, dan Newell (1980):
    // tombol, menunjuk, tombol mouse, tangan pindah, langkah mental.
    static final double[] TIMES = {0.28, 1.10, 0.10, 0.40, 1.35};

    /** Hitung operator dalam teks seperti "M P 2B H K". */
    static int[] count(String ops) {
        int[] counts = new int[ORDER.length()];
        for (String token : ops.trim().split("\\s+")) {
            if (token.isEmpty()) continue;
            char op = token.charAt(token.length() - 1);
            String n = token.substring(0, token.length() - 1);
            int i = ORDER.indexOf(op);
            if (i < 0) throw new IllegalArgumentException("unknown " + token);
            counts[i] += n.isEmpty() ? 1 : Integer.parseInt(n);
        }
        return counts;
    }

    /** Prediksi waktu: jumlah tiap operator kali waktu satuannya. */
    static double seconds(int[] counts) {
        double total = 0;
        for (int i = 0; i < counts.length; i++) total += counts[i] * TIMES[i];
        return total;
    }

    /** Baca baris "nama", masing-masing diikuti baris operator menjorok. */
    static Map<String, int[]> readDesigns(String path) throws IOException {
        Map<String, String> ops = new LinkedHashMap<>();
        String name = null;
        for (String line : Files.readAllLines(Path.of(path))) {
            String text = line.split("#")[0].stripTrailing();
            if (text.isEmpty()) continue;
            if (!Character.isWhitespace(line.charAt(0))) {
                name = text;
                ops.put(name, "");
            } else {
                ops.put(name, ops.get(name) + " " + text);
```

Kode 1.7 (Lanjutan) *Klm.java: kalkulator Keystroke-Level Model yang sama dalam Java.*

```
            }
        }
        Map<String, int[]> designs = new LinkedHashMap<>();
        ops.forEach((n, o) -> designs.put(n, count(o)));
        return designs;
    }

    public static void main(String[] args) throws IOException {
        String path = args.length > 0 ? args[0] : "data/orders.txt";
        int perDay = args.length > 1 ? Integer.parseInt(args[1]) : 120;
        Map<String, int[]> designs = readDesigns(path);
        System.out.println("Keystroke-Level Model, times in seconds"
                           + " (Card, Moran and Newell)");
        var head = new StringBuilder(String.format("%-18s", "design"));
        for (char op : ORDER.toCharArray()) head.append(String.format("%4s", op));
        System.out.println(head + "   time");
        String slow = null, fast = null;
        for (var e : designs.entrySet()) {
            var row = new StringBuilder(String.format("%-18s", e.getKey()));
            for (int c : e.getValue()) row.append(String.format("%4d", c));
            System.out.println(row + String.format("%7.2f", seconds(e.getValue())));
            double t = seconds(e.getValue());
            if (slow == null || t > seconds(designs.get(slow))) slow = e.getKey();
            if (fast == null || t < seconds(designs.get(fast))) fast = e.getKey();
        }
        double a = seconds(designs.get(slow));
        double saved = a - seconds(designs.get(fast));
        System.out.printf("%s saves %.2f s per order (%.1f%%).%n",
                          fast.charAt(0), saved, 100 * saved / a);
        double minutes = saved * perDay / 60;
        System.out.printf("At %d orders a day: %.1f minutes a day, "
                          + "%.1f hours a month (26 days).%n",
                          perDay, minutes, minutes * 26 / 60);
    }
}
```

Versi Java menyimpan jumlah operator dalam larik `int` dengan urutan K, P, B, H, M, sehingga `ORDER.indexOf(op)` memberikan posisi sebuah operator dan `TIMES[i]` waktunya. `Files.readAllLines` membaca seluruh berkas; `LinkedHashMap` menjaga desain dalam urutan berkas, sesuatu yang tidak dilakukan `HashMap` biasa. Alih-alih mengurutkan, gelung yang mencetak baris-baris tabel sekaligus mengingat desain terlambat dan tercepat.

1.11.4 Menjalankannya

Dari folder `python`, dan dari folder `java` setelah dikompilasi, kedua program mencetak baris yang sama:

```
$ python3 klm.py
$ java -cp out Klm
```

```
Keystroke-Level Model, times in seconds (Card, Moran and Newell)
design              K   P   B   H   M   time
A: dropdown form    3  13  26   6   4  25.54
B: menu tiles       0   4  10   0   4  10.80
B saves 14.74 s per order (57.7%).
At 120 orders a day: 29.5 minutes a day, 12.8 hours a month (26 days).
```

Angka-angkanya cocok dengan hitungan tangan di Subbab 1.11.2. Pada hari dengan 120 pesanan sebesar itu, ubin mengembalikan setengah jam kepada Bu Ratna, dan lebih dari dua belas jam kerja sebulan. Untuk hari yang lebih sepi dengan 60 pesanan, `python3 klm.py data/orders.txt 60` melaporkan 14,7 menit sehari dan 6,4 jam sebulan.

Kita perlu jujur tentang apa yang dikatakan dan tidak dikatakan model ini. KLM memprediksi waktu seorang *ahli* yang tidak berbuat salah; Bu Ratna pada minggu pertamanya bukan ahli, dan setiap kesalahan (empat es teh di Subbab 1.2.2) memakan waktu lebih banyak daripada yang terlihat oleh model. Model ini tidak mengatakan apa pun tentang proses belajar, kepuasan, atau apakah pelanggan menerima makanan yang benar. Ia adalah perkiraan pertama yang cukup murah untuk dibuat sebelum menulis kode apa pun, dan dalam kasus ini ia menegaskan apa yang sudah ditunjukkan oleh pengamatan terhadap Bu Ratna: desainnya yang lambat, bukan penggunanya.

1.11.5 Mengujinya

Program yang tugasnya menyelesaikan perdebatan harus dapat dipercaya. **Kode 1.8** memeriksa penghitungan, satu klik tunggal, penolakan operator yang tidak dikenal, dan dua total di Subbab 1.11.2.

Kode 1.8 `test_klm.py`: uji kalkulator Keystroke-Level Model.

```
"""Uji Keystroke-Level Model: python -m unittest test_klm.py"""
import unittest
from klm import count, seconds, read_designs


class KlmTests(unittest.TestCase):
    def test_counting_with_repeats(self):
        c = count("M P 2B H K 3K")
        self.assertEqual((c["M"], c["P"], c["B"], c["H"], c["K"]),
                         (1, 1, 2, 1, 4))

    def test_one_click(self):
        self.assertAlmostEqual(seconds(count("M P 2B")), 2.65)

    def test_unknown_operator(self):
        with self.assertRaises(ValueError):
            count("M X")

    def test_the_two_designs(self):
        d = read_designs("data/orders.txt")
        self.assertAlmostEqual(seconds(d["A: dropdown form"]), 25.54)
        self.assertAlmostEqual(seconds(d["B: menu tiles"]), 10.80)


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

Kode 1.9 melakukan hal yang sama dalam Java tanpa pustaka pengujian apa pun, sehingga dapat berjalan di JDK polos.

Kode 1.9 `KlmTest.java`: pemeriksaan yang sama dalam Java.

```java
import java.util.Map;

/** Uji Keystroke-Level Model: java -cp out KlmTest */
public class KlmTest {
    static int passed = 0;

    static void expect(boolean ok, String what) {
        if (!ok) throw new AssertionError(what);
        passed++;
    }

    public static void main(String[] args) throws Exception {
        int[] c = Klm.count("M P 2B H K 3K");          // K P B H M
        expect(c[0] == 4 && c[1] == 1 && c[2] == 2 && c[3] == 1 && c[4] == 1,
               "counting with repeats");
        expect(Math.abs(Klm.seconds(Klm.count("M P 2B")) - 2.65) < 1e-9,
               "one click");
        try {
            Klm.count("M X");
            expect(false, "unknown operator");
        } catch (IllegalArgumentException e) {
            passed++;
        }
        Map<String, int[]> d = Klm.readDesigns("data/orders.txt");
        expect(Math.abs(Klm.seconds(d.get("A: dropdown form")) - 25.54) < 1e-9,
               "design A");
        expect(Math.abs(Klm.seconds(d.get("B: menu tiles")) - 10.80) < 1e-9,
               "design B");
        System.out.println("All " + passed + " checks passed.");
    }
}
```

```
$ python3 -m unittest test_klm.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
$ java -cp out KlmTest
All 5 checks passed.
```

1.12 IMK dalam Praktik: Profitina

Profitina adalah platform *AI Visibility* (GEO, *generative engine optimization*) asal Indonesia yang beroperasi di app.profitina.com: ia mengukur seberapa siap situs web sebuah merek untuk dikutip oleh mesin jawab AI, menjelaskan setiap sinyal yang diukurnya, dan membantu memperbaiki apa yang ditemukannya [42]. Penggunanya kebanyakan pemilik merek dan pemasar usaha kecil, bukan insinyur, dan fakta tunggal itu menjadikannya masalah IMK sebelum menjadi masalah perangkat lunak.

Tiga tempat dalam produk ini memperlihatkan gagasan bab ini bekerja. Yang pertama adalah *skor*. Angka seperti “62” tidak berarti apa-apa bagi pemilik toko; karena itu dasbor menampilkan setiap sinyal dengan penjelasan sederhana dan tren mingguan, sehingga jurang evaluasi (“apakah situs saya membaik?”) dapat diseberangi sekilas pandang. Dasbor dirancang sebagai kokpit: satu layar mengumpulkan notifikasi, kanal yang diikuti, peringkat, dan aktivitas, dan setiap butirnya dapat diklik langsung ke bagiannya, yang memperpendek jurang eksekusi untuk langkah berikutnya yang paling umum. Yang kedua adalah *persetujuan*. Sesuai UU PDP, layanan ini meminta persetujuan eksplisit dan menyediakan ekspor data, penggantian email dan kata sandi, serta penghapusan akun secara penuh sebagai tindakan swalayan; masing-masing adalah antarmuka kecil yang harus dipahami, bukan sekadar diklik, persis standar yang dijelaskan di

Subbab 1.4.2. Yang ketiga adalah *waktu umpan balik* di linimasa komunitas: ketika seorang anggota mengirim tulisan, kotak tulisnya langsung kosong dan tulisannya segera muncul, dan jika pengiriman gagal, teksnya dikembalikan sehingga tidak ada ketikan yang hilang. Itulah prinsip umpan balik Norman yang diwujudkan menjadi fitur.

Produk ini juga mengikuti satu aturan yang akan diikuti setiap antarmuka bertenaga AI dalam buku ini: aplikasi memberi saran dan pengguna yang memutuskan. Content Engine menyusun draf teks dengan model yang berjalan secara lokal, dan ketika tidak ada model yang tersedia, aplikasi mengatakannya terus terang alih-alih diam-diam menampilkan templat. Bab 13 menjelaskan mengapa kejujuran semacam itu adalah keputusan desain, bukan detail rekayasa. Studi kasus praktik dalam buku ini bukan produk yang sesungguhnya, melainkan contoh pengajaran, *Profitina Laundry*, yang dipakai bab-bab selanjutnya untuk latihan yang memerlukan sebuah layanan kecil yang utuh.

1.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

1. **Merancang untuk diri sendiri.** Orang yang membangun sistem tahu cara kerjanya dan tidak dapat melihat apa yang dilihat pendatang baru. Amati setidaknya satu pengguna sungguhan sebelum meyakini bahwa sebuah desain sudah jelas (Bab 5 menunjukkan caranya).
2. **Menyalahkan pengguna.** “Nanti juga terbiasa” menyembunyikan jurang yang lebar. Tanyakan jurang mana yang terlalu lebar, lalu persempit.
3. **Mengatakan “mudah dipakai” tanpa menyebut bagi siapa.** *Usability* selalu untuk pengguna, tujuan, dan konteks penggunaan tertentu. Tuliskan ketiganya.
4. **Memercayai statistik tanpa bertanya apa yang dihitungnya.** StatCounter menghitung tayangan halaman di situs yang terlacak, bukan orang; survei menghitung responden yang menjawab. Baca metodenya sebelum merancang berdasarkan angka itu.
5. **Menganggap KLM sebagai kebenaran.** KLM memprediksi waktu ahli yang tidak berbuat salah. Pakailah untuk membandingkan desain sejak dini, lalu ujilah dengan orang sungguhan.
6. **Menaruh operator M di mana-mana, atau tidak sama sekali.** Keduanya memelesetkan prediksi. Taruh satu M sebelum setiap bongkahan yang memerlukan keputusan, mengikuti kaidah Kieras.
7. **Memasang perangkat lunak dari situs sembarangan.** Gunakan sumber resmi dan pengelola paket di Subbab 1.6 sampai 1.8, lalu periksa hasilnya dengan `check_setup.py` dan `CheckSetup.java`.
8. **Membuka satu berkas, bukan folder, di VS Code.** Terminal, Explorer, dan proyek Java semuanya bekerja dari folder yang dibuka.

1.14 Ringkasan Bab

- Interaksi manusia dan komputer adalah disiplin perancangan, evaluasi, dan implementasi sistem interaktif untuk digunakan manusia. *Usability* (ISO 9241-11) adalah efektivitas, efisiensi, dan kepuasan bagi pengguna, tujuan, dan konteks tertentu; pengalaman pengguna (ISO 9241-210) mencakup persepsi sebelum, selama, dan sesudah penggunaan; aksesibilitas memperluas *usability* ke rentang orang yang seluas-luasnya.
- Jurang eksekusi Norman (bagaimana caranya?) dan jurang evaluasi (berhasilkah?) menjelaskan sebagian besar masalah antarmuka; penanda, pemetaan, dan batasan mempersempit yang pertama, umpan balik dan model konseptual yang jelas mempersempit yang kedua.

- Bidang ini bergeser dari spesialis di baris perintah ke semua orang di ponsel, dan pada 2025–2026 ke Liquid Glass, Material 3 Expressive, komputasi spasial, dan agen AI. CHI 2026 menerima 1.702 makalah.
- Di Indonesia, 81,72% penduduk daring (APJII 2026), Android membawa 77,07% tayangan halaman seluler (StatCounter, Agustus 2026), dan 11,97% penduduk berusia enam puluh tahun ke atas (BPS, 2025). UU 8/2016, PP 42/2020, UU PDP beserta PP 33/2026, *European Accessibility Act*, dan WCAG 2.2 menjadikan antarmuka yang baik urusan hukum.
- Meja kerjanya adalah JDK 25 LTS, Python 3.14, dan VS Code dengan ekstensi Java dan Python, di Windows, macOS, atau Linux.
- *Keystroke-Level Model* memprediksi waktu tugas seorang ahli sebagai jumlah waktu operator (Persamaan (1.1)); untuk pesanan Bu Ratna, halaman ubin memerlukan 10,80 s alih-alih 25,54 s, atau 12,8 jam kerja sebulan pada 120 pesanan sehari.

1.15 Soal Latihan

1. Dengan Persamaan (1.1) dan **Tabel 1.3**, hitunglah dengan tangan waktu KLM pada desain A dan desain B untuk pesanan yang hanya berisi *satu* menu (satu es teh). Berapa penghematannya dalam detik dan dalam persen? Mengapa persentasenya lebih kecil daripada untuk pesanan tiga menu?
2. Seorang pelanggan memesan lima menu berbeda, masing-masing satu porsi. Tuliskan rangkaian operatornya untuk kedua desain, hitung waktunya, lalu periksa jawaban Anda dengan menambahkan desain-desain itu ke salinan `data/orders.txt` dan menjalankan `klm.py`.
3. Dimas mengusulkan desain ketiga C untuk Bu Ratna, yang pandai mengetik: setiap menu punya kode dua huruf (ET untuk es teh), dan pesanan diketik sebagai kode, jumlah, dan Enter per menu, dengan Enter pada baris kosong untuk menyimpan. Tuliskan desain C untuk pesanan di Subbab 1.11.2 dalam format berkas **Kode 1.5** (ingat bahwa tangan harus pindah ke keyboard terlebih dahulu), jalankan programnya, dan laporkan apa yang dicetaknya. Apakah C desain terbaik bagi Bu Ratna? Berikan satu argumen yang tidak dapat dilihat KLM.
4. Di kios puskesmas pada Subbab 1.1, seorang pasien berusia tujuh puluh tahun harus memilih satu dari dua belas tombol kecil yang hanya berisi teks, dan setelah menekannya, kios mencetak sebuah nomor tanpa menyebut untuk poli mana nomor itu. Sebutkan jurang eksekusi dan jurang evaluasi dalam situasi ini, lalu usulkan satu perubahan untuk masing-masing.
5. Tuliskan tiga sasaran *usability* yang terukur untuk halaman pesanan Bu Ratna, masing-masing satu untuk efektivitas, efisiensi, dan kepuasan, dalam pengertian ISO 9241-11. Setiap sasaran harus menyebut penggunanya, tugasnya, dan angka yang harus dicapai.
6. StatCounter melaporkan bahwa 60,59% tayangan halaman di Indonesia pada Agustus 2026 berasal dari komputer desktop. Seorang teman menyimpulkan: “Jadi layanan di Indonesia sebaiknya dirancang untuk desktop terlebih dahulu.” Jelaskan dalam tiga atau empat kalimat mengapa kesimpulan ini tidak sahih.
7. Kembangkan `klm.py` dan `Klm.java` sehingga argumen baris perintah ketiga yang opsional memberikan jumlah hari kerja per bulan sebagai pengganti angka tetap 26. Tunjukkan keluarannya untuk 120 pesanan sehari dan 30 hari kerja.

1.16 Penyelesaian Soal Latihan

1. Desain A untuk satu menu: M P B B P B B P B B H K H P B B, lalu M P B B untuk *Save*: $n_M = 2$, $n_P = 5$, $n_B = 10$, $n_H = 2$, $n_K = 1$, sehingga $T_A = 2.70 + 5.50 + 1.00 + 0.80 + 0.28 = 10.28$ s. Desain B: M P B B untuk ubin dan M P B B untuk *Save*: $T_B = 2.70 + 2.20 + 0.40 = 5.30$ s.

Penghematannya 4.98 s, atau 48,4%. Persentasenya lebih kecil karena *Save* penutup memakan M P B B yang sama pada kedua desain, dan dengan hanya satu menu, biaya bersama itu menjadi bagian yang lebih besar dari keseluruhan.

2. Desain A: lima kali bongkahan per menu ditambah *Save*: $n_M = 6$, $n_P = 21$, $n_B = 42$, $n_H = 10$, $n_K = 5$; $T_A = 8.10 + 23.10 + 4.20 + 4.00 + 1.40 = 40.80$ s. Desain B: lima kali M P B B ditambah *Save*: $n_M = 6, n_P = 6, n_B = 12$; $T_B = 8.10 + 6.60 + 1.20 = 15.90$ s. Penghematannya naik menjadi 24.90 s (61,0%). Dalam berkas, tulis `M P 2B P 2B P 2B H K H P 2B` lima kali di bawah desain A dan `M P 2B` lima kali di bawah desain B, masing-masing diikuti baris *Save*; program mencetak total yang sama.

3. Desain C dalam format berkas:

```
C: keyboard codes
  H                              # tangan dari mouse ke keyboard
  M 2K K K                       # "ET", jumlah 2, Enter
  M 2K K K                       # "NG", 1, Enter
  M 2K K K                       # "AG", 1, Enter
  M K                            # Enter pada baris kosong menyimpan
```

Setelah ditambahkan ke berkas, program mencetak:

```
Keystroke-Level Model, times in seconds (Card, Moran and Newell)
design               K   P   B   H   M   time
A: dropdown form     3  13  26   6   4  25.54
B: menu tiles        0   4  10   0   4  10.80
C: keyboard codes   13   0   0   1   4   9.44
C saves 16.10 s per order (63.0%).
At 120 orders a day: 32.2 minutes a day, 14.0 hours a month (26 days).
```

C adalah yang tercepat *bagi ahli yang hafal semua kodenya*, 1,36 s lebih cepat daripada ubin. KLM tidak dapat melihat biaya mempelajari dan mengingat empat puluh dua kode. Ubin mengandalkan *pengenalan* (*recognition*, menunya terlihat), sedangkan kode mengandalkan *pengingatan* (*recall*, menunya harus diingat), dan Bab 2 menjelaskan mengapa pengingatan lebih lambat dan lebih rawan salah bagi pendatang baru dan di bawah tekanan. Produk yang masuk akal menyediakan keduanya: ubin untuk semua orang dan kode sebagai jalan pintas bagi pengguna yang terlatih, yaitu heuristik Nielsen "fleksibilitas dan efisiensi penggunaan" di Bab 3.

4. Jurang eksekusi: pasien tidak dapat memetakan tujuan "berobat ke dokter gigi" ke salah satu dari dua belas label teks kecil, mungkin dengan penglihatan yang kurang baik. Perubahan: tombol yang lebih sedikit dan lebih besar dengan ikon dan kata-kata sederhana ("Gigi"), dikelompokkan menurut frekuensi, serta petugas atau suara pemandu untuk bantuan. Jurang evaluasi: setelah menekan tombol, tiket hanya menampilkan nomor, sehingga pasien tidak dapat memastikan bahwa poli yang dipilih benar. Perubahan: cetak dan tampilkan nama poli dalam huruf besar di sebelah nomornya, dan izinkan pasien membatalkan dalam beberapa detik.

5. Contohnya. Efektivitas: "Bu Ratna memasukkan 95% pesanan tiga menu tanpa koreksi, diukur selama satu jam makan siang hari Sabtu." Efisiensi: "Bu Ratna, setelah satu minggu pemakaian, memasukkan pesanan tiga menu dalam rata-rata paling lama 15 detik." Kepuasan: "Bu Ratna dan pembantunya memberi skor paling sedikit 70 pada *System Usability Scale* (Bab 7) setelah dua minggu." Setiap sasaran menyebut pengguna, tugas, dan targetnya.

6. Angka itu menghitung tayangan halaman di situs web yang memasang kode StatCounter, bukan orang dan bukan waktu. Segelintir pengguna desktop yang intens di kantor dapat menghasilkan banyak tayangan halaman, sementara sebagian besar orang Indonesia memakai internet di ponsel di dalam aplikasi seperti WhatsApp dan lokapasar, yang tidak terlihat oleh penghitung web. Survei APJII terhadap orang, dan fakta bahwa orang yang masih luring kebanyakan tidak punya

perangkat dan kurang literasi digital, menunjuk ponsel sebagai perangkat utama. Kesimpulan yang tepat adalah merancang *mobile-first* dan memeriksa pengguna nyata dari layanan tertentu itu.

7. Di Python, tambahkan setelah `per_day`: `days = int(sys.argv[3]) if len(sys.argv) > 3 else 26` lalu ganti `26` pada `print` terakhir dengan `days`, baik dalam perhitungan maupun dalam teks `({days} days)`. Di Java, tambahkan `int days = args.length > 2 ? Integer.parseInt(args[2]) : 26;` dan pakai `days` di dua tempat yang sama. Untuk `python3 klm.py data/orders.txt 120 30`, baris terakhirnya menjadi `At 120 orders a day: 29.5 minutes a day, 14.7 hours a month (30 days).`, karena $1768.8 \times 30/3600 = 14.74$ jam.

Daftar Pustaka

[1] T. T. Hewett, R. Baecker, S. Card, T. Carey, J. Gasen, M. Mantei, G. Perlman, G. Strong, and W. Verplank, *ACM SIGCHI Curricula for Human-Computer Interaction*. New York, NY, USA: ACM, 1992. [Online]. Available: https://dl.acm.org/doi/book/10.1145/2594128

[2] International Organization for Standardization, *ISO 9241-11:2018 Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts*. Geneva, Switzerland: ISO, 2018. [Online]. Available: https://www.iso.org/standard/63500.html

[3] International Organization for Standardization, *ISO 9241-210:2019 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. Geneva, Switzerland: ISO, 2019. [Online]. Available: https://www.iso.org/standard/77520.html

[4] D. A. Norman, *The Design of Everyday Things*, revised and expanded ed. New York, NY, USA: Basic Books, 2013.

[5] J. Grudin, *From Tool to Partner: The Evolution of Human-Computer Interaction*. San Rafael, CA, USA: Morgan & Claypool, 2017.

[6] Apple Inc., “Apple introduces a delightful and elegant new software design,” Apple Newsroom, Jun. 9, 2025. [Online]. Available: https://www.apple.com/newsroom/2025/06/apple-introduces-a-delightful-and-elegant-new-software-design/

[7] Google Design, “Expressive Design: Google’s UX Research,” accessed Sep. 2026. [Online]. Available: https://design.google/library/expressive-material-design-google-research

[8] Microsoft, “Fluent 2 Design System,” accessed Sep. 2026. [Online]. Available: https://fluent2.microsoft.design/

[9] Apple Inc., “Apple Vision Pro available in the U.S. on February 2,” Apple Newsroom, Jan. 8, 2024. [Online]. Available: https://www.apple.com/newsroom/2024/01/apple-vision-pro-available-in-the-us-on-february-2/

[10] TechCrunch, “ChatGPT reaches 900M weekly active users,” Feb. 27, 2026. [Online]. Available: https://techcrunch.com/2026/02/27/chatgpt-reaches-900m-weekly-active-users

[11] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, and E. Horvitz, “Guidelines for Human-AI Interaction,” in *Proc. CHI Conference on Human Factors in Computing Systems (CHI ’19)*, Glasgow, UK, 2019, paper 3. doi: 10.1145/3290605.3300233

[12] Interactive Media Lab Dresden, “ACM CHI 2026 in Barcelona,” Apr. 2026. [Online]. Available: https://imld.de/en/news/2026/04/acm-chi-2026-in-barcelona/

[13] Figma, “What’s new from Config 2026,” Figma Help Center, Jun. 2026. [Online]. Available: https://help.figma.com/hc/en-us/articles/39582753756695-What-s-new-from-Config-2026

[14] detikInet, “Pengguna Internet Indonesia Tembus 235 Juta, Penetrasi 81,72%,” May 2026, reporting Asosiasi Penyelenggara Jasa Internet Indonesia (APJII), *Survei Penetrasi Internet Indonesia 2026*. [Online]. Available: https://inet.detik.com/cyberlife/d-8499661/pengguna-internet-indonesia-tembus-235-juta-penetrasi-81-72

[15] StatCounter Global Stats, "Mobile Operating System Market Share Indonesia," August 2026. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/indonesia

[16] StatCounter Global Stats, "Desktop vs Mobile vs Tablet Market Share Indonesia," August 2026. [Online]. Available: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/indonesia

[17] Periskop.id, "Jumlah Lansia Indonesia 2025 11,97%, Indonesia Masuk Aging Population," May 5, 2026, reporting Badan Pusat Statistik, *Survei Penduduk Antar Sensus (SUPAS) 2025*. [Online]. Available: https://periskop.id/kependudukan/20260505/jumlah-lansia-indonesia-2025-1197-indonesia-masuk-aging-population

[18] Republik Indonesia, *Undang-Undang Nomor 8 Tahun 2016 tentang Penyandang Disabilitas*. JDIH BPK RI. [Online]. Available: https://peraturan.bpk.go.id/Details/37251/uu-no-8-tahun-2016

[19] Republik Indonesia, *Peraturan Pemerintah Nomor 42 Tahun 2020 tentang Aksesibilitas terhadap Permukiman, Pelayanan Publik, dan Pelindungan dari Bencana bagi Penyandang Disabilitas*. JDIH BPK RI. [Online]. Available: https://peraturan.bpk.go.id/Details/142301/pp-no-42-tahun-2020

[20] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. JDIH BPK RI. [Online]. Available: https://peraturan.bpk.go.id/Details/229798/uu-no-27-tahun-2022

[21] Veritask, "Pengaturan Teknis Pelindungan Data Pribadi dan Kewajiban Pengendali serta Prosesor Akhirnya Terbit Lewat PP 33/2026," 2026, on *Peraturan Pemerintah Nomor 33 Tahun 2026 tentang Peraturan Pelaksanaan Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. [Online]. Available: https://veritask.ai/id/artikel/pengaturan-teknis-pelindungan-data-pribadi-dan-kewajiban-pengendali-serta-prosesor-akhirnya-terbit-lewat-pp-33-2026

[22] European Parliament and Council of the European Union, "Directive (EU) 2019/882 on the accessibility requirements for products and services (European Accessibility Act)," *Official Journal of the European Union*, L 151, Jun. 7, 2019, applicable from Jun. 28, 2025. [Online]. Available: https://eur-lex.europa.eu/eli/dir/2019/882/oj

[23] World Wide Web Consortium, *Web Content Accessibility Guidelines (WCAG) 2.2*, W3C Recommendation. [Online]. Available: https://www.w3.org/TR/WCAG22/

[24] W3C Web Accessibility Initiative, "For Review: WCAG 3 Working Draft – September 2026," Sep. 10, 2026, and "WCAG 3 Introduction." [Online]. Available: https://www.w3.org/WAI/news/2026-09-10/wcag3/

[25] Oracle, "Java Downloads: JDK 25 (LTS) and JDK 27," accessed Sep. 2026. [Online]. Available: https://www.oracle.com/java/technologies/downloads/

[26] OpenJDK, "JDK 25," General Availability Sep. 16, 2025. [Online]. Available: https://openjdk.org/projects/jdk/25/

[27] Python Software Foundation, "Python 3.14.7," Aug. 5, 2026. [Online]. Available: https://www.python.org/downloads/release/python-3147/

[28] Microsoft, "Visual Studio Code 1.139," release notes, Sep. 23, 2026. [Online]. Available: https://code.visualstudio.com/updates

[29] Microsoft, "Java in Visual Studio Code," *VS Code Documentation*. [Online]. Available: https://code.visualstudio.com/docs/languages/java

[30] Microsoft, "Python in Visual Studio Code," *VS Code Documentation*. [Online]. Available: https://code.visualstudio.com/docs/languages/python

[31] Inside.java, "The Arrival of Java 27!," Sep. 15, 2026. [Online]. Available: https://inside.java/2026/09/15/jdk-27-available/

[32] Adoptium, "Eclipse Temurin 8u504, 11.0.32.1, 17.0.20.1, 21.0.12.1, 25.0.4.1 and 26.0.2.1 Available," Sep. 2, 2026. [Online]. Available: https://adoptium.net/news/2026/09/eclipse-temurin-8u504-110321-170201-210121-25041-26021-available

[33] Python Software Foundation, "What's New in Python 3.14," *Python 3.14 Documentation*. [Online]. Available: https://docs.python.org/3.14/whatsnew/3.14.html

[34] OpenJDK, "JEP 330: Launch Single-File Source-Code Programs," JDK 11. [Online]. Available: https://openjdk.org/jeps/330

[35] Python Software Foundation, "Using Python on Windows," *Python 3.14 Documentation*. [Online]. Available: https://docs.python.org/3.14/using/windows.html

[36] Phoronix, "Ubuntu 26.04 LTS Moving To OpenJDK 25 By Default," 2026. [Online]. Available: https://www.phoronix.com/news/Ubuntu-26.04-OpenJDK-25

[37] Python Software Foundation, "venv — Creation of virtual environments," *The Python Standard Library*. [Online]. Available: https://docs.python.org/3/library/venv.html

[38] OpenJDK, "JEP 512: Compact Source Files and Instance Main Methods," JDK 25. [Online]. Available: https://openjdk.org/jeps/512

[39] S. K. Card, T. P. Moran, and A. Newell, "The Keystroke-Level Model for User Performance Time with Interactive Systems," *Communications of the ACM*, vol. 23, no. 7, pp. 396–410, 1980. doi: 10.1145/358886.358895

[40] S. K. Card, T. P. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*. Hillsdale, NJ, USA: Lawrence Erlbaum, 1983.

[41] D. Kieras, "Using the Keystroke-Level Model to Estimate Execution Times," tutorial notes, University of Michigan, Ann Arbor, MI, USA, 2001.

[42] Profitina, "Profitina — AI Visibility (GEO) platform," accessed Sep. 2026. [Online]. Available: https://profitina.com

Bab 2

Faktor Manusia: Persepsi, Kognisi, Memori, Hukum Fitts dan Hick

Sebuah antarmuka dipakai oleh tubuh dan pikiran yang punya batas, dan batas itu tidak berubah ketika perangkat lunaknya diperbarui: mata yang hanya melihat tajam di satu titik kecil, perhatian yang mengikuti satu hal pada satu waktu, memori kerja yang menampung segelintir butir selama beberapa detik, dan tangan yang butuh waktu untuk mencapai sasaran. Bab ini menjelaskan batas-batas itu dengan kata-kata sederhana, menyajikan bukti di baliknya, dan mengubah dua hasil paling andal dari psikologi eksperimental, hukum Fitts dan hukum Hick, menjadi sebuah kalkulator. Di akhir bab ini Anda akan mampu menggambarkan manusia sebagai pemroses informasi dengan bagian persepsi, kognitif, dan motorik; menerapkan prinsip Gestalt, batas perhatian, dan batas memori kerja pada sebuah layar; menjelaskan bagaimana penglihatan, memori, dan gerak berubah seiring usia; menghitung indeks kesulitan dan waktu gerak sebuah sasaran dengan hukum Fitts serta waktu keputusan sebuah menu dengan hukum Hick; mencocokkan kedua hukum dengan pengukuran Anda sendiri memakai metode kuadrat terkecil; dan memakai hasilnya untuk membandingkan dua tata letak kios antrean puskesmas dalam Java dan Python.

2.1 Bu Siti di Kios Puskesmas

Puskesmas Kenanga, sebuah pusat kesehatan masyarakat di pinggiran Sidoarjo, memasang kios antrean layar sentuh di ruang tunggunya pada 2025. Pasien menyentuh nama poli yang dituju, lalu kios mencetak tiket bernomor. Di layar pertama ada dua belas tombol, satu per layanan, tersusun rapi dalam kisi tiga kolom dan empat baris di bagian atas layar, masing-masing berlabel huruf kecil: Umum, Gigi, KIA, Lansia, Anak, Laboratorium, TB, Gizi, Imunisasi, Jiwa, Sanitasi, Farmasi. (Program di bab ini memakai label bahasa Inggris yang setara, seperti *General* dan *Elderly*.)

Bu Siti berusia enam puluh delapan tahun. Ia datang untuk pemeriksaan tekanan darah bulanan di poli lansia, seperti yang sudah dijalaninya selama dua tahun. Ia berdiri lama di depan kios, mencondongkan badan ke depan lalu ke belakang, karena hurufnya terlalu kecil untuk kacamata jauhnya dan terlalu jauh untuk kacamata bacanya. Ia menemukan *Lansia* di baris kedua, menjulurkan tangan, lalu menyentuh tombol di sebelahnya, *Anak*, karena jarinya menutupi label yang ditujunya dan tombol-tombolnya berdekatan. Tiketnya bertuliskan “A-017, Anak”. Ia tidak menyadarinya sampai perawat memanggilnya ke ruangan yang salah empat puluh menit kemudian.

Penjelasan pertama dari petugas adalah bahwa pasien lansia “belum terbiasa dengan teknologi”. Pemasok kios menawarkan video pelatihan. Namun kepala puskesmas, dr. Rina, mengajukan pertanyaan lain: apa tepatnya, pada tubuh dan pikiran orang seperti Bu Siti, yang dituntut kios itu secara berlebihan? Matanya harus membaca teks kecil pada jarak yang canggung. Perhatiannya harus menyisir dua belas label yang tampak serupa. Memorinya harus menahan kata yang dicarinya sambil membaca kata-kata lain. Tangannya harus mengenai sasaran setinggi dua puluh

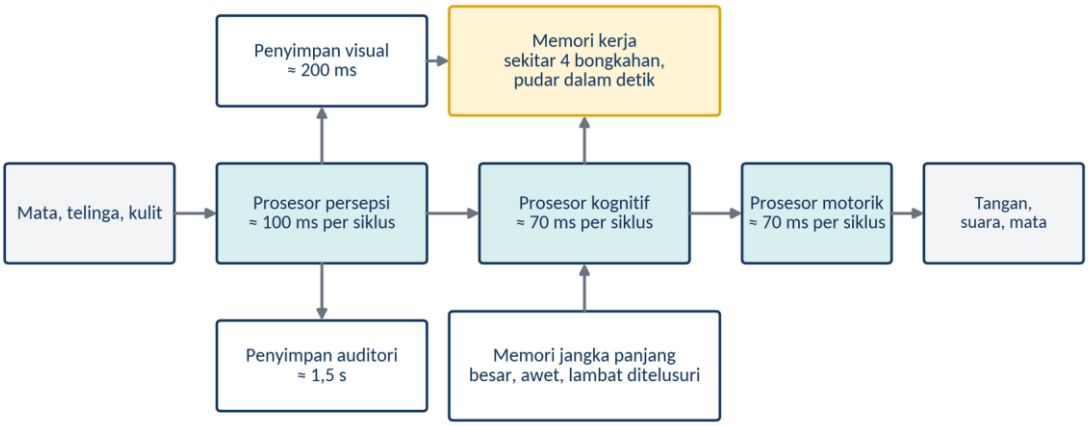
milimeter sejauh jangkauan lengan. Dan kios tidak memberinya kesempatan kedua untuk menyadari kekeliruannya.

Masing-masing hal itu sudah diukur para psikolog selama puluhan tahun, dan beberapa di antaranya mengikuti hukum yang cukup presisi untuk dihitung. Bab ini menelusuri pertanyaan dr. Rina melalui persepsi, perhatian, memori, dan gerak, lalu diakhiri dengan menghitung, untuk kios lama dan kios yang dirancang ulang, berapa lama seorang pasien memerlukan waktu untuk memilih dan mencapai tombolnya.

Bu Siti bukan pengecualian. Menurut Badan Pusat Statistik, 11,97% penduduk Indonesia berusia enam puluh tahun ke atas pada 2025 [1], dan poli lansia di puskesmas ada justru karena mereka termasuk pengunjung yang paling sering datang. Kios yang gagal melayani mereka gagal melayani pengguna tetapnya sendiri.

2.2 Gagasannya dengan Kata-Kata Sederhana: Manusia sebagai Pemroses Informasi

Untuk menalar apa yang dituntut sebuah layar dari seseorang, IMK meminjam gambaran sederhana dari psikologi kognitif: manusia sebagai pemroses informasi dengan tiga tahap. Card, Moran, dan Newell menggambarkannya pada 1983 sebagai *Model Human Processor* [2]; **Gambar 2.1** memperlihatkannya. Informasi masuk melalui indra dan ditahan sangat singkat di penyimpan sensorik. *Prosesor persepsi* mengubahnya menjadi sesuatu yang dikenali, misalnya "tombol berlabel Lansia". *Prosesor kognitif* memutuskan apa yang dilakukan, dengan memakai *memori kerja* yang kecil dan cepat serta *memori jangka panjang* yang besar dan lambat. *Prosesor motorik* menggerakkan tangan, mata, atau suara.



Gambar 2.1 Model Human Processor: prosesor persepsi, kognitif, dan motorik dengan waktu siklus yang khas, penyimpan sensorik yang singkat, dan dua jenis memori.

Waktu siklus pada gambar adalah nilai khas dari Card, Moran, dan Newell; masing-masing punya rentang yang lebar antarorang dan antartugas (siklus persepsi, misalnya, sekitar 50 hingga 200 ms) [2]. Ada dua akibat yang patut diingat. Pertama, reaksi yang paling sederhana pun memakan beberapa ratus milidetik, karena melewati ketiga prosesor setidaknya sekali; antarmuka yang menanggapi sentuhan dalam sekitar sepersepuluh detik terasa seketika, sedangkan yang memerlukan satu detik terasa lambat, karena siklus orang itu sendiri jauh lebih cepat. Kedua, setiap tahap punya *kapasitas*: masing-masing dapat kelebihan beban, dan ketika itu terjadi, tahap

lain menunggu. Kebanyakan antarmuka yang buruk membebani satu tahap secara berlebihan sementara tahap lain menganggur.

Model ini adalah penyederhanaan, dan buku-buku psikologi rekayasa modern menggambarkan model yang lebih kaya dengan sumber daya paralel dan umpan balik [3]. Bagi perancang, tiga tahap itu tetap menjadi daftar periksa yang baik: *Bisakah mereka melihatnya? Bisakah mereka memutuskan? Bisakah mereka menjangkaunya?*

2.3 Persepsi: Apa yang Sampai ke Pikiran

2.3.1 Penglihatan

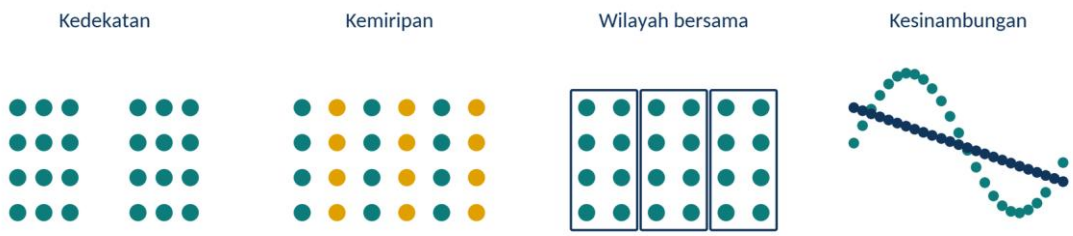
Mata hanya melihat tajam di *fovea*, daerah kecil di tengah retina yang mencakup sekitar dua derajat bidang pandang, kira-kira selebar kuku ibu jari pada jarak sepanjang lengan. Selebihnya dilihat dengan *penglihatan tepi* (*peripheral vision*), yang pandai menangkap gerakan dan bentuk besar tetapi buruk untuk membaca. Kita merasa melihat seluruh layar sekaligus karena mata melompat beberapa kali per detik dari satu titik ke titik lain, dalam gerakan yang disebut *sakade*, dan otak merangkai hasilnya [3]. Bagi desain, ini berarti teks hanya terbaca di tempat mata mendarat, dan pesan yang diletakkan jauh dari tempat orang memandang, misalnya galat di bagian atas formulir panjang sementara pengguna berada di bagian bawah, kemungkinan besar terlewat.

Penglihatan berubah seiring usia dengan cara yang berpengaruh pada setiap layar. Sejak usia empat puluhan, lensa mata mengeras dan tidak lagi dapat memfokus benda dekat, kondisi yang disebut *presbiopia*, dan itulah sebabnya Bu Siti punya kacamata baca. Lensa juga menguning dan meloloskan lebih sedikit cahaya, sehingga mata yang menua memerlukan lebih banyak cahaya dan kontras, serta lebih mudah silau [4]. Teks kecil, kontras rendah, huruf tipis, dan layar mengilap di ruang tunggu yang terang semuanya pertama-tama merugikan pengguna lanjut usia. Panduan Czaja dan rekan-rekannya tentang perancangan untuk orang lanjut usia menghimpun temuan-temuan ini menjadi aturan praktis, seperti ukuran teks yang lega, kontras yang kuat, dan sasaran yang besar [4].

Penglihatan warna berbeda antarorang. Sekitar 1 dari 12 laki-laki dan 1 dari 200 perempuan mengalami suatu bentuk defisiensi penglihatan warna, paling sering berkurangnya kemampuan membedakan merah dari hijau [5]. Layar antrean yang menampilkan nomor yang dipanggil dengan warna hijau dan nomor yang menunggu dengan warna merah, dengan demikian, tidak terbaca oleh sebagian pasien yang cukup berarti. Aturannya sederhana: jangan pernah menjadikan warna *satu-satunya* pembawa makna; tambahkan kata, bentuk, atau posisi. WCAG 2.2 menjadikannya kriteria keberhasilan (1.4.1, "*Use of Color*") [6], dan Bab 11 menambahkan rasio kontras yang dapat dihitung untuk setiap pasangan warna.

2.3.2 Pengelompokan: prinsip Gestalt

Sebelum kita membaca satu label pun, persepsi sudah mengelompokkan apa yang kita lihat. Pada awal abad kedua puluh, para psikolog Gestalt menggambarkan aturan pengelompokan ini, dan satu abad riset sesudahnya menegaskan serta memperhalusnya [7]. **Gambar 2.2** memperlihatkan empat prinsip yang paling penting untuk layar.



Gambar 2.2 Empat prinsip Gestalt: kedekatan (dua kelompok karena jarak), kemiripan (kolom karena warna), wilayah bersama (pasangan karena bingkai), dan kesinambungan (dua garis mulus, bukan zig-zag).

Kedekatan (*proximity*): butir yang berdekatan terlihat sebagai satu kelompok, sehingga dua belas titik di panel kiri membentuk dua kelompok berisi enam karena satu celah yang lebih lebar. *Kemiripan* (*similarity*): butir yang tampak serupa dikelompokkan, sehingga panel kedua terbaca sebagai kolom-kolom *teal* dan kuning. *Wilayah bersama* (*common region*): butir di dalam bingkai yang sama dianggap satu kesatuan, apa pun warna atau jaraknya. *Kesinambungan* (*continuity*): mata mengikuti garis yang mulus, sehingga panel terakhir menampilkan sebuah gelombang dan sebuah garis lurus yang bersilangan, bukan dua zig-zag yang bertemu.

Kios lama melanggar keempatnya dengan cara yang sama: dua belas tombol identik dengan jarak identik mengatakan, secara perseptual, bahwa kedua belas layanan itu sejenis dan sama pentingnya. Tata letak yang memisahkan enam layanan tersering dari sisanya dengan jarak, ukuran, atau bingkai membiarkan persepsi mengerjakan separuh pertama pencarian sebelum pasien membaca satu kata pun. Prinsip yang sama menjelaskan mengapa formulir yang labelnya lebih dekat ke kolom isian *di bawahnya* daripada ke kolom miliknya sendiri menimbulkan isian yang keliru: kedekatan mengalahkan niat.

2.3.3 Apa yang menonjol: fitur praatensi

Sebagian perbedaan visual tertangkap hampir seketika, berapa pun banyaknya butir lain: satu butir merah di antara butir abu-abu, satu butir besar di antara yang kecil, satu butir bergerak di antara yang diam. Teori integrasi fitur dari Treisman dan Gelade menjelaskan sebabnya: fitur sederhana seperti warna, ukuran, orientasi, dan gerak dicatat secara paralel di seluruh bidang pandang, sedangkan *kombinasi* fitur harus diperiksa butir demi butir [8]. Pelajarannya bagi perancang: satu elemen yang disorot (aksi utama, sebuah galat) dapat ditemukan seketika, tetapi sorotan yang dipakai di mana-mana tidak menyorot apa pun. Material 3 Expressive, revisi bahasa desain Google pada 2025, mengandalkan efek ini: Google melaporkan bahwa orang menemukan elemen kunci hingga empat kali lebih cepat pada tata letak ekspresif yang memakai ukuran, bentuk, dan warna untuk menonjolkan aksi utama, dan bahwa kesenjangan kinerja antara partisipan tua dan muda menyempit [9].

2.3.4 Pendengaran dan sentuhan

Pendengaran adalah saluran untuk peringatan dan bagi orang yang tidak dapat melihat layar; penyimpan sensoriknya menahan bunyi selama satu atau dua detik, lebih lama daripada penglihatan, dan itulah sebabnya nomor tiket yang diucapkan masih dapat diingat sesaat setelah disebutkan [2]. Pendengaran juga saluran yang gagal di ruang tunggu yang bising dan bagi banyak orang lanjut usia yang mengalami penurunan pendengaran, sehingga nomor yang dipanggil harus ditampilkan sekaligus diumumkan. Sentuhan membawa umpan balik tombol fisik, yang tidak dimiliki layar sentuh yang datar: getaran singkat atau keadaan tertekan yang terlihat menggantikan bunyi klik yang dulu dirasakan jari.

2.4 Perhatian: Satu Hal pada Satu Waktu

Perhatian adalah pemilihan apa yang diproses lebih lanjut. Perhatian dapat diarahkan dengan sengaja (mencari *Lansia*) atau direnggut oleh dunia (iklan berkedip di layar kios). Perhatian terbatas: kita dapat memusatkan diri pada satu tugas berat pada satu waktu, dan berpindah antartugas memakan waktu serta menambah kesalahan [3].

Demonstrasi paling mencolok atas batas ini adalah *inattentional blindness* (kebutaan karena tidak memperhatikan). Dalam eksperimen terkenal Simons dan Chabris, penonton yang menghitung operan bola basket dalam sebuah video sangat sering tidak melihat seseorang berkostum gorila yang berjalan melintasi adegan dan menepuk dadanya [10]. Apa yang tidak diperhatikan orang sering sama sekali tidak terlihat, bahkan ketika berada tepat di depan mata. Dalam antarmuka, hal ini menjelaskan mengapa spanduk peringatan di samping sebuah tugas terlewat ketika pengguna berkonsentrasi pada tugasnya, dan mengapa pesan konfirmasi yang muncul di tempat yang tidak sedang dilihat pengguna tidak mengonfirmasi apa pun. Bu Siti, yang berkonsentrasi mencari sebuah kata, tidak membaca tiket yang bertuliskan *Anak*.

Tiga aturan desain mengikutinya. Letakkan umpan balik di tempat perhatian sudah berada: di sebelah tombol yang baru ditekan, atau pada benda yang sedang dilihat pengguna. Jangan berebut perhatian tanpa alasan: setiap animasi, lencana, dan spanduk mengambil dari anggaran kecil yang sama. Dan ketika sesuatu benar-benar memerlukan perhatian, seperti pilihan keliru yang akan menghabiskan satu jam menunggu, interupsilah dengan sengaja dan bertanyalah, karena pesan yang pasif akan terlewat.

2.5 Memori: Apa yang Bertahan, dan Berapa Lama

2.5.1 Memori kerja

Memori kerja adalah tempat kita menahan apa yang sedang kita pikirkan: nama poli yang kita cari, empat angka kata sandi sekali pakai, menu yang baru saja kita putuskan untuk dipesan. Pada 1956 George Miller dengan terkenal mengusulkan bahwa rentangnya adalah “tujuh, plus atau minus dua” butir [11]. Riset sesudahnya menunjukkan bahwa angka Miller itu terlalu besar karena orang pandai mengelompokkan butir, dan bahwa batas yang sebenarnya lebih dekat ke sekitar empat satuan bermakna, atau *bongkahan* (*chunk*) [12]. Informasi di memori kerja juga pudar dalam hitungan detik kecuali diulang-ulang, dan mudah tergeser oleh informasi baru.

Chunking (pembongkahan) adalah jalan keluar dari batas ini. Angka 081234567890 adalah dua belas butir, terlalu banyak; ditulis sebagai 0812-3456-7890 ia menjadi tiga bongkahan, dan “0812” dikenali sebagai awalan nomor seluler yang akrab, sehingga ia menjadi satu bongkahan yang hampir tanpa biaya. Karena itulah nomor telepon, nomor rekening bank, dan nomor *virtual account* untuk pembayaran di Indonesia sebaiknya selalu ditampilkan berkelompok, dan kata sandi sekali pakai enam angka lebih mudah disalin sebagai “482 913” daripada “482913”.

2.5.2 Mengenali, bukan mengingat

Memori jangka panjang besar dan awet, tetapi mengambil sesuatu darinya tanpa petunjuk, yaitu *recall* (mengingat), lambat dan rawan salah. Mengenali sesuatu yang diperlihatkan, yaitu *recognition* (mengenali), jauh lebih mudah: kebanyakan orang tidak dapat menyebutkan kedua belas layanan puskesmasnya, tetapi mereka langsung mengenali *Lansia* ketika melihatnya. Norman menyebut ini perbedaan antara *pengetahuan di kepala* dan *pengetahuan di dunia*: antarmuka yang baik menaruh pengetahuan yang diperlukan pengguna di dunia, di layar,

sehingga tidak perlu ada di kepala [13]. Menu, opsi yang terlihat, butir yang baru dipakai, dan pelengkapan otomatis semuanya mengubah pengingatan menjadi pengenalan. Kode keyboard pada Soal Latihan 1.3 adalah kebalikannya: cepat bagi ahli, tetapi menuntut pengingatan. Bab 3 bertemu gagasan ini lagi sebagai heuristik Nielsen "*recognition rather than recall*".

2.5.3 Model mental

Orang juga membawa *model mental*: penjelasan sederhana versi mereka sendiri tentang cara kerja sebuah sistem. Pasien yang mengira kios "memanggil dokter" mungkin menekan tombol kedua untuk memastikan, dengan harapan membatalkan yang pertama; sistem yang justru menerbitkan tiket kedua berbenturan dengan model itu. Perancang tidak dapat memasang model mental secara langsung, tetapi mereka membentuknya melalui apa yang diperlihatkan antarmuka. Antrean yang terlihat ("Anda nomor 17 untuk Poli Lansia; 6 orang sebelum Anda") membangun model yang benar; nomor tiket polos membangun model yang samar.

> **WAWASAN KUNCI:** Rancanglah untuk sekitar empat bongkahan, bukan tujuh. Jika sebuah tugas menuntut pengguna membawa lebih dari beberapa butir dari satu layar ke layar berikutnya, desainnyalah yang harus membawanya: tampilkan, kelompokkan, atau isikan.

2.6 Hukum Fitts: Berapa Lama Mencapai Sasaran

2.6.1 Hukumnya

Pada 1954 psikolog Paul Fitts mengukur berapa lama orang memerlukan waktu untuk mengetuk bergantian di antara dua pelat logam dengan berbagai lebar yang diletakkan pada berbagai jarak [14]. Ia menemukan bahwa waktunya tidak bergantung pada jarak saja maupun lebar saja, melainkan pada *rasio* keduanya. Sasaran yang dua kali lebih jauh sama cepatnya untuk dikenai seperti sebelumnya jika ia juga dua kali lebih lebar. Scott MacKenzie kemudian menyesuaikan hukum ini untuk penunjukan di komputer dalam bentuk yang kini dipakai di IMK dan di standar internasional untuk mengevaluasi perangkat penunjuk, ISO 9241-411 [15], [16].

> **DEFINISI.** *Hukum Fitts* menyatakan bahwa waktu untuk bergerak ke sasaran selebar W pada jarak D dari titik awal (keduanya dalam satuan yang sama) tumbuh secara linear terhadap *indeks kesulitan* ID sasaran itu, yang hanya bergantung pada rasio D/W. Kedua besaran didefinisikan oleh Persamaan (2.1) dan Persamaan (2.2).

Indeks kesulitannya adalah

$$ID = \log_2\left(\frac{D}{W} + 1\right) \tag{2.1}$$

dalam satuan bit, dan *waktu gerak*-nya adalah

$$MT = a + b \cdot ID \tag{2.2}$$

dengan a (detik) dan b (detik per bit) sebagai konstanta yang bergantung pada orang, perangkat, dan posturnya, dan ditemukan melalui pengukuran.

Dengan kata lain, Persamaan (2.1) menyatakan bahwa kesulitan tumbuh sebanding dengan logaritma berapa kali lebar sasaran yang harus ditempuh tangan. Logaritma inilah yang membuat hukum ini sangat berguna: menggandakan jarak tidak menggandakan waktu, melainkan menambah jumlah tetap b, karena $\log_2(2x) = \log_2 x + 1$ ketika D/W besar. Konstanta a adalah waktu yang diperlukan bahkan untuk sasaran yang sangat dekat, untuk memulai dan berhenti.

Persamaan (2.2) adalah garis lurus dalam ID, dan karena itulah kedua konstanta dapat ditemukan dengan mencocokkan garis pada titik-titik hasil pengukuran, seperti dilakukan di Subbab 2.8.

Kebalikan dari kemiringan garis adalah *throughput*, laju sistem orang-dan-perangkat dalam menyalurkan informasi:

$$TP = \frac{1}{b} \tag{2.3}$$

dalam bit per detik. *Throughput* yang lebih tinggi berarti sistem penunjuk yang lebih cepat. (ISO 9241-411 mendefinisikan *throughput* lebih cermat, sebagai rata-rata ID_e/MT dengan lebar *efektif* yang dihitung dari sebaran ketukan yang sebenarnya [16], [17]; Persamaan (2.3) adalah versi sederhananya, yang cukup untuk membandingkan tata letak.)

2.6.2 Contoh yang dikerjakan

Ambil konstanta latihan yang diperoleh Subbab 2.9 dari log ketukan: $a = 0.156$ s dan $b = 0.147$ s/bit. Pada kios lama, tombol *General* (Umum) lebarnya 60 mm dan tingginya 20 mm dengan titik tengah di (45,225) mm, diukur dari sudut kiri bawah layar, dan tangan pasien diam di birai di bawah layar pada $(110, -40)$ mm. Jaraknya adalah

$$D = \sqrt{(45-110)^2 + (225-(-40))^2} = \sqrt{4225 + 70225} = \sqrt{74450} \\ = 272.9 \text{ mm}.$$

Untuk W kita ambil sisi tombol yang *lebih pendek*, 20 mm, karena jari mendekat dari bawah dan harus berhenti di dalam tinggi tombol; ini penyederhanaan yang umum dan sedikit pesimistis untuk sasaran persegi panjang [15]. Maka

$$ID = \log_2\left(\frac{272.9}{20} + 1\right) = \log_2 14.645 = 3.872 \text{ bits},$$

$$MT = 0.156 + 0.147 \times 3.872 = 0.725 \text{ s}.$$

Pada kios yang dirancang ulang, tombol *General* berukuran 95 kali 60 mm dengan titik tengah di (57.5,200): $D = 245.7$ mm, $W = 60$ mm, $ID = \log_2(245.7/60 + 1) = 2.349$ bit, dan $MT = 0.156 + 0.147 \times 2.349 = 0.501$ s. Tombol yang lebih besar menghemat sekitar seperempat detik gerak, dan, yang lebih penting bagi Bu Siti, tingginya tiga kali lipat, sehingga jari yang mendarat sedikit meleset tetap mengenainya.

2.6.3 Arti hukum ini bagi desain

Hukum Fitts menjadi lima aturan sehari-hari. *Buat sasaran yang sering dipakai berukuran besar. Letakkan sasaran yang sering dipakai dekat dengan posisi tangan atau penunjuk saat itu. Manfaatkan tepi dan sudut layar ketika memakai mouse*, karena penunjuk berhenti di sana dan sasarannya secara efektif sedalam tak hingga. *Buat aksi berbahaya kecil dan jauh dari aksi yang sering dipakai*, pemakaian terbalik dari hukum yang sama. Dan *jangan membuat sasaran lebih kecil daripada yang dapat dikenai orang dengan andal*. Pedoman platform dan WCAG 2.2 mengubah aturan terakhir itu menjadi angka, yang dihimpun dalam **Tabel 2.1**.

Tabel 2.1 Ukuran minimum sasaran sentuh dalam pedoman terkini.

Sumber	Ukuran sasaran minimum	Catatan
WCAG 2.2, kriteria 2.5.8 *Target Size (Minimum)*, tingkat AA	24 kali 24 piksel CSS	atau ruang yang cukup di sekitar sasaran yang lebih kecil [6]
WCAG 2.2, kriteria 2.5.5 *Target Size (Enhanced)*, tingkat AAA	44 kali 44 piksel CSS	[6]
Apple Human Interface Guidelines	44 kali 44 *point*	untuk kontrol di iPhone dan iPad [18]
Android dan Material Design	48 kali 48 piksel mandiri-kerapatan (dp)	sekitar 9 mm di layar [19]

Pada kios publik yang dipakai orang lanjut usia yang berdiri sejauh jangkauan lengan, angka-angka minimum ini adalah lantai, bukan sasaran: tombol kios lama, setinggi 20 mm, lebih besar daripada semuanya dan tetap terlalu kecil bagi Bu Siti.

2.7 Hukum Hick: Berapa Lama Memutuskan

2.7.1 Hukumnya

Sebelum tangan bergerak, pikiran harus memilih. Pada 1952 William Hick mengukur berapa lama orang bereaksi terhadap salah satu dari n lampu yang sama mungkin menyala dengan menekan tombol yang sesuai, dan menemukan bahwa waktunya tumbuh sebanding dengan logaritma banyaknya pilihan; Ray Hyman menunjukkan setahun kemudian bahwa yang sebenarnya menentukan adalah *informasi* dalam pilihan itu, yang lebih kecil bila sebagian opsi lebih mungkin daripada yang lain [20], [21]. *Hukum Hick–Hyman* adalah

$$RT = a + b\log_2(n+1) \tag{2.4}$$

untuk n opsi yang sama mungkin, dengan $+1$ yang memperhitungkan kemungkinan tidak menanggapi sama sekali, dan a serta b lagi-lagi konstanta hasil pengukuran. Ketika opsi-opsinya punya peluang $p_1, p_2, \ldots, p_n$, banyaknya bit menjadi *entropi*

$$H = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} \tag{2.5}$$

dan $RT = a + bH$ [21]. Dengan konstanta latihan dari Subbab 2.9, $a = 0.227$ s dan $b = 0.205$ s/bit, memilih di antara dua belas tombol kios lama memakan $0.227 + 0.205\log_2 13 = 0.227 + 0.205 \times 3.700 = 0.986$ s, dan memilih di antara tujuh tombol layar pertama kios baru $0.227 + 0.205 \times 3 = 0.842$ s.

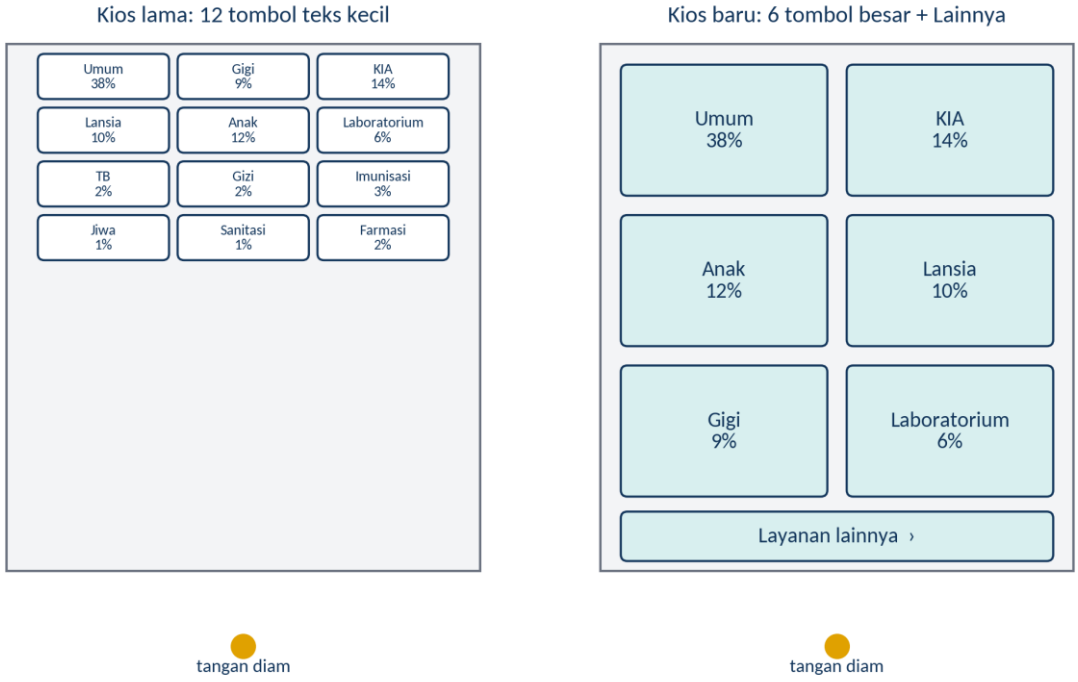
2.7.2 Kapan hukum ini berlaku, dan kapan tidak

Hukum Hick menggambarkan orang yang *sudah tahu* opsi-opsinya beserta letaknya dan hanya perlu memilih satu, seperti kasir yang terlatih atau pasien yang datang setiap bulan. Hukum ini tidak menggambarkan pendatang baru yang *mencari* dalam daftar label teks yang belum dikenalnya. Pencarian semacam itu lebih mirip membaca label satu per satu, dan waktunya tumbuh kurang lebih sebanding dengan banyaknya label, bukan dengan logaritmanya [3]. Karena itu hukum ini tidak boleh dipakai untuk membenarkan menu yang panjang (“logaritma tumbuh lambat, jadi empat puluh butir tidak masalah”): bagi pengguna pertama kali, empat puluh butir yang belum dikenal tetap lambat, bagaimanapun cara menghitungnya.

Pelajaran praktisnya adalah sebagai berikut. Opsi yang lebih sedikit berarti keputusan yang lebih cepat, jadi singkirkan dari layar pertama opsi yang hampir tidak diperlukan siapa pun. Urutkan dan kelompokkan sisanya sehingga mencari berubah menjadi mengenali: butir yang sering dipakai lebih dulu, butir yang berkaitan berdekatan (lagi-lagi Gestalt). Dan pilihlah menu yang *lebar dan dangkal* daripada yang *sempit dan dalam* ketika setiap tingkat memerlukan keputusan dan gerakan baru: Soal Latihan 2.2 menghitung sebabnya.

2.8 Membangun Kalkulatornya Langkah demi Langkah

Tim dr. Rina kini punya usulan: enam tombol besar untuk enam layanan yang diperlukan 89% pasien, dan sebuah tombol lebar *Layanan lainnya* yang menuju layar kedua berisi enam layanan sisanya. **Gambar 2.3** memperlihatkan kedua tata letak sesuai skala, dengan porsi pasien untuk setiap layanan yang diambil dari hitungan puskesmas sendiri pada bulan sebelumnya.



Gambar 2.3 Layar pertama kios antrean yang lama dan yang baru, sesuai skala, dengan porsi pasien setiap layanan dan titik diam tangan.

Apakah usulan itu lebih cepat, bagi siapa, dan seberapa banyak? Program di subbab ini menjawabnya dengan kedua hukum. Program ini melakukan tiga hal: mencocokkan hukum Fitts pada log ketukan dan hukum Hick pada log pilihan untuk memperoleh konstanta a dan b; membaca setiap tata letak dari sebuah tabel kecil; lalu menghitung, untuk setiap layanan, waktu yang diharapkan dari tangan yang siap hingga tombol ditekan, serta rata-ratanya atas semua pasien yang dibobot dengan porsinya.

2.8.1 Berkas datanya

Tata letaknya berupa berkas CSV. **Kode 2.1** adalah kios lama: setiap tombol dengan layar, label, posisi, dan ukurannya dalam milimeter (posisinya adalah sudut kiri bawah), serta porsi pasiennya.

Kode 2.1 `data/kiosk_old.csv`: tata letak lama, dua belas tombol dalam satu layar.

```
screen,label,x_mm,y_mm,w_mm,h_mm,share
1,General,15,215,60,20,0.38
1,Dental,80,215,60,20,0.09
1,Mother and child,145,215,60,20,0.14
1,Elderly,15,190,60,20,0.10
1,Children,80,190,60,20,0.12
1,Laboratory,145,190,60,20,0.06
1,Tuberculosis,15,165,60,20,0.02
1,Nutrition,80,165,60,20,0.02
1,Immunization,145,165,60,20,0.03
1,Mental health,15,140,60,20,0.01
1,Sanitation,80,140,60,20,0.01
1,Pharmacy,145,140,60,20,0.02
```

Kode 2.2 adalah tata letak baru. Tombol berlabel `Other` tidak punya porsi sendiri; ia menuju layar 2.

Kode 2.2 `data/kiosk_new.csv`: tata letak baru, enam tombol besar dan *Other* di layar 1, enam lagi di layar 2.

```
screen,label,x_mm,y_mm,w_mm,h_mm,share
1,General,10,170,95,60,0.38
1,Mother and child,115,170,95,60,0.14
1,Children,10,100,95,60,0.12
1,Elderly,115,100,95,60,0.10
1,Dental,10,30,95,60,0.09
1,Laboratory,115,30,95,60,0.06
1,Other,10,0,200,22,
2,Immunization,10,170,95,60,0.03
2,Tuberculosis,115,170,95,60,0.02
2,Nutrition,10,100,95,60,0.02
2,Pharmacy,115,100,95,60,0.02
2,Mental health,10,30,95,60,0.01
2,Sanitation,115,30,95,60,0.01
```

Konstanta a dan b harus berasal dari pengukuran terhadap orang sungguhan dengan perangkat sungguhan, idealnya dari pasien itu sendiri. Folder bab ini berisi dua log *latihan*, `taps.csv` (60 ketukan pada empat jarak dan tiga lebar) dan `choices.csv` (48 pilihan di antara 1 hingga 12 opsi). Keduanya bukan hasil pengukuran: keduanya dibangkitkan oleh skrip kecil `make_practice_logs.py` dari kedua hukum dengan konstanta pilihan ditambah derau acak, agar proses pencocokan punya sesuatu yang realistis untuk ditemukan kembali, dan keduanya menggantikan log yang Anda rekam sendiri pada Soal Latihan 2.6. Baris-baris pertama log ketukan tampak seperti ini:

```
distance_mm,width_mm,seconds
40,8,0.577
40,8,0.574
```

2.8.2 Mencocokkan garis dengan kuadrat terkecil

Kedua hukum berbentuk $y = a + bx$: untuk hukum Fitts, x adalah ID dan y adalah MT; untuk hukum Hick, x adalah $\log_2(n+1)$ dan y adalah RT. Diberikan titik-titik hasil pengukuran $(x_1, y_1), \ldots, (x_n, y_n)$, garis *kuadrat terkecil* (*least squares*) adalah garis yang membuat jumlah kuadrat jarak vertikal dari titik-titik ke garis sekecil mungkin. Menyamakan turunan jumlah itu terhadap a dan b dengan nol menghasilkan rumus yang sudah dikenal

$$b = \frac{\sum_i (x_i - \bar{x})\,(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}, \qquad a = \bar{y} - b\,\bar{x} \tag{2.6}$$

dengan $\bar{x}$ dan $\bar{y}$ sebagai rata-rata. Seberapa baik garis itu cocok diukur dengan

$$r^2 = 1 - \frac{\sum_i (y_i - a - bx_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{2.7}$$

yaitu bagian variasi y yang dijelaskan oleh garis; 1 berarti cocok sempurna dan 0 berarti garis tidak menjelaskan apa pun.

2.8.3 Waktu harapan sebuah tata letak

Untuk layanan di layar pertama, waktunya adalah keputusan di antara semua tombol layar 1 (Persamaan (2.4)) ditambah gerakan dari tangan yang diam ke tombol itu (Persamaan (2.2)). Untuk layanan di layar kedua, waktunya adalah keputusan dan gerakan ke *Other*, lalu keputusan di antara tombol layar 2 dan gerakan dari *Other* ke layanan itu. Waktu harapan per pasien adalah jumlah, atas semua layanan, dari waktu dikali porsi. Perhitungan ini sengaja mengabaikan waktu kios berganti layar dan waktu membaca tata letak yang baru pertama kali dilihat, yang keduanya dibahas di Subbab 2.9.

2.8.4 Programnya

Kode 2.3 adalah versi Python-nya.

Kode 2.3 `humanfactors.py`: mencocokkan hukum Fitts dan Hick pada log dan menghitung waktu harapan setiap tata letak kios.

```
"""Hukum Fitts dan Hick: cocokkan ke log, lalu bandingkan dua tata letak kios."""
import csv
import math
import sys

HOME = (110.0, -40.0)      # tangan diam di birai di bawah layar (mm)


def fitts_id(distance, width):
    """Indeks kesulitan dalam bit (bentuk Shannon)."""
    return math.log2(distance / width + 1)


def hick_bits(choices):
    """Bit sebuah pilihan di antara n opsi sama mungkin (plus 'tidak ada')."""
    return math.log2(choices + 1)


def fit_line(xs, ys):
    """Kuadrat terkecil: a dan b dari y = a + b*x, serta r kuadrat."""
    n = len(xs)
    mx, my = sum(xs) / n, sum(ys) / n
    sxy = sum((x - mx) * (y - my) for x, y in zip(xs, ys))
    sxx = sum((x - mx) ** 2 for x in xs)
    b = sxy / sxx
    a = my - b * mx
    ss_res = sum((y - a - b * x) ** 2 for x, y in zip(xs, ys))
    ss_tot = sum((y - my) ** 2 for y in ys)
    return a, b, 1 - ss_res / ss_tot


def read_rows(path):
    with open(path, newline="", encoding="utf-8") as f:
        return list(csv.DictReader(f))


def centre(b):
    return (float(b["x_mm"]) + float(b["w_mm"]) / 2,
            float(b["y_mm"]) + float(b["h_mm"]) / 2)
```

Kode 2.3 (Lanjutan) *humanfactors.py: mencocokkan hukum Fitts dan Hick pada log dan menghitung waktu harapan setiap tata letak kios.*

```
def move_time(fitts, start, button):
    """Waktu Fitts dari titik ke tombol; lebarnya adalah sisi terpendek."""
    cx, cy = centre(button)
    d = math.hypot(cx - start[0], cy - start[1])
    w = min(float(button["w_mm"]), float(button["h_mm"]))
    return fitts[0] + fitts[1] * fitts_id(d, w)


def service_times(layout, fitts, hick):
    """Detik dari tangan siap hingga layanan ditekan, per layanan."""
    screen1 = [b for b in layout if b["screen"] == "1"]
    screen2 = [b for b in layout if b["screen"] == "2"]
    other = next((b for b in screen1 if b["label"] == "Other"), None)
    think1 = hick[0] + hick[1] * hick_bits(len(screen1))
    times = {}
    for b in screen1:
        if b is not other:
            times[b["label"]] = (think1 + move_time(fitts, HOME, b),
                                 float(b["share"]))
    if other is not None:
        via = think1 + move_time(fitts, HOME, other)
        think2 = hick[0] + hick[1] * hick_bits(len(screen2))
        for b in screen2:
            t = via + think2 + move_time(fitts, centre(other), b)
            times[b["label"]] = (t, float(b["share"]))
    return times


def main():
    folder = sys.argv[1] if len(sys.argv) > 1 else "data"
    taps = read_rows(f"{folder}/taps.csv")
    ids = [fitts_id(float(r["distance_mm"]), float(r["width_mm"]))
           for r in taps]
    fitts = fit_line(ids, [float(r["seconds"]) for r in taps])
    print(f"Fitts: MT = {fitts[0]:.3f} + {fitts[1]:.3f} * ID   "
          f"(r2 = {fitts[2]:.3f}, {len(taps)} taps)")
    print(f"       throughput 1/b = {1 / fitts[1]:.2f} bits/s")
    picks = read_rows(f"{folder}/choices.csv")
    hick = fit_line([hick_bits(int(r["choices"])) for r in picks],
                    [float(r["seconds"]) for r in picks])
    print(f"Hick:  RT = {hick[0]:.3f} + {hick[1]:.3f} * log2(n+1)   "
          f"(r2 = {hick[2]:.3f}, {len(picks)} choices)")
    for name in ("kiosk_old", "kiosk_new"):
        times = service_times(read_rows(f"{folder}/{name}.csv"), fitts, hick)
        mean = sum(t * s for t, s in times.values())
        slow = max(times, key=lambda k: times[k][0])
```

Kode 2.3 (Lanjutan) *humanfactors.py: mencocokkan hukum Fitts dan Hick pada log dan menghitung waktu harapan setiap tata letak kios.*

```
        print(f"{name}: expected {mean:.2f} s per patient; "
              f"slowest {slow} {times[slow][0]:.2f} s")
        for label, (t, s) in sorted(times.items(), key=lambda kv: -kv[1][1]):
            print(f"  {label:17}{s:6.0%}{t:7.2f} s")


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

`fitts_id` dan `hick_bits` adalah Persamaan (2.1) dan logaritma pada Persamaan (2.4). `fit_line` adalah Persamaan (2.6) dan Persamaan (2.7) yang dituliskan. `read_rows` membaca berkas CSV menjadi daftar *dictionary*, satu per baris, dengan kunci nama-nama kolom di baris pertama.

`centre` mencari titik tengah tombol dari sudut dan ukurannya, dan `move_time` menerapkan hukum Fitts dari sebuah titik awal ke sebuah tombol, dengan sisi terpendek sebagai W. `service_times` mewujudkan Subbab 2.8.3: fungsi ini membagi tata letak menjadi dua layar, mencari tombol `Other` bila ada, lalu mengembalikan pasangan waktu dan porsi untuk setiap layanan. `main` mencocokkan kedua hukum, mencetak konstantanya, lalu mencetak waktu harapan dan waktu setiap layanan untuk kedua tata letak, yang paling sering lebih dulu.

Kode 2.4 adalah versi Java, yang mencetak baris-baris yang persis sama.

Kode 2.4 `HumanFactors.java`: pencocokan dan perbandingan tata letak yang sama dalam Java.

```java
import java.io.IOException;
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;

/** Hukum Fitts dan Hick: cocokkan ke log, lalu bandingkan dua tata letak kios. */
public class HumanFactors {
    // Tangan diam di birai di bawah layar (mm).
    static final double HOME_X = 110, HOME_Y = -40;

    /** Indeks kesulitan dalam bit (bentuk Shannon). */
    static double fittsId(double distance, double width) {
        return Math.log(distance / width + 1) / Math.log(2);
    }

    /** Bit sebuah pilihan di antara n opsi sama mungkin (plus "tidak ada"). */
    static double hickBits(int choices) {
        return Math.log(choices + 1) / Math.log(2);
    }

    /** Kuadrat terkecil: {a, b, r kuadrat} dari y = a + b*x. */
    static double[] fitLine(List<Double> xs, List<Double> ys) {
        int n = xs.size();
        double mx = 0, my = 0;
        for (int i = 0; i < n; i++) { mx += xs.get(i) / n; my += ys.get(i) / n; }
        double sxy = 0, sxx = 0, ssTot = 0;
        for (int i = 0; i < n; i++) {
            sxy += (xs.get(i) - mx) * (ys.get(i) - my);
            sxx += (xs.get(i) - mx) * (xs.get(i) - mx);
            ssTot += (ys.get(i) - my) * (ys.get(i) - my);
        }
        double b = sxy / sxx, a = my - b * mx, ssRes = 0;
        for (int i = 0; i < n; i++) {
            double e = ys.get(i) - a - b * xs.get(i);
            ssRes += e * e;
        }
        return new double[] {a, b, 1 - ssRes / ssTot};
    }
```

Kode 2.4 (Lanjutan) *HumanFactors.java: pencocokan dan perbandingan tata letak yang sama dalam Java.*

```
    /** Berkas CSV sebagai daftar baris, tiap baris peta nama kolom ke teks. */
    static List<Map<String, String>> readRows(String path) throws IOException {
        List<String> lines = Files.readAllLines(Path.of(path));
        String[] head = lines.get(0).split(",");
        List<Map<String, String>> rows = new ArrayList<>();
        for (String line : lines.subList(1, lines.size())) {
            String[] cells = line.split(",", -1);
            Map<String, String> row = new HashMap<>();
            for (int i = 0; i < head.length; i++) row.put(head[i], cells[i]);
            rows.add(row);
        }
        return rows;
    }

    static double num(Map<String, String> row, String key) {
        return Double.parseDouble(row.get(key));
    }

    static double[] centre(Map<String, String> b) {
        return new double[] {num(b, "x_mm") + num(b, "w_mm") / 2,
                             num(b, "y_mm") + num(b, "h_mm") / 2};
    }

    /** Waktu Fitts dari titik ke tombol; lebarnya adalah sisi terpendek. */
    static double moveTime(double[] fitts, double[] start, Map<String, String> b) {
        double[] c = centre(b);
        double d = Math.hypot(c[0] - start[0], c[1] - start[1]);
        double w = Math.min(num(b, "w_mm"), num(b, "h_mm"));
        return fitts[0] + fitts[1] * fittsId(d, w);
    }

    /** Detik dari tangan siap hingga layanan ditekan: {waktu, porsi}. */
    static Map<String, double[]> serviceTimes(List<Map<String, String>> layout,
                                              double[] fitts, double[] hick) {
        List<Map<String, String>> s1 = new ArrayList<>(), s2 = new ArrayList<>();
        Map<String, String> other = null;
        for (var b : layout) {
            (b.get("screen").equals("1") ? s1 : s2).add(b);
            if (b.get("label").equals("Other")) other = b;
        }
        double think1 = hick[0] + hick[1] * hickBits(s1.size());
        double[] home = {HOME_X, HOME_Y};
        Map<String, double[]> times = new LinkedHashMap<>();
        for (var b : s1) {
            if (b != other) {
                times.put(b.get("label"), new double[] {
```

Kode 2.4 (Lanjutan) *HumanFactors.java: pencocokan dan perbandingan tata letak yang sama dalam Java.*

```
                    think1 + moveTime(fitts, home, b), num(b, "share")});
            }
        }
        if (other != null) {
            double via = think1 + moveTime(fitts, home, other);
            double think2 = hick[0] + hick[1] * hickBits(s2.size());
            for (var b : s2) {
                double t = via + think2 + moveTime(fitts, centre(other), b);
                times.put(b.get("label"), new double[] {t, num(b, "share")});
            }
        }
        return times;
    }

    public static void main(String[] args) throws IOException {
        String folder = args.length > 0 ? args[0] : "data";
        List<Double> ids = new ArrayList<>(), mt = new ArrayList<>();
        var taps = readRows(folder + "/taps.csv");
        for (var r : taps) {
            ids.add(fittsId(num(r, "distance_mm"), num(r, "width_mm")));
            mt.add(num(r, "seconds"));
        }
        double[] fitts = fitLine(ids, mt);
        System.out.printf("Fitts: MT = %.3f + %.3f * ID   (r2 = %.3f, %d taps)%n",
                          fitts[0], fitts[1], fitts[2], taps.size());
        System.out.printf("       throughput 1/b = %.2f bits/s%n", 1 / fitts[1]);
        List<Double> bits = new ArrayList<>(), rt = new ArrayList<>();
        var picks = readRows(folder + "/choices.csv");
        for (var r : picks) {
            bits.add(hickBits(Integer.parseInt(r.get("choices"))));
            rt.add(num(r, "seconds"));
        }
        double[] hick = fitLine(bits, rt);
        System.out.printf("Hick:  RT = %.3f + %.3f * log2(n+1)   "
                          + "(r2 = %.3f, %d choices)%n",
                          hick[0], hick[1], hick[2], picks.size());
        for (String name : new String[] {"kiosk_old", "kiosk_new"}) {
            var times = serviceTimes(readRows(folder + "/" + name + ".csv"),
                                     fitts, hick);
            double mean = 0;
            String slow = null;
            for (var e : times.entrySet()) {
                mean += e.getValue()[0] * e.getValue()[1];
                if (slow == null || e.getValue()[0] > times.get(slow)[0]) {
                    slow = e.getKey();
                }
```

Kode 2.4 (Lanjutan) *HumanFactors.java: pencocokan dan perbandingan tata letak yang sama dalam Java.*

```
            }
            System.out.printf("%s: expected %.2f s per patient; "
                              + "slowest %s %.2f s%n",
                              name, mean, slow, times.get(slow)[0]);
            var list = new ArrayList<>(times.entrySet());
            list.sort((p, q) -> Double.compare(q.getValue()[1], p.getValue()[1]));
            for (var e : list) {
                System.out.printf("  %-17s%5.0f%%%7.2f s%n", e.getKey(),
                                  100 * e.getValue()[1], e.getValue()[0]);
            }
        }
    }
}
```

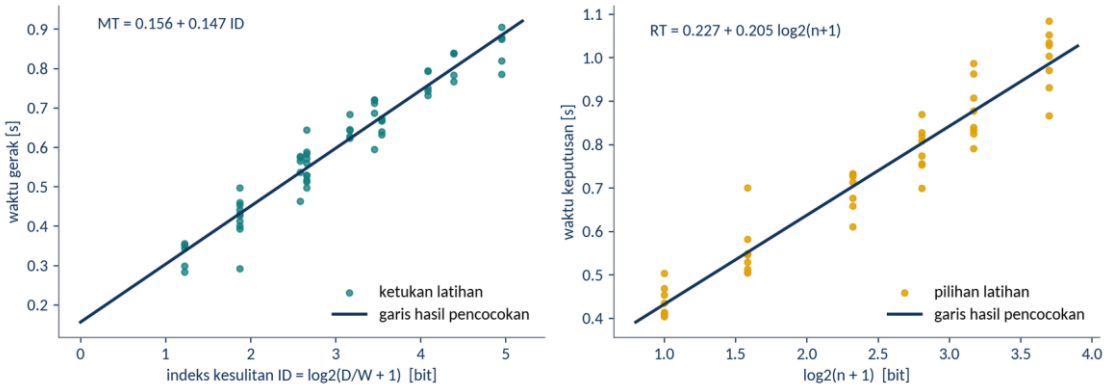
Java tidak punya pembaca CSV bawaan, sehingga `readRows` memecah setiap baris di tanda koma (argumen `-1` mempertahankan sel terakhir yang kosong, seperti porsi `Other` yang tidak ada) dan membangun sebuah `HashMap` per baris. `Math.log` di Java adalah logaritma natural, sehingga logaritma basis 2 ditulis `Math.log(x) / Math.log(2)`. `LinkedHashMap` menjaga layanan dalam urutan berkas, dan pengurutan penutup berdasarkan porsi bersifat stabil, sehingga layanan dengan porsi sama tetap dalam urutan itu, persis seperti di Python.

2.9 Menjalankannya

```
$ python3 humanfactors.py
$ java -cp out HumanFactors
```

```
Fitts: MT = 0.156 + 0.147 * ID   (r2 = 0.932, 60 taps)
       throughput 1/b = 6.81 bits/s
Hick:  RT = 0.227 + 0.205 * log2(n+1)   (r2 = 0.922, 48 choices)
kiosk_old: expected 1.70 s per patient; slowest General 1.71 s
  General             38%   1.71 s
  Mother and child    14%   1.71 s
  Children            12%   1.68 s
  Elderly             10%   1.69 s
  Dental               9%   1.70 s
  Laboratory           6%   1.69 s
  Immunization         3%   1.67 s
  Tuberculosis         2%   1.67 s
  Nutrition            2%   1.66 s
  Pharmacy             2%   1.65 s
  Mental health        1%   1.65 s
  Sanitation           1%   1.64 s
kiosk_new: expected 1.44 s per patient; slowest Immunization 2.52 s
  General             38%   1.34 s
  Mother and child    14%   1.34 s
  Children            12%   1.29 s
  Elderly             10%   1.29 s
  Dental               9%   1.22 s
  Laboratory           6%   1.22 s
  Immunization         3%   2.52 s
  Tuberculosis         2%   2.52 s
  Nutrition            2%   2.45 s
  Pharmacy             2%   2.45 s
  Mental health        1%   2.38 s
  Sanitation           1%   2.38 s
```

Tiga baris pertama adalah hukum hasil pencocokan. Untuk log latihan, hukum Fitts adalah $MT = 0.156 + 0.147\,ID$ dengan $r^2 = 0.932$ dan *throughput* 6,81 bit per detik, dan hukum Hick adalah $RT = 0.227 + 0.205\log_2(n+1)$ dengan $r^2 = 0.922$. **Gambar 2.4** memperlihatkan titik-titik dan garis hasil pencocokannya: sebaran di sekitar setiap garis adalah derau yang dimiliki setiap pengukuran nyata, dan garis-garis itu menemukan kembali konstanta pilihannya dengan dekat (0,18 dan 0,14 untuk Fitts, 0,25 dan 0,20 untuk Hick) tetapi tidak persis, sebagaimana mestinya bila 60 atau 48 titik berderau dicocokkan.



Gambar 2.4 Log latihan dengan garis hasil pencocokan: hukum Fitts (kiri) dan hukum Hick (kanan).

Dua blok tata letak menjawab pertanyaan dr. Rina. Kios lama memerlukan rata-rata 1,70 s per pasien, hampir sama untuk setiap layanan, karena semua tombolnya sama kecil dan sama jauh. Kios baru memerlukan 1,44 s, 15% lebih sedikit, dan enam layanan yang umum, 89% pasien, masing-masing memerlukan 1,22 hingga 1,34 s. Harganya dibayar oleh 11% pasien yang memerlukan layanan di layar kedua: mereka perlu sekitar 2,4 hingga 2,5 s, satu keputusan dan satu gerakan tambahan.

Dua catatan menjaga hasil ini tetap jujur. Pertama, ini adalah waktu bagi orang yang sudah mengenal tata letaknya: model ini tidak punya suku untuk membaca label yang belum dikenal, yang bagi pengguna pertama kali memakan waktu jauh lebih lama daripada 1,7 detik dan justru di situlah teks kecil kios lama paling merugikan Bu Siti. Desain ulang juga membantu di sana, dengan label yang lebih sedikit, teks yang lebih besar, dan pengelompokan, tetapi model ini tidak dapat mengukurnya; Bab 7 mengukurnya dengan uji *usability*. Kedua, hukum-hukum ini tidak mengatakan apa pun tentang *kesalahan*. Tiket keliru Bu Siti berasal dari sasaran yang tertutup jarinya; tombol baru, yang tiga kali lebih tinggi, membuat kesalahan itu jauh lebih kecil kemungkinannya. Perbaikan terpenting dari sebuah desain ulang sering kali justru yang tidak terlihat oleh model waktu, dan perancang yang baik memakai model untuk apa yang diukurnya dan metode lain untuk sisanya.

2.10 Mengujinya

Kode 2.5 menguji bagian-bagian yang jawabannya diketahui dengan pasti: dua indeks kesulitan ($\log_2(30/10+1) = \log_2 4 = 2$ dan $\log_2(70/10+1) = \log_2 8 = 3$), satu nilai Hick ($\log_2 8 = 3$), dan garis yang dicocokkan melalui tiga titik yang tepat berada pada $y = 0.3 + 0.2x$.

Kode 2.5 `test_humanfactors.py`: uji kedua hukum dan pencocokan kuadrat terkecil.

```python
"""Uji Fitts dan Hick: python -m unittest test_humanfactors.py"""
import unittest
from humanfactors import fitts_id, hick_bits, fit_line


class HumanFactorsTests(unittest.TestCase):
    def test_index_of_difficulty(self):
        self.assertAlmostEqual(fitts_id(30, 10), 2.0)
        self.assertAlmostEqual(fitts_id(70, 10), 3.0)

    def test_hick_bits(self):
        self.assertAlmostEqual(hick_bits(7), 3.0)

    def test_fit_line_recovers_a_perfect_line(self):
        a, b, r2 = fit_line([1, 2, 3], [0.5, 0.7, 0.9])   # tepat 0.3 + 0.2x
        self.assertAlmostEqual(a, 0.3)
        self.assertAlmostEqual(b, 0.2)
        self.assertAlmostEqual(r2, 1.0)


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

Kode 2.6 menjalankan enam pemeriksaan yang sama dalam Java.

Kode 2.6 `HumanFactorsTest.java`: pemeriksaan yang sama dalam Java.

```java
/** Uji kalkulator Fitts dan Hick: java -cp out HumanFactorsTest */
public class HumanFactorsTest {
    static int passed = 0;

    static void near(double got, double wanted, String what) {
        if (Math.abs(got - wanted) > 1e-9) {
            throw new AssertionError(what + ": wanted " + wanted + ", got " + got);
        }
        passed++;
    }

    public static void main(String[] args) throws Exception {
        near(HumanFactors.fittsId(30, 10), 2.0, "ID of D=30, W=10");
        near(HumanFactors.fittsId(70, 10), 3.0, "ID of D=70, W=10");
        near(HumanFactors.hickBits(7), 3.0, "7 choices are 3 bits");
        var xs = java.util.List.of(1.0, 2.0, 3.0);
        var ys = java.util.List.of(0.5, 0.7, 0.9);    // tepat 0.3 + 0.2x
        double[] f = HumanFactors.fitLine(xs, ys);
        near(f[0], 0.3, "intercept");
        near(f[1], 0.2, "slope");
        near(f[2], 1.0, "perfect fit");
        System.out.println("All " + passed + " checks passed.");
    }
}
```

```
$ python3 -m unittest test_humanfactors.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
$ java -cp out HumanFactorsTest
All 6 checks passed.
```

2.11 Memakainya Sungguh-sungguh

Tinjauan faktor manusia atas layar apa pun, baik kios, aplikasi, maupun formulir, dapat mengikuti tiga pertanyaan di Subbab 2.2 sebagai daftar periksa. **Tabel 2.2** mengubahnya menjadi

pertanyaan yang diajukan sambil melihat sebuah desain, idealnya bersama seseorang dari kelompok pengguna yang dituju.

Tabel 2.2 Daftar periksa faktor manusia untuk meninjau sebuah layar.

Tahap	Pertanyaan	Yang diperhatikan
Persepsi	Dapatkah setiap pengguna melihat dan membacanya?	Ukuran teks dan kontras untuk mata yang menua; makna tidak hanya dibawa warna; butir yang berkaitan dikelompokkan dengan kedekatan atau wilayah
Perhatian	Akankah hal yang penting diperhatikan?	Satu aksi utama yang jelas; umpan balik di tempat pengguna memandang; tidak ada animasi yang berebut perhatian
Memori	Muatkah tugas ini dalam sekitar empat bongkahan?	Mengenali alih-alih mengingat; angka panjang dikelompokkan; tidak ada informasi yang harus dibawa antarlayar
Keputusan	Berapa banyak opsi di setiap langkah?	Opsi yang jarang dipakai dipindah dari layar pertama; yang sering dipakai lebih dulu
Gerak	Dapatkah setiap sasaran dijangkau dan dikenai?	Sasaran minimal seukuran **Tabel 2.1**, lebih besar untuk kios publik; sasaran yang sering dekat; yang berbahaya berjauhan

Untuk memakai kalkulator ini bagi desain Anda sendiri, ganti kedua berkas tata letak dengan tombol-tombol Anda, diukur pada layar sebenarnya dalam milimeter (atau piksel, asalkan semua ukuran memakai satuan yang sama), dan ganti porsinya dengan hitungan nyata. Yang terpenting, ganti log latihan dengan pengukuran terhadap pengguna Anda. Soal Latihan 2.6 menjelaskan prosedur sederhana dengan ponsel atau tablet dan seorang pembantu yang memegang aplikasi *stopwatch*; bahkan dua puluh ketukan dan dua puluh pilihan dari tiga orang seusia pengguna yang dituju menghasilkan konstanta yang lebih dapat dipercaya untuk kios Anda daripada nilai buku teks mana pun.

Pendekatan yang sama berlaku jauh melampaui kios. Halaman pesanan warung, daftar harga laundry di ponsel, layar penjemputan aplikasi ojek: masing-masing punya sasaran yang sering dan yang jarang dipakai, dan masing-masing dapat diperbaiki dengan membuat yang sering dipakai besar, dekat, dan sedikit.

2.12 IMK dalam Praktik: Profitina

Pengguna Profitina adalah pemilik merek yang membuka dasbor *AI Visibility* di sela tugas lain, sering di ponsel, dan desain produknya mencerminkan batas-batas yang dibahas bab ini. Dasbor mengumpulkan ke satu layar segala hal yang biasanya diperlukan pengguna berikutnya, dengan setiap notifikasi dapat diklik langsung ke bagiannya: itu menghapus satu keputusan dan beberapa gerakan untuk jalur yang paling sering, sesuai penalaran hukum Hick dan Fitts. Tren skor mingguan ditampilkan sebagai garis, bukan tabel angka, sehingga persepsi, bukan memori kerja, yang menjawab pertanyaan “apakah saya membaik?”. Dan di kanal komunitas, penghitung “N baru” hanya menghitung tulisan tingkat teratas, sehingga satu angka yang bermakna tetap terlihat alih-alih meminta anggota mengingat apa yang sudah mereka baca.

Batas memori juga membentuk layar akun yang diwajibkan UU PDP. Mengganti alamat email atau kata sandi, mengekspor data, atau menghapus akun adalah tugas yang jarang, yang berarti pengguna menjumpainya tanpa latihan: mereka harus mengandalkan pengenalan, sehingga masing-masing dibuat sebagai aksi terpisah yang berlabel jelas dengan akibatnya dituliskan, bukan kode atau pengaturan yang terkubur dalam daftar panjang.

2.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

1. **Merancang teks dan sasaran untuk mata dan tangan muda.** Ujilah dengan pengguna lanjut usia dan ikuti daftar periksa **Tabel 2.2**; pengguna yang paling membutuhkan layanan publik sering kali justru yang penglihatannya paling lemah.
2. **Memakai warna sebagai satu-satunya sinyal.** Tambahkan kata, ikon, atau posisi; 1 dari 12 laki-laki tidak dapat mengandalkan merah dan hijau.
3. **Memercayai "tujuh plus atau minus dua".** Rancanglah untuk sekitar empat bongkahan dan biarkan antarmuka membawa sisanya.
4. **Memakai hukum Hick untuk membela menu yang panjang.** Hukum ini berlaku untuk pilihan yang sudah terlatih, bukan untuk mencari di antara label yang belum dikenal; bagi pendatang baru, opsi yang lebih sedikit memang lebih cepat.
5. **Memasukkan konstanta buku teks ke hukum Fitts.** a dan b bergantung pada orang, perangkat, dan posturnya; ukurlah, dan sebutkan asal-usulnya.
6. **Mengambil lebar pada arah yang salah.** Untuk jari yang mendekat dari bawah, tinggi tombol yang lebar dan pipih itulah lebar yang menentukan.
7. **Membaca model waktu sebagai kebenaran seutuhnya.** Model ini mengabaikan kesalahan, proses belajar, dan membaca; gabungkan dengan pengamatan dan uji *usability*.
8. **Meletakkan umpan balik di tempat yang tidak dilihat pengguna.** Letakkan konfirmasi di sebelah aksinya atau pada hasilnya, dan interupsi hanya ketika sebuah kekeliruan akan mahal akibatnya.

2.14 Ringkasan Bab

- *Model Human Processor* menggambarkan manusia sebagai prosesor persepsi, kognitif, dan motorik dengan penyimpan sensorik, memori kerja yang kecil, dan memori jangka panjang yang besar (**Gambar 2.1**); setiap tahap dapat kelebihan beban.
- Penglihatan hanya tajam di fovea; seiring usia datang presbiopia dan kebutuhan akan cahaya serta kontras yang lebih banyak; 1 dari 12 laki-laki mengalami defisiensi penglihatan warna. Prinsip Gestalt kedekatan, kemiripan, wilayah bersama, dan kesinambungan mengelompokkan apa yang kita lihat sebelum kita membaca.
- Perhatian terbatas dan selektif; apa yang tidak diperhatikan sering sama sekali tidak terlihat.
- Memori kerja menampung sekitar empat bongkahan selama beberapa detik; pembongkahan dan mengenali alih-alih mengingat adalah cara mengatasinya.
- Hukum Fitts: $MT = a + b\,ID$ dengan $ID = \log_2(D/W + 1)$ (Persamaan (2.2), Persamaan (2.1)); *throughput* adalah $1/b$. Sasaran sebaiknya besar, dekat, dan minimal seukuran **Tabel 2.1**.
- Hukum Hick: $RT = a + b\log_2(n + 1)$ (Persamaan (2.4)), atau $a + bH$ dengan entropi pilihan yang tidak sama mungkin (Persamaan (2.5)); hukum ini berlaku untuk pilihan yang terlatih, bukan untuk pencarian.
- Setelah dicocokkan pada log latihan, kedua hukum memprediksi 1,70 s per pasien untuk kios lama dan 1,44 s untuk yang baru; tombol yang lebih besar juga mencegah kesalahan yang tidak dihitung model.

2.15 Soal Latihan

1. Sebuah tombol setinggi 9 mm berjarak 120 mm dari jari yang diam. Dengan konstanta latihan $a = 0.156$ s dan $b = 0.147$ s/bit, hitunglah indeks kesulitan dan waktu geraknya. Setinggi apa tombol itu harus dibuat agar MT di bawah 0,60 s?
2. Sebuah menu punya 15 butir yang sama mungkin. Dengan konstanta Hick latihan, bandingkan waktu keputusan satu menu datar berisi 15 butir dengan menu dua tingkat berisi 3 kelompok masing-masing 5 butir (satu keputusan di setiap tingkat). Mana yang lebih cepat, dan seberapa banyak? Abaikan gerakan.
3. Hitunglah entropi H (Persamaan (2.5)) dari kedua belas porsi layanan kios lama, lalu waktu keputusan Hick–Hyman $a + bH$ dengan konstanta latihan. Bandingkan dengan nilai sama mungkin $a + b\log_2 13$ dan jelaskan perbedaannya.
4. Sebuah nomor *virtual account* untuk pembayaran ditampilkan sebagai 8808123456789012. Tuliskan ulang untuk orang yang harus mengetiknya di ponsel lain, dan jelaskan pilihan Anda dari sudut memori kerja.
5. Seorang rekan mengusulkan agar dua belas tombol kecil kios lama dipertahankan tetapi dipindah 140 mm lebih rendah, dekat dengan tangan. Buatlah salinan folder data dengan perubahan ini, jalankan program padanya, lalu bandingkan waktu harapannya dengan tata letak lama dan baru. Apa yang dikatakan hasilnya tentang jarak dan lebar?
6. Jelaskan prosedur untuk merekam log ketukan dan log pilihan Anda sendiri untuk kios dengan sebuah tablet dan seorang pembantu, dalam format `taps.csv` dan `choices.csv`. Siapa partisipannya, dan berapa banyak ketukan dan pilihan yang akan Anda rekam?
7. Layar ruang tunggu Puskesmas Kenanga menampilkan nomor yang sedang dipanggil dengan warna hijau dan nomor berikutnya dengan warna merah, disertai bunyi bip singkat. Daftarlah masalah faktor manusia dari desain ini bagi pasien poli lansia dan usulkan desain yang lebih baik.

2.16 Penyelesaian Soal Latihan

1. $ID = \log_2(120/9 + 1) = \log_2 14.333 = 3.841$ bit dan $MT = 0.156 + 0.147 \times 3.841 = 0.721$ s. Agar $MT < 0.60$ s kita perlu $ID < (0.60 - 0.156)/0.147 = 3.020$ bit, sehingga $120/W + 1 < 2^{3.020} = 8.11$, yaitu $W > 120/7.11 = 16.9$ mm. Tombol setinggi 17 mm mencapai sasaran itu.

2. Datar: $RT = 0.227 + 0.205\log_2 16 = 0.227 + 0.205 \times 4 = 1.047$ s. Dua tingkat: $(0.227 + 0.205\log_2 4) + (0.227 + 0.205\log_2 6) = (0.227 + 0.410) + (0.227 + 0.530) = 1.394$ s. Menu datar lebih cepat sekitar 0,35 s, karena setiap tingkat membayar lagi intersep a dan menambah opsi "+1"-nya sendiri. Dalam praktik setiap tingkat juga memakan gerakan dan pergantian layar, sehingga yang lebar dan dangkal menang lebih jelas lagi, selama daftar datarnya cukup pendek untuk dikenali alih-alih dicari.

3. Dengan porsi 0,38; 0,09; 0,14; 0,10; 0,12; 0,06; 0,02; 0,02; 0,03; 0,01; 0,01; 0,02, entropinya $H = \sum p_i \log_2(1/p_i) = 2.806$ bit, sehingga $RT = 0.227 + 0.205 \times 2.806 = 0.802$ s, dibandingkan 0.986 s untuk $\log_2 13 = 3.700$ bit. Pasien yang terlatih, yang baginya Umum jauh lebih mungkin daripada Sanitasi, membawa ketidakpastian yang lebih kecil daripada yang tersirat dari dua belas opsi yang sama mungkin, dan hukum Hick–Hyman memprediksi pilihan yang lebih cepat. Efek ini hanya berlaku bagi orang yang mengetahui frekuensinya, seperti pasien tetap.

4. Tampilkan sebagai 8808 1234 5678 9012: empat kelompok berisi empat angka, yang dapat ditahan satu bongkahan demi satu bongkahan, disalin, dan diperiksa per kelompok. Enam belas

angka tanpa kelompok melampaui memori kerja berkali-kali lipat, sehingga memaksa pembacaan ulang dan mengundang angka yang terlewat atau terulang. Tombol salin di samping nomor itu lebih baik lagi, karena menghapus tugas memori sama sekali.

5. Salin folder itu, misalnya ke `data_low`, kurangi setiap `y_mm` di `kiosk_old.csv` dengan 140, lalu jalankan `python3 humanfactors.py data_low`. Waktu harapan tata letak lama turun dari 1,70 s menjadi 1,56 s (Umum 1,58 s), tetap lebih lambat daripada 1,44 s tata letak baru. Memindahkan sasaran lebih dekat membantu, tetapi logaritma D/W didominasi oleh lebar yang kecil: tata letak baru menang terutama karena tombolnya tiga kali lebih tinggi, dan karena layar pertamanya punya lebih sedikit opsi.

6. Rekrut setidaknya lima orang dari kelompok pengguna yang sebenarnya, termasuk pasien poli lansia, dan mintalah persetujuan mereka. Pada tablet yang dipasang setinggi kios, tampilkan satu sasaran pada satu waktu pada jarak yang diketahui dari titik diam yang ditandai, untuk empat jarak dan tiga lebar, masing-masing lima ketukan, dalam urutan acak; aplikasi, atau pembantu dengan aplikasi *stopwatch*, mencatat jarak, lebar, dan waktu, dan ketukan yang meleset dicatat terpisah. Untuk pilihan, tampilkan sebuah kata dan n tombol, n dari 1 hingga 12, masing-masing delapan kali, lalu catat waktu hingga tombol yang benar ditekan. Hasilnya 60 ketukan dan 48 pilihan per orang dalam format `taps.csv` dan `choices.csv`; cocokkan untuk setiap orang secara terpisah dan secara bersama-sama, dan pakai konstanta partisipan lanjut usia untuk kios.

7. Warna saja yang memisahkan nomor yang dipanggil dari yang menunggu, dan ini gagal bagi pemirsa dengan defisiensi warna; hijau dan merah dengan kecerahan serupa juga sulit dibedakan orang lain dari kejauhan. Angkanya mungkin terlalu kecil untuk dibaca dari kursi tunggu dengan penglihatan yang menua. Bunyi bip singkat mudah terlewat bila pendengaran menurun atau ruangan bising, dan tidak mengatakan *nomor* atau *poli* mana. Lebih baik: tampilkan nomor yang dipanggil sangat besar dengan kata “DIPANGGIL” serta nama poli dan ruangannya, daftarkan nomor yang menunggu secara terpisah dengan teks lebih kecil, dan umumkan nomor serta polinya dengan suara, dua kali, didahului bunyi lonceng.

Daftar Pustaka

[1] Periskop.id, “Jumlah Lansia Indonesia 2025 11,97%, Indonesia Masuk Aging Population,” May 5, 2026, reporting Badan Pusat Statistik, *Survei Penduduk Antar Sensus (SUPAS) 2025*. [Online]. Available: https://periskop.id/kependudukan/20260505/jumlah-lansia-indonesia-2025-1197-indonesia-masuk-aging-population

[2] S. K. Card, T. P. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*. Hillsdale, NJ, USA: Lawrence Erlbaum, 1983.

[3] C. D. Wickens, W. S. Helton, J. G. Hollands, and S. Banbury, *Engineering Psychology and Human Performance*, 5th ed. New York, NY, USA: Routledge, 2021.

[4] S. J. Czaja, W. R. Boot, N. Charness, and W. A. Rogers, *Designing for Older Adults: Principles and Creative Human Factors Approaches*, 3rd ed. Boca Raton, FL, USA: CRC Press, 2019.

[5] Colour Blind Awareness, “About Colour Blindness,” accessed Sep. 2026. [Online]. Available: https://www.colourblindawareness.org/colour-blindness/

[6] World Wide Web Consortium, *Web Content Accessibility Guidelines (WCAG) 2.2*, W3C Recommendation, success criteria 1.4.1, 2.5.5 and 2.5.8. [Online]. Available: https://www.w3.org/TR/WCAG22/

[7] J. Wagemans, J. H. Elder, M. Kubovy, S. E. Palmer, M. A. Peterson, M. Singh, and R. von der Heydt, “A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure–ground organization,” *Psychological Bulletin*, vol. 138, no. 6, pp. 1172–1217, 2012.

[8] A. M. Treisman and G. Gelade, "A feature-integration theory of attention," *Cognitive Psychology*, vol. 12, no. 1, pp. 97–136, 1980.

[9] Google Design, "Expressive Design: Google's UX Research," accessed Sep. 2026. [Online]. Available: https://design.google/library/expressive-material-design-google-research

[10] D. J. Simons and C. F. Chabris, "Gorillas in our midst: Sustained inattentional blindness for dynamic events," *Perception*, vol. 28, no. 9, pp. 1059–1074, 1999.

[11] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *Psychological Review*, vol. 63, no. 2, pp. 81–97, 1956.

[12] N. Cowan, "The magical number 4 in short-term memory: A reconsideration of mental storage capacity," *Behavioral and Brain Sciences*, vol. 24, no. 1, pp. 87–114, 2001.

[13] D. A. Norman, *The Design of Everyday Things*, revised and expanded ed. New York, NY, USA: Basic Books, 2013.

[14] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," *Journal of Experimental Psychology*, vol. 47, no. 6, pp. 381–391, 1954.

[15] I. S. MacKenzie, "Fitts' law as a research and design tool in human-computer interaction," *Human-Computer Interaction*, vol. 7, no. 1, pp. 91–139, 1992.

[16] International Organization for Standardization, *ISO 9241-411:2012 Ergonomics of human-system interaction — Part 411: Evaluation methods for the design of physical input devices*. Geneva, Switzerland: ISO, 2012.

[17] R. W. Soukoreff and I. S. MacKenzie, "Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI," *International Journal of Human-Computer Studies*, vol. 61, no. 6, pp. 751–789, 2004.

[18] Apple Inc., "Accessibility," *Human Interface Guidelines*, accessed Sep. 2026. [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/accessibility

[19] Google, "Touch target size," *Android Accessibility Help*, accessed Sep. 2026. [Online]. Available: https://support.google.com/accessibility/android/answer/7101858

[20] W. E. Hick, "On the rate of gain of information," *Quarterly Journal of Experimental Psychology*, vol. 4, no. 1, pp. 11–26, 1952.

[21] R. Hyman, "Stimulus information as a determinant of reaction time," *Journal of Experimental Psychology*, vol. 45, no. 3, pp. 188–196, 1953.

Bab 3

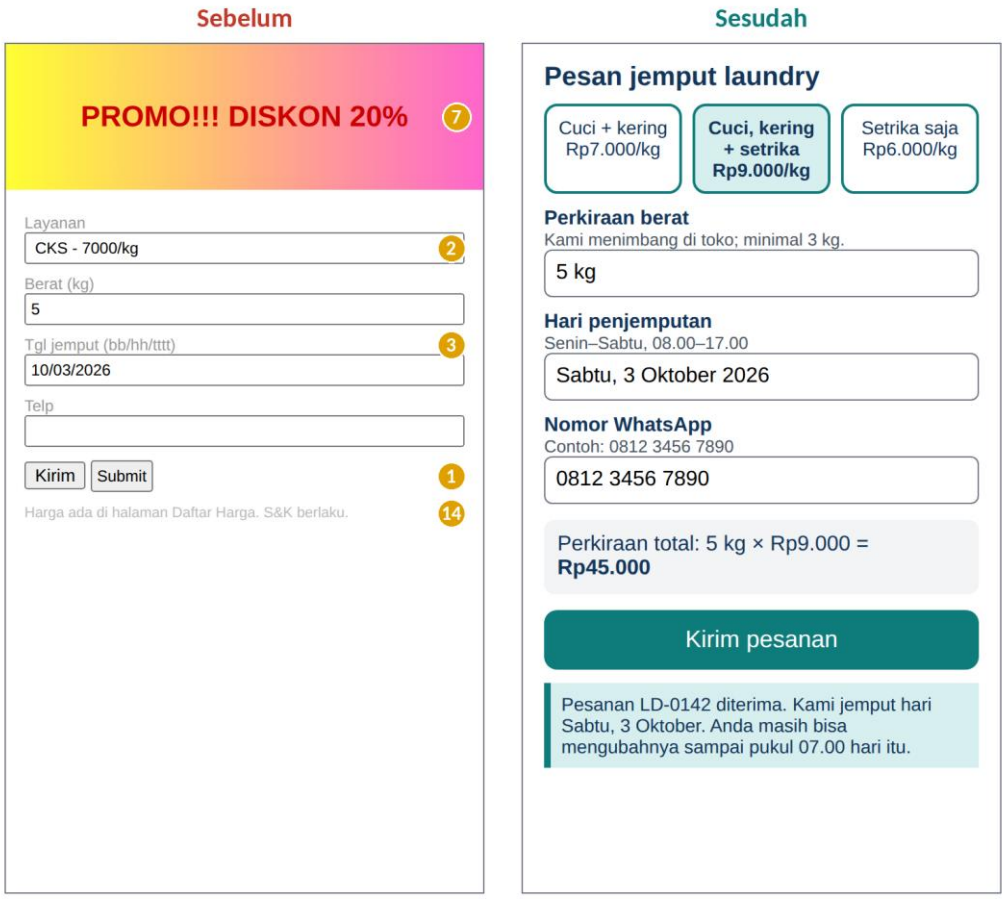
Prinsip dan Heuristik Desain

Sebagian besar masalah *usability* bukanlah hal baru. Tombol yang tidak memberi tanda bahwa ia sudah bekerja, formulir yang menerima tanggal mustahil, pesan galat yang berbunyi “Error 422”, harga yang disembunyikan sampai akhir: masing-masing sudah dilihat, dinamai, dan dipecahkan ribuan kali. Prinsip dan heuristik desain adalah sari pengalaman dari pemecahan-pemecahan itu, aturan singkat yang memungkinkan perancang atau evaluator mengenali masalah yang sudah dikenal sebelum pengguna menderita karenanya. Bab ini menyajikan prinsip-prinsip yang paling penting, mulai dari penanda, pemetaan, batasan, dan umpan balik Norman hingga aturan emas Shneiderman, prinsip interaksi ISO 9241-110, dan sepuluh heuristik *usability* Nielsen; memperlihatkan bagaimana bahasa desain tahun 2025 dan 2026 menerapkan dan kadang meregangkannya; serta mengajarkan evaluasi heuristik, metode termurah yang banyak dipakai untuk menemukan masalah *usability*. Di akhir bab ini Anda akan mampu menjelaskan setiap prinsip dengan contoh sehari-hari; melaksanakan evaluasi heuristik dengan beberapa evaluator dan skala keparahan; memperkirakan dengan model Nielsen–Landauer berapa evaluator yang Anda perlukan; serta mengubah temuan para evaluator menjadi laporan berperingkat dengan program dalam Java dan Python, yang diterapkan pada halaman pesanan Profitina Laundry.

3.1 Laundry Mbak Dewi Meninggalkan WhatsApp

Profitina Laundry adalah sebuah laundry kecil di Surabaya, studi kasus pengajaran dalam buku ini. Mbak Dewi mengelolanya bersama dua pembantu, sebuah sepeda motor untuk penjemputan, dan sederet mesin cuci. Bertahun-tahun setiap pesanan datang lewat WhatsApp: pelanggan mengirim “Mbak, jemput besok ya”, Mbak Dewi membalas dengan menu bernomor (1 untuk cuci kering, 2 untuk cuci kering setrika, 3 untuk setrika saja), lalu menyusul percakapan panjang tentang hari, alamat, dan harga. Cara itu berjalan, tetapi menghabiskan satu jam setiap malam, dan salah paham antara “2” dan “3” sering terjadi.

Pada 2026 seorang pengembang web menawarkan halaman pesanan dengan biaya yang wajar. Versi pertamanya, di sebelah kiri **Gambar 3.1**, tampak sudah jadi dan secara teknis berjalan: setiap pesanan yang dikirim sampai ke basis datanya. Namun pada minggu pertama tiga pelanggan mengirim pesanan yang sama dua kali karena halaman hanya memuat ulang setelah *Kirim*, seorang pelanggan memilih penjemputan hari Minggu padahal toko tutup, dan beberapa orang bertanya lewat WhatsApp berapa yang harus mereka bayar, padahal justru itulah yang seharusnya tidak perlu lagi ditanyakan.



Gambar 3.1 Halaman pesanan Profitina Laundry sebelum dan sesudah evaluasi heuristik, dirender dari halaman pengajaran kita sendiri; penanda bernomor menunjuk masalah P01, P02, P03, P07, dan P14 pada **Kode 3.1**.

Keponakan Mbak Dewi, seorang mahasiswa informatika, mengusulkan cara yang lebih murah daripada menebak dan lebih murah daripada studi pengguna lengkap: mintalah empat orang yang memahami prinsip desain antarmuka untuk memeriksa halaman itu, masing-masing sendiri-sendiri, dengan berpegang pada daftar aturan singkat, lalu gabungkan temuan mereka. Inilah *evaluasi heuristik* (*heuristic evaluation*). Dalam satu sore, keempat evaluator menuliskan 26 temuan, yang ternyata merupakan 14 masalah yang berbeda. Bab ini menjelaskan aturan yang mereka pakai, metodenya, dan program kecil yang menggabungkan temuan mereka menjadi daftar berperingkat yang kemudian dikerjakan pengembang itu satu per satu. Halaman di sebelah kanan gambar adalah hasilnya.

3.2 Gagasannya dengan Kata-Kata Sederhana: Prinsip, Pedoman, dan Heuristik

Ada tiga jenis pengetahuan desain yang sering tercampur.

Prinsip adalah pernyataan umum tentang cara manusia dan antarmuka berinteraksi, yang berlaku lintas produk dan lintas tahun: “orang memerlukan umpan balik untuk tahu bahwa sebuah tindakan berhasil.” Prinsip mengatakan *mengapa*, tetapi tidak persis *apa yang harus dilakukan* pada layar tertentu.

Pedoman (*guideline*) adalah aturan khusus untuk jenis produk tertentu: “tampilkan konfirmasi beserta nomor pesanan setelah pesanan dikirim,” atau “sasaran sentuh minimal 48 kali 48 dp.”

Pedoman adalah bahan penyusun *design system* seperti Material Design, Human Interface Guidelines dari Apple, dan Fluent 2 dari Microsoft. Pedoman itu konkret, tetapi ikut menua bersama platformnya.

Heuristik adalah aturan praktis di antara keduanya: cukup umum untuk berlaku pada antarmuka apa pun, cukup konkret untuk diperiksa pada sebuah layar. "Visibilitas status sistem: desain harus selalu memberi tahu pengguna apa yang sedang terjadi, melalui umpan balik yang sesuai dalam waktu yang wajar" adalah heuristik pertama Nielsen [1]. Heuristik dibuat untuk evaluasi: evaluator menelusuri sebuah layar dan bertanya, untuk setiap heuristik, apakah heuristik itu dilanggar dan di mana.

> **DEFINISI.** *Evaluasi heuristik* adalah metode inspeksi yang di dalamnya sejumlah kecil evaluator memeriksa sebuah antarmuka secara mandiri, menilai kesesuaiannya dengan seperangkat prinsip *usability* yang diakui (heuristiknya), mencatat setiap pelanggaran sebagai masalah *usability* beserta lokasi dan keparahannya, lalu menggabungkan temuan mereka menjadi satu daftar [2], [3].

Metode ini disebut *usability* "diskon" karena tidak memerlukan pengguna, laboratorium, dan hanya sedikit waktu. Itulah kekuatan sekaligus batasnya: ia menemukan banyak masalah dengan cepat, tetapi yang ditemukannya adalah masalah yang dapat diramalkan para ahli, bukan masalah yang hanya terungkap oleh pengguna sungguhan. Bab 7 menambahkan pengujian dengan pengguna; proyek yang baik memakai keduanya.

3.3 Prinsip-Prinsip Norman

The Design of Everyday Things karya Don Norman menguraikan segelintir prinsip yang menjelaskan mengapa pintu, kompor, dan layar mudah atau sulit dipakai [4]. Prinsip-prinsip itu adalah fondasi bagi setiap daftar heuristik.

3.3.1 Afordans dan penanda

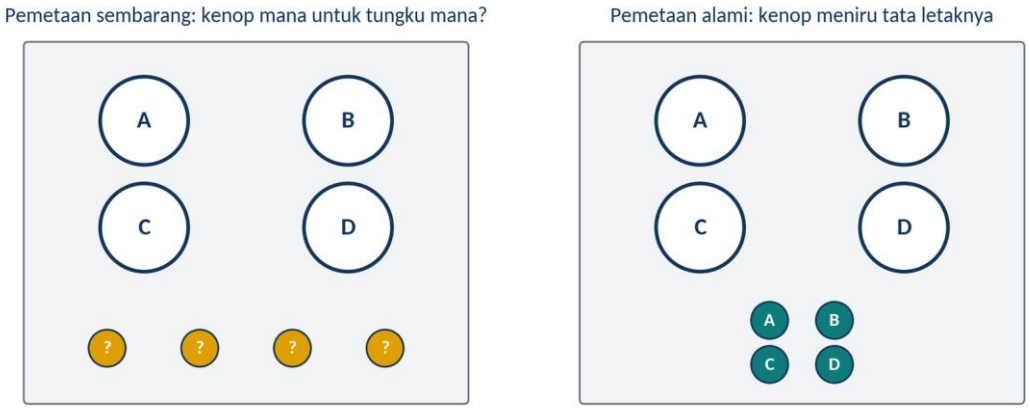
Afordans (*affordance*) adalah hubungan antara sebuah benda dan seseorang: tindakan apa yang dimungkinkan benda itu bagi orang itu. Pelat datar di pintu memungkinkan didorong; kursi memungkinkan diduduki oleh orang dewasa dan dipanjat oleh balita. *Penanda* (*signifier*) adalah sinyal yang dapat ditangkap yang memberi tahu seseorang apa yang dapat dilakukan dan di mana: pegangan di sisi tarik sebuah pintu, tampilan timbul sebuah tombol, garis bawah sebuah tautan, teks petunjuk abu-abu "Contoh: 0812 3456 7890" di kolom nomor telepon. Di layar, pada prinsipnya hampir semua hal mungkin, karena piksel mana pun dapat bereaksi terhadap sentuhan, sehingga yang menentukan adalah penandanya. Desain datar menghapus banyak penanda pada 2010-an, dan keluhan berulang bahwa pengguna "tidak melihat bahwa itu tombol" adalah keluhan tentang penanda yang hilang.

Pada halaman pesanan pertama, dua tombol kirim, *Kirim* dan *Submit*, adalah penanda yang saling bertentangan (yang mana yang harus saya tekan?), dan *dropdown* layanan menampilkan kode "CKS" tanpa tanda apa pun tentang arti pilihan-pilihannya.

3.3.2 Pemetaan

Pemetaan (*mapping*) adalah hubungan antara kontrol dan akibatnya. Pemetaan disebut *alami* bila susunan kontrol mencerminkan susunan hal yang dikendalikannya. **Gambar 3.2** memperlihatkan contoh klasik Norman, sebuah kompor: empat tungku yang tersusun persegi dan empat kenop yang berjajar memaksa juru masak menghafal atau membaca kenop mana untuk tungku mana,

sedangkan empat kenop yang disusun dalam persegi yang sama dengan tungkunya tidak memerlukan label sama sekali.



Gambar 3.2 Pemetaan pada kompor: kenop yang berjajar memberi pemetaan sembarang, kenop dalam persegi yang sama dengan tungkunya memberi pemetaan alami.

Layar penuh dengan pemetaan: penggeser volume yang naik untuk lebih keras, peta yang bergeser ke kiri ketika diseret ke kiri, denah kursi di aplikasi tiket yang menampilkan kursi pada posisi sebenarnya. Tiga ubin layanan pada halaman pesanan yang diperbaiki memetakan tiga layanan ke tiga pilihan yang terlihat lengkap dengan harganya, alih-alih menyembunyikannya di balik kode dalam sebuah daftar.

3.3.3 Batasan dan fungsi pemaksa

Batasan (*constraint*) membatasi tindakan yang mungkin dilakukan, sehingga tindakan keliru tidak dapat terjadi. Batasan fisik (colokan USB-C dapat dipasang bolak-balik, yang menghapus sebuah batasan yang dulu harus dipelajari pengguna), batasan logis (pemilih tanggal yang mengabu-abukan hari Minggu ketika toko tutup), dan batasan budaya (merah berarti berhenti) semuanya mencegah kesalahan sebelum terjadi. *Fungsi pemaksa* (*forcing function*) adalah batasan kuat yang membuat langkah berikutnya mustahil sampai langkah sebelumnya selesai.

ATM adalah contoh klasiknya. Pada mesin-mesin awal, orang menerima uang lebih dulu dan kartu belakangan. Setelah mencapai tujuannya, yaitu uang, banyak orang pergi tanpa kartunya. Para psikolog menyebutnya *post-completion error* (kesalahan pascapenyelesaian): langkah yang datang setelah tujuan utama tercapai adalah langkah yang paling sering terlupa, terutama ketika memori kerja sedang sibuk [5]. Pemecahannya adalah fungsi pemaksa: kebanyakan mesin kini mengembalikan kartu *sebelum* mengeluarkan uang, sehingga tujuan tidak dapat dicapai tanpa lebih dulu mengambil kartu. Tanpa pelatihan, tanpa papan peringatan, tanpa menyalahkan pengguna: perubahan urutan menghapus kesalahan itu.

3.3.4 Umpan balik dan model konseptual

Umpan balik (*feedback*) memberi tahu orang apa yang terjadi sebagai akibat tindakannya, dengan cepat dan jelas. Tombol yang berubah ketika ditekan, bilah kemajuan, bunyi, dan terutama konfirmasi yang menyatakan hasilnya (“Pesanan LD-0142 diterima”) menutup jurang evaluasi dari Bab 1. Umpan balik yang terlambat, samar, atau tidak ada menyebabkan tindakan diulang, dan itulah persisnya sebab pelanggan mengirim pesanan laundry dua kali.

Model konseptual adalah penjelasan tentang cara kerja sistem yang dikomunikasikan oleh desainnya. Halaman pesanan yang diperbaiki membangun model yang baik: layanan punya harga

per kilogram, beratnya adalah perkiraan yang akan dipastikan toko, dan sebuah pesanan punya nomor serta dapat diubah sampai waktu yang disebutkan. Dengan model itu pelanggan dapat meramalkan apa yang akan terjadi dan membetulkan kekeliruan tanpa perlu bertanya.

Akhirnya, *discoverability* (ketertemuan) merangkai semua prinsip itu: dapatkah seseorang mengetahui tindakan apa yang mungkin dan bagaimana melakukannya? Ketertemuan adalah hasil kerja sama penanda yang baik, pemetaan yang alami, batasan yang masuk akal, umpan balik, dan model konseptual yang jelas [4].

3.4 Tiga Keluarga Aturan

Sejumlah peneliti telah mengubah prinsip seperti prinsip Norman menjadi daftar periksa. Tiga di antaranya dipakai di mana-mana, dan ketiganya jauh lebih banyak sepakat daripada berbeda.

Delapan aturan emas desain antarmuka dari Shneiderman, yang diperhalus selama enam edisi *Designing the User Interface*, adalah: usahakan konsistensi; upayakan kebergunaan universal; berikan umpan balik yang informatif; rancang dialog yang berujung pada penyelesaian; cegah kesalahan; izinkan tindakan dibatalkan dengan mudah; biarkan pengguna memegang kendali; dan kurangi beban memori jangka pendek [6]. Aturan kedua, yang ditambahkan pada edisi-edisi belakangan, meminta perancang melayani pemula dan ahli, muda dan tua, orang dengan dan tanpa disabilitas, tema yang dikembangkan di Bab 11.

ISO 9241-110:2020 mendefinisikan tujuh *prinsip interaksi* sebagai bagian dari standar ergonomi internasional: kesesuaian dengan tugas pengguna; kejelasan diri (*self-descriptiveness*); kesesuaian dengan harapan pengguna; kemudahan dipelajari; keterkendalian; ketahanan terhadap kesalahan penggunaan; dan keterlibatan pengguna [7]. Nilainya terletak pada statusnya sebagai standar: sebuah tender sektor publik atau sebuah kontrak dapat mensyaratkan kepatuhan terhadapnya.

Sepuluh heuristik usability Nielsen diturunkan pada 1994 dari analisis faktor atas 249 masalah *usability* dan dipakai hampir tanpa perubahan sejak itu, dengan rumusan dan contoh yang diperbarui oleh Nielsen Norman Group [1]. Inilah daftar yang paling luas dipakai untuk evaluasi heuristik dan daftar yang dipakai para evaluator Profitina Laundry. **Tabel 3.1** memuat setiap heuristik dengan penjelasan singkat dan contoh dari halaman pesanan.

Tabel 3.1 Sepuluh heuristik usability Nielsen, dengan kata-kata sederhana, beserta contoh dari halaman pesanan pertama Profitina Laundry.

Kode	Heuristik	Dengan kata sederhana	Contoh pada halaman pesanan pertama
H1	*Visibility of system status* (visibilitas status sistem)	Selalu tunjukkan apa yang sedang terjadi, dengan segera	Tidak ada konfirmasi setelah mengirim; tidak ada total harga
H2	*Match between the system and the real world* (kecocokan dengan dunia nyata)	Pakai bahasa pengguna dan ikuti kebiasaan dunia nyata	Kode "CKS", "CK", "SL" alih-alih cuci, kering, setrika
H3	*User control and freedom* (kendali dan kebebasan pengguna)	Sediakan jalan keluar yang jelas: urungkan, batalkan, ubah	Pesanan tidak dapat diubah atau dibatalkan
H4	*Consistency and standards* (konsistensi dan standar)	Kata dan tindakan yang sama bermakna sama; ikuti konvensi platform	Dua tombol kirim; urutan tanggal bulan/hari
H5	*Error prevention* (pencegahan kesalahan)	Cegah masalah, jangan hanya melaporkannya	Tanggal lampau atau hari Minggu dapat dipilih
H6	*Recognition rather than recall* (mengenali, bukan mengingat)	Buat opsi terlihat; jangan paksa pengguna mengingat	Harga ada di halaman lain
H7	*Flexibility and efficiency of use* (fleksibilitas dan efisiensi)	Jalan pintas bagi pengguna tetap, tak terlihat oleh pemula	Tidak ada "ulangi pesanan terakhir"
H8	*Aesthetic and minimalist design* (desain estetis dan minimalis)	Tampilkan hanya yang relevan; setiap elemen tambahan bersaing	Promo beranimasi mendorong formulir ke bawah
H9	*Help users recognize, diagnose, and recover from errors* (bantu pulih dari kesalahan)	Pesan galat dengan bahasa sederhana: apa yang terjadi dan apa yang harus dilakukan	"Error 422"
H10	*Help and documentation* (bantuan dan dokumentasi)	Bantuan yang mudah ditemukan dan terfokus pada tugas	Tidak ada penjelasan tentang penimbangan atau minimal 3 kg

Ketiga keluarga itu sejalan erat. "Berikan umpan balik yang informatif" dari Shneiderman, "kejelasan diri" dari ISO, dan H1 dari Nielsen adalah satu gagasan; "izinkan tindakan dibatalkan", "keterkendalian", dan H3 adalah gagasan lain; "cegah kesalahan", "ketahanan terhadap kesalahan penggunaan", dan H5 gagasan ketiga; "kurangi beban memori jangka pendek" dan H6 gagasan keempat. Pilihlah satu daftar untuk sebuah evaluasi, agar semua evaluator berbicara dengan bahasa yang sama, dan ketahuilah bahwa daftar lain tidak akan menemukan sesuatu yang secara mendasar berbeda.

3.5 Prinsip Bertemu Bahasa Desain 2025–2026

Bahasa desain berganti setiap beberapa tahun; prinsip tidak. Pembaruan-pembaruan pada 2025 memperlihatkan hal ini dengan baik.

Liquid Glass. Bahasa desain Apple untuk iOS 26 dan sistem-sistem saudaranya, yang diperkenalkan pada 9 Juni 2025, adalah material tembus pandang yang "memantulkan dan membiaskan sekelilingnya" dan berubah mengikuti konten di bawahnya [8]. Material ini indah dan menjadikan konten, bukan kontrol, pusat perhatian, yang melayani prinsip desain minimalis (H8). Ia juga meregangkan prinsip lain: teks di atas lapisan tembus pandang di depan foto yang ramai dapat kehilangan kontras, padahal kontraslah yang membuat teks dapat ditangkap mata sama sekali (Bab 2). Setelah kritik tentang keterbacaan, terutama dari pengguna lanjut usia, Apple menambahkan sebuah pengaturan di iOS 26.1, *Settings › Display and Brightness › Liquid Glass*, yang menawarkan pilihan *Clear* dan *Tinted* yang lebih pekat [9]. Peristiwa ini sendiri adalah

pelajaran kecil: ketika sebuah gaya berbenturan dengan sebuah prinsip, pada akhirnya prinsiplah yang menang, dan pemecahannya adalah memberi pengguna kendali (H3, H7).

Material 3 Expressive. Pembaruan Material Design dari Google pada 2025 memakai warna, bentuk, ukuran, dan gerak yang lebih kuat untuk menonjolkan aksi utama. Google melaporkan 46 studi dengan lebih dari 18.000 partisipan dan elemen yang ditemukan hingga empat kali lebih cepat pada tata letak ekspresif [10]. Dalam bahasa bab ini, pembaruan itu memperkuat penanda dan hierarki visual, yaitu efek praatensi dari Bab 2 yang diterapkan secara sistematis.

Fluent 2. Design system Microsoft untuk Windows dan Microsoft 365 menekankan konsistensi lintas perangkat dan produk, dengan komponen dan *token* bersama [11]. Ia melayani H4 di keluarga aplikasi yang sangat besar.

Bagi perancang di Indonesia, tempat sebagian besar pengguna memakai ponsel Android dan sebagian kecil yang tetap berarti memakai iPhone (Bab 1), kesimpulan praktisnya adalah mengikuti konvensi terkini dari platformnya (H4: pengguna berharap sebuah layar berperilaku seperti aplikasi lain di ponselnya) sambil memeriksa setiap pilihan gaya terhadap prinsip-prinsip itu, terutama kontras dan ukuran sasaran bagi pengguna lanjut usia.

> **PERINGATAN:** *Design system* tidak menjamin *usability*. Setiap komponen Material, pedoman Apple, atau Fluent 2 dapat dirakit menjadi halaman yang membingungkan. Heuristik adalah pemeriksaan bahwa hasil rakitannya tetap masuk akal.

3.6 Evaluasi Heuristik Langkah demi Langkah

3.6.1 Prosedurnya

Evaluasi heuristik terdiri atas lima langkah [2], [3].

1. **Persiapan.** Pilih heuristiknya (biasanya sepuluh heuristik Nielsen), layar atau tugas yang dicakup, dan templat pencatatan bersama: masalah, lokasi, heuristik, keparahan, dan usulan.
2. **Evaluasi mandiri.** Setiap evaluator menelusuri antarmuka setidaknya dua kali, pertama untuk merasakan alurnya, lalu layar demi layar terhadap heuristik, dan mencatat setiap pelanggaran. Kemandirian itu penting: evaluator yang saling berbicara akan menemukan masalah yang sama.
3. **Pembahasan dan penggabungan.** Para evaluator bertemu, membacakan temuan mereka, dan memutuskan temuan mana yang menggambarkan masalah yang sama. Setiap masalah yang berbeda diberi pengenal.
4. **Penilaian keparahan.** Setiap evaluator menilai setiap masalah yang ditemukannya (atau, lebih baik, setiap masalah hasil penggabungan) dengan sebuah skala keparahan.
5. **Pelaporan dan prioritas.** Masalah diurutkan menurut keparahan dan diubah menjadi daftar pekerjaan bagi pengembang.

Skala keparahan Nielsen, yang tampak pada **Tabel 3.2**, menggabungkan seberapa sering sebuah masalah terjadi, seberapa parah ia memengaruhi pengguna yang mengalaminya, dan apakah ia terus mengganggu setelah pengguna mengetahuinya [12].

Tabel 3.2 Skala keparahan Nielsen untuk masalah usability.

Nilai	Arti	Tindakan
0	Bukan masalah *usability*	Tidak ada
1	Masalah kosmetik	Perbaiki bila ada waktu
2	Masalah *usability* kecil	Perbaiki dengan prioritas rendah
3	Masalah *usability* besar	Perbaiki dengan prioritas tinggi
4	Bencana *usability*	Perbaiki sebelum rilis

3.6.2 Berapa banyak evaluator?

Tidak ada evaluator tunggal yang menemukan segalanya. Nielsen dan Landauer memodelkan penemuan masalah sebagai berikut: jika setiap evaluator secara mandiri menemukan bagian λ dari N masalah dalam sebuah antarmuka, maka i evaluator bersama-sama menemukan

$$\text{Found}(i) = N(1 - (1 - \lambda)^i) \tag{3.1}$$

masalah [13]. Penalarannya singkat. Sebuah masalah terlewat oleh satu evaluator dengan peluang $1 - \lambda$, dan terlewat oleh semua i evaluator yang mandiri dengan peluang $(1 - \lambda)^i$; masalah itu ditemukan oleh setidaknya satu evaluator dengan peluang $1 - (1 - \lambda)^i$, dan mengalikannya dengan N memberikan banyaknya masalah yang diharapkan ditemukan. Di seluruh studi yang dianalisis Nielsen dan Landauer, λ rata-rata sekitar 0,31 [13].

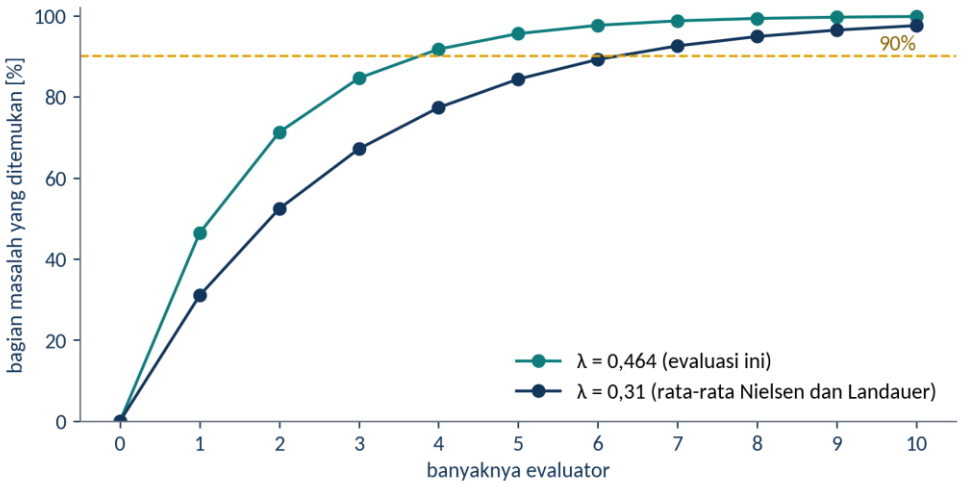
Dalam praktik N tidak diketahui, tetapi λ dapat diperkirakan dari evaluasi itu sendiri. Jika n evaluator membuat total F temuan tentang P masalah yang berbeda, rata-rata bagian yang ditemukan per evaluator adalah

$$\hat{\lambda} = \frac{F}{n \cdot P} \tag{3.2}$$

Perkiraan ini memperlakukan masalah yang ditemukan sebagai seluruh masalah, sehingga agak terlalu optimistis; dengan sedikit evaluator, masalah yang tidak ditemukan siapa pun tidak terlihat. Menyelesaikan $1 - (1 - \lambda)^i \geq t$ untuk i memberikan banyaknya evaluator yang diperlukan untuk menemukan bagian sasaran t:

$$i \geq \frac{\ln(1 - t)}{\ln(1 - \lambda)} \tag{3.3}$$

(kedua logaritma bernilai negatif, sehingga arah ketaksamaannya tetap). Dengan $\lambda = 0.31$ dan $t = 0.75$, $i \geq \ln 0.25 / \ln 0.69 = 3.74$, jadi empat evaluator; untuk $t = 0.90$ kita memerlukan tujuh. **Gambar 3.3** memperlihatkan kurvanya untuk rata-rata Nielsen dan Landauer dan untuk evaluasi Profitina Laundry. Melandainya kurva itulah alasan di balik saran umum tiga hingga lima evaluator: setiap evaluator tambahan menemukan lebih sedikit masalah baru daripada evaluator sebelumnya [3].



Gambar 3.3 Bagian masalah yang ditemukan oleh i evaluator, dari Persamaan (3.1), untuk $\lambda = 0.464$ (evaluasi ini) dan $\lambda = 0.31$ (rata-rata Nielsen dan Landauer).

Satu kehati-hatian lagi datang dari riset tentang *evaluator effect* (efek evaluator): evaluator yang berbeda tidak hanya menemukan masalah yang berbeda, tetapi juga menilai masalah yang sama secara berbeda, kadang berselisih dua tingkat keparahan [14]. Karena itu menggabungkan dan merata-ratakan beberapa penilaian mandiri bukan formalitas; itulah yang membuat daftarnya dapat dipercaya.

3.6.3 Evaluasi heuristik dan kerabatnya

Evaluasi heuristik adalah salah satu dari beberapa *metode inspeksi*, yang di dalamnya para ahli, bukan pengguna, memeriksa sebuah desain. Dua kerabatnya patut diketahui. *Cognitive walkthrough* (penelusuran kognitif) mengikuti satu tugas konkret langkah demi langkah, misalnya "seorang pelanggan baru memesan cuci, kering, dan setrika untuk hari Sabtu", dan di setiap langkah mengajukan empat pertanyaan: akankah pengguna berusaha mencapai akibat yang benar, akankah ia menyadari bahwa tindakan yang benar tersedia, akankah ia mengaitkan tindakan itu dengan akibat yang diinginkannya, dan akankah ia melihat bahwa kemajuan sedang terjadi? Metode ini lebih lambat daripada evaluasi heuristik, tetapi lebih baik dalam menemukan masalah kemudahan dipelajari bagi pengguna pertama kali, yaitu orang-orang yang paling perlu dilayani oleh kios di Bab 2 dan halaman pesanan laundry ini. *Guideline review* (tinjauan pedoman) memeriksa sebuah desain terhadap daftar panjang aturan khusus, seperti pedoman sebuah platform atau kriteria keberhasilan WCAG 2.2; tinjauan ini teliti tetapi mekanis, dan yang ditemukannya adalah pelanggaran aturan, bukan masalah penggunaan.

Ketiga metode itu saling melengkapi. Urutan yang praktis untuk proyek kecil adalah evaluasi heuristik atas seluruh antarmuka lebih dulu, penelusuran kognitif atas dua atau tiga tugas terpenting berikutnya, dan tinjauan pedoman untuk aksesibilitas (Bab 11) sebelum rilis, disusul pengujian dengan pengguna sungguhan (Bab 7). Masing-masing memakan waktu berjam-jam, bukan berminggu-minggu, dan bersama-sama ketiganya menghapus sebagian besar masalah yang kalau tidak akan sampai ke pelanggan.

3.7 Membangun Pembuat Laporan

Pembahasan pada langkah 3 adalah percakapan dan tetap menjadi pekerjaan manusia: hanya manusia yang dapat memutuskan bahwa "halaman memuat ulang setelah mengirim" dan "saya tidak tahu apakah pesanan saya sudah terkirim" adalah masalah yang sama. Semua sesudahnya,

yaitu menghitung, merata-ratakan, mengurutkan, dan memperkirakan, adalah pembukuan yang lebih baik dikerjakan program. Program di subbab ini membaca dua berkas lalu mencetak laporannya.

3.7.1 Datanya

Kode 3.1 adalah daftar masalah berbeda yang disepakati dalam pembahasan, dengan heuristik yang dilanggar masing-masing dan tempat terjadinya. Deskripsi yang mengandung koma diberi tanda kutip, sebagaimana lazimnya dalam CSV. (Isi berkas data ditulis dalam bahasa Inggris agar sama dengan edisi bahasa Inggris dan keluaran programnya.)

Kode 3.1 `data/problems.csv`: empat belas masalah berbeda pada halaman pesanan pertama.

```
id,heuristic,location,problem
P01,H1,Send button,No confirmation after sending; the page just reloads
P02,H2,Service list,"Codes CKS, CK and SL instead of wash, dry, iron"
P03,H5,Pickup date,Accepts a past date and Sundays when the shop is closed
P04,H9,Error message,"Shows ""Error 422"" with no cause or remedy"
P05,H6,Prices,Price list is on another page; users must remember it
P06,H4,Buttons,"Two send buttons, ""Kirim"" and ""Submit"", in different styles"
P07,H8,Promo banner,Animated banner pushes the form below the fold
P08,H3,After sending,No way to change or cancel an order
P09,H7,Regular customers,No way to repeat the last order
P10,H10,Weight,No explanation of weighing or the 3 kg minimum
P11,H1,Total price,Total price is not shown before sending
P12,H5,Phone number,Accepts letters; no example of the expected format
P13,H4,Date format,Month/day order unlike every other Indonesian form
P14,H8,Small print,Light grey 11 px text is hard to read
```

Kode 3.2 mencatat siapa menemukan masalah apa dan keparahan yang diberikannya. Evaluator E1 menemukan delapan masalah, E2 enam, E3 lima, dan E4 tujuh.

Kode 3.2 `data/findings.csv`: 26 temuan dari empat evaluator, masing-masing dengan keparahan 0 hingga 4.

```
evaluator,id,severity
E1,P01,4
E1,P02,3
E1,P03,3
E1,P04,3
E1,P08,3
E1,P11,4
E1,P12,2
E1,P14,2
E2,P01,3
E2,P02,2
E2,P05,2
E2,P07,2
E2,P11,3
E2,P13,2
E3,P01,4
E3,P03,2
E3,P06,1
E3,P09,2
E3,P11,3
E4,P02,3
E4,P04,2
E4,P08,3
E4,P10,1
E4,P11,4
E4,P12,2
E4,P14,3
```

3.7.2 Programnya

Kode 3.3 adalah versi Python-nya.

Kode 3.3 `heuristics.py`: menggabungkan temuan, mengurutkan masalah, dan memperkirakan banyaknya evaluator yang diperlukan.

```
"""Gabungkan temuan evaluasi heuristik menjadi satu laporan berperingkat."""
import csv
import math
import sys


def read_csv(path):
    with open(path, newline="", encoding="utf-8") as f:
        return list(csv.DictReader(f))


def merge(problems, findings):
    """Per masalah: siapa yang menemukan dan dengan keparahan berapa (0-4)."""
    merged = {p["id"]: dict(p, severities=[]) for p in problems}
    for f in findings:
        merged[f["id"]]["severities"].append(int(f["severity"]))
    return merged


def rank(merged):
    """Paling parah dulu; lalu yang ditemukan lebih banyak evaluator; lalu id."""
    def key(p):
        s = p["severities"]
        return (-sum(s) / len(s), -len(s), p["id"])
    return sorted((p for p in merged.values() if p["severities"]), key=key)


def found_share(lam, i):
    """Nielsen dan Landauer: bagian masalah yang ditemukan i evaluator."""
    return 1 - (1 - lam) ** i


def evaluators_needed(lam, target):
    """Banyak evaluator terkecil yang prediksi bagiannya >= target."""
    return math.ceil(math.log(1 - target) / math.log(1 - lam))
```

Kode 3.3 (Lanjutan) *heuristics.py: menggabungkan temuan, mengurutkan masalah, dan memperkirakan banyaknya evaluator yang diperlukan.*

```
def main():
    folder = sys.argv[1] if len(sys.argv) > 1 else "data"
    problems = read_csv(f"{folder}/problems.csv")
    findings = read_csv(f"{folder}/findings.csv")
    evaluators = sorted({f["evaluator"] for f in findings})
    ranked = rank(merge(problems, findings))
    n = len(evaluators)
    print(f"Heuristic evaluation: {n} evaluators, {len(ranked)} problems, "
          f"{len(findings)} findings")
    print("rank id   heur found mean max location")
    for r, p in enumerate(ranked, 1):
        s = p["severities"]
        print(f"{r:4} {p['id']:4} {p['heuristic']:4} {len(s)}/{n}  "
              f"{sum(s) / len(s):4.1f} {max(s):3}  {p['location']}")
    urgent = [p["id"] for p in ranked
              if sum(p["severities"]) / len(p["severities"]) >= 3]
    print("Fix before launch (mean severity 3 or more): " + ", ".join(urgent))
    per_h = {}
    for p in ranked:
        per_h[p["heuristic"]] = per_h.get(p["heuristic"], 0) + 1
    order = sorted(per_h, key=lambda h: int(h[1:]))
    print("Problems per heuristic: "
          + ", ".join(f"{h} {per_h[h]}" for h in order))
    lam = len(findings) / (n * len(ranked))
    print(f"Each evaluator found {100 * lam:.1f}% of the problems "
          f"on average (lambda).")
    print("Predicted share found by i evaluators:")
    print("  " + "  ".join(f"{i}: {100 * found_share(lam, i):.0f}%"
                           for i in range(1, 7)))
    print(f"Evaluators needed for 90%: {evaluators_needed(lam, 0.90)}")


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

`merge` membuat, untuk setiap masalah, salinan barisnya dengan daftar keparahan yang masih kosong, lalu menambahkan keparahan setiap temuan ke masalahnya. `rank` mengurutkan berdasarkan sebuah *tuple*: negatif rata-rata keparahan lebih dulu (sehingga rata-rata tertinggi berada paling atas), lalu negatif banyaknya evaluator yang menemukannya, lalu pengenalnya, sehingga urutannya selalu sama. `found_share` adalah Persamaan (3.1) dibagi N, dan `evaluators_needed` adalah Persamaan (3.3) yang dibulatkan ke atas dengan `math.ceil`. `main` mencetak tabel berperingkat, daftar masalah dengan rata-rata keparahan 3 atau lebih, banyaknya masalah per heuristik, perkiraan $\hat{\lambda}$ dari Persamaan (3.2), prediksi bagian yang ditemukan oleh satu hingga enam evaluator, dan banyaknya evaluator yang diperlukan untuk 90%.

Kode 3.4 adalah versi Java, yang mencetak laporan yang sama.

Kode 3.4 `Heuristics.java`: laporan yang sama dalam Java.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;
import java.util.TreeSet;

/** Gabungkan temuan evaluasi heuristik menjadi satu laporan berperingkat. */
public class Heuristics {
    record Problem(String id, String heuristic, String location,
                   List<Integer> severities) {
        double mean() {
            return severities.stream().mapToInt(i -> i).average().orElse(0);
        }
    }

    /** Pecah satu baris CSV; koma di dalam "teks berkutip" tidak memecah. */
    static List<String> cells(String line) {
        List<String> out = new ArrayList<>();
        StringBuilder cell = new StringBuilder();
        boolean quoted = false;
        for (int i = 0; i < line.length(); i++) {
            char c = line.charAt(i);
            if (c == '"' && quoted && i + 1 < line.length()
                    && line.charAt(i + 1) == '"') {
                cell.append('"');                // "" di dalam kutip = satu "
                i++;
            } else if (c == '"') {
                quoted = !quoted;
            } else if (c == ',' && !quoted) {
                out.add(cell.toString());
                cell.setLength(0);
            } else {
                cell.append(c);
            }
        }
        out.add(cell.toString());
        return out;
    }
```

Kode 3.4 (Lanjutan) *Heuristics.java: laporan yang sama dalam Java.*

```
    static List<List<String>> readCsv(String path) throws IOException {
        List<List<String>> rows = new ArrayList<>();
        List<String> lines = Files.readAllLines(Path.of(path));
        for (String line : lines.subList(1, lines.size())) rows.add(cells(line));
        return rows;
    }

    /** Nielsen dan Landauer: bagian masalah yang ditemukan i evaluator. */
    static double foundShare(double lambda, int i) {
        return 1 - Math.pow(1 - lambda, i);
    }

    /** Banyak evaluator terkecil yang prediksi bagiannya >= target. */
    static int evaluatorsNeeded(double lambda, double target) {
        return (int) Math.ceil(Math.log(1 - target) / Math.log(1 - lambda));
    }

    /** Per masalah: siapa penemunya dan keparahannya; paling parah dulu. */
    static List<Problem> rank(List<List<String>> problems,
                              List<List<String>> findings) {
        Map<String, Problem> merged = new LinkedHashMap<>();
        for (var p : problems) {
            merged.put(p.get(0), new Problem(p.get(0), p.get(1), p.get(2),
                                             new ArrayList<>()));
        }
        for (var f : findings) {
            merged.get(f.get(1)).severities().add(Integer.parseInt(f.get(2)));
        }
        List<Problem> ranked = new ArrayList<>();
        for (Problem p : merged.values()) {
            if (!p.severities().isEmpty()) ranked.add(p);
        }
        ranked.sort(Comparator.comparingDouble((Problem p) -> -p.mean())
                .thenComparingInt(p -> -p.severities().size())
                .thenComparing(Problem::id));
        return ranked;
    }
```

Kode 3.4 (Lanjutan) *Heuristics.java: laporan yang sama dalam Java.*

```
    public static void main(String[] args) throws IOException {
        String folder = args.length > 0 ? args[0] : "data";
        var findings = readCsv(folder + "/findings.csv");
        var ranked = rank(readCsv(folder + "/problems.csv"), findings);
        TreeSet<String> evaluators = new TreeSet<>();
        for (var f : findings) evaluators.add(f.get(0));
        int n = evaluators.size();
        System.out.printf("Heuristic evaluation: %d evaluators, %d problems, "
                          + "%d findings%n", n, ranked.size(), findings.size());
        System.out.println("rank id   heur found mean max location");
        List<String> urgent = new ArrayList<>();
        Map<Integer, Integer> perH = new TreeMap<>();
        for (int r = 0; r < ranked.size(); r++) {
            Problem p = ranked.get(r);
            int max = p.severities().stream().mapToInt(i -> i).max().orElse(0);
            System.out.printf("%4d %-4s %-4s %d/%d  %4.1f %3d  %s%n", r + 1,
                              p.id(), p.heuristic(), p.severities().size(), n,
                              p.mean(), max, p.location());
            if (p.mean() >= 3) urgent.add(p.id());
            perH.merge(Integer.parseInt(p.heuristic().substring(1)), 1,
                       Integer::sum);
        }
        System.out.println("Fix before launch (mean severity 3 or more): "
                           + String.join(", ", urgent));
        List<String> counts = new ArrayList<>();
        perH.forEach((h, c) -> counts.add("H" + h + " " + c));
        System.out.println("Problems per heuristic: " + String.join(", ", counts));
        double lambda = (double) findings.size() / (n * ranked.size());
        System.out.printf("Each evaluator found %.1f%% of the problems "
                          + "on average (lambda).%n", 100 * lambda);
        System.out.println("Predicted share found by i evaluators:");
        List<String> shares = new ArrayList<>();
        for (int i = 1; i <= 6; i++) {
            shares.add(String.format("%d: %.0f%%", i, 100 * foundShare(lambda, i)));
        }
        System.out.println("  " + String.join("  ", shares));
        System.out.println("Evaluators needed for 90%: "
                           + evaluatorsNeeded(lambda, 0.90));
    }
}
```

Dua hal pada versi Java patut dicatat. `record` adalah kelas ringkas yang *field*-nya ditetapkan saat konstruksi; `Problem` menyimpan pengenal, heuristik, lokasi, dan daftar keparahan, serta menambahkan metode `mean`. Karena deskripsinya mengandung koma di dalam tanda kutip, memecah di setiap koma akan merusak baris, sehingga metode `cells` menelusuri setiap baris karakter demi karakter, membalik sebuah penanda ketika menjumpai tanda kutip dan memperlakukan dua tanda kutip di dalam kutipan sebagai satu tanda kutip harfiah, sesuai aturan CSV. Pengurutannya memakai rangkaian *comparator* yang mencerminkan *tuple* di Python.

3.8 Menjalankannya

```
$ python3 heuristics.py
$ java -cp out Heuristics
```

```
Heuristic evaluation: 4 evaluators, 14 problems, 26 findings
rank id   heur found mean max location
   1 P01  H1   3/4   3.7   4  Send button
   2 P11  H1   4/4   3.5   4  Total price
   3 P08  H3   2/4   3.0   3  After sending
   4 P02  H2   3/4   2.7   3  Service list
   5 P03  H5   2/4   2.5   3  Pickup date
   6 P04  H9   2/4   2.5   3  Error message
   7 P14  H8   2/4   2.5   3  Small print
   8 P12  H5   2/4   2.0   2  Phone number
   9 P05  H6   1/4   2.0   2  Prices
  10 P07  H8   1/4   2.0   2  Promo banner
  11 P09  H7   1/4   2.0   2  Regular customers
  12 P13  H4   1/4   2.0   2  Date format
  13 P06  H4   1/4   1.0   1  Buttons
  14 P10  H10  1/4   1.0   1  Weight
Fix before launch (mean severity 3 or more): P01, P11, P08
Problems per heuristic: H1 2, H2 1, H3 1, H4 2, H5 2, H6 1, H7 1, H8 2, H9 1, H10 1
Each evaluator found 46.4% of the problems on average (lambda).
Predicted share found by i evaluators:
  1: 46%  2: 71%  3: 85%  4: 92%  5: 96%  6: 98%
Evaluators needed for 90%: 4
```

Laporan itu memberi tahu pengembang dari mana harus mulai. Tiga masalah punya rata-rata keparahan 3 atau lebih: P01, tidak ada konfirmasi setelah mengirim (ditemukan tiga dari empat evaluator, rata-rata 3,7); P11, total harga yang tidak ditampilkan (ditemukan keempatnya); dan P08, tidak ada cara mengubah atau membatalkan pesanan. Inilah persisnya masalah yang menimbulkan pesanan ganda dan pertanyaan lewat WhatsApp pada minggu pertama. Berikutnya menyusul kode dalam daftar layanan (P02), tanggal yang mustahil (P03), pesan "Error 422" (P04), dan cetakan kecil berwarna abu-abu muda (P14). Hanya satu masalah, total harga yang tidak ada, yang ditemukan semua evaluator, dan tujuh masalah hanya ditemukan oleh satu evaluator: peragaan yang gamblang mengapa diperlukan beberapa evaluator yang mandiri.

Perkiraan $\hat{\lambda} = 26/(4 \times 14) = 0.464$ lebih tinggi daripada rata-rata Nielsen dan Landauer sebesar 0,31, yang masuk akal untuk halaman kecil dengan masalah yang mencolok; perkiraan itu memprediksi bahwa empat evaluator menemukan sekitar 92% masalah, dan bahwa empat evaluator diperlukan untuk 90%. Karena perkiraan itu memperlakukan keempat belas masalah yang ditemukan sebagai seluruh masalah yang ada, bagian yang sebenarnya agak lebih rendah, dan evaluator kelima akan menjadi investasi yang bijak untuk produk yang lebih besar.

Pengembang memperbaiki masalah-masalah itu sesuai urutan laporan. Halaman kanan pada **Gambar 3.1** memperlihatkan hasilnya: layanan menjadi ubin yang terlihat dengan harga dalam bahasa pelanggan (P02, P05), berat dan batas minimalnya dijelaskan di tempat pengisiannya (P10), hari penjemputan berupa nama hari lengkap dengan jam buka (P03, P13), kolom telepon menampilkan contoh (P12), perkiraan total ditampilkan sebelum mengirim (P11), ada satu tombol kirim yang berlabel jelas (P06), dan setelah mengirim halaman menyebutkan nomor pesanan, hari penjemputan, dan sampai kapan pesanan dapat diubah (P01, P08). Promo dan cetakan kecil abu-abu dihilangkan (P07, P14). Bab 7 akan menunjukkan cara memeriksa bersama pelanggan sungguhan bahwa halaman baru itu memang lebih baik; evaluasi heuristik memastikan masalah-masalah yang mencolok sudah hilang sebelum mereka ditanya.

3.9 Mengujinya

Kode 3.5 memeriksa model Persamaan (3.1) dan Persamaan (3.3) pada kasus yang mudah dihitung dengan tangan ($\lambda = 0.5$: dua evaluator menemukan $1 - 0.25 = 75\%$; untuk 90%,

ln0.1/ln0.5 = 3.32, jadi empat) dan memeriksa bahwa pengurutan menaruh masalah yang lebih parah lebih dulu.

Kode 3.5 `test_heuristics.py`: uji model dan pengurutan.

```
"""Uji laporan evaluasi: python -m unittest test_heuristics.py"""
import unittest
from heuristics import merge, rank, found_share, evaluators_needed


class HeuristicsTests(unittest.TestCase):
    def test_found_share(self):
        self.assertAlmostEqual(found_share(0.5, 2), 0.75)

    def test_evaluators_needed(self):
        self.assertEqual(evaluators_needed(0.5, 0.90), 4)

    def test_rank_puts_the_more_severe_problem_first(self):
        problems = [{"id": "A"}, {"id": "B"}]
        findings = [{"id": "A", "severity": "2"}, {"id": "B", "severity": "4"}]
        self.assertEqual([p["id"] for p in rank(merge(problems, findings))],
                         ["B", "A"])


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

Kode 3.6 menambahkan dua pemeriksaan pemecahan CSV, yang di Python dikerjakan oleh modul `csv`.

Kode 3.6 `HeuristicsTest.java`: pemeriksaan yang sama dalam Java, ditambah pemecahan CSV.

```
import java.util.List;

/** Uji laporan evaluasi heuristik: java -cp out HeuristicsTest */
public class HeuristicsTest {
    static int passed = 0;

    static void expect(boolean ok, String what) {
        if (!ok) throw new AssertionError(what);
        passed++;
    }

    public static void main(String[] args) throws Exception {
        expect(Heuristics.cells("P02,H2,x,\"a, b\"").get(3).equals("a, b"),
               "a quoted comma does not split");
        expect(Heuristics.cells("\"say \"\"hi\"\"\"").get(0).equals("say \"hi\""),
               "doubled quotes");
        expect(Math.abs(Heuristics.foundShare(0.5, 2) - 0.75) < 1e-12,
               "two evaluators at lambda 0.5 find 75%");
        expect(Heuristics.evaluatorsNeeded(0.5, 0.90) == 4, "0.5 needs 4 for 90%");
        var ranked = Heuristics.rank(
            List.of(List.of("A", "H1", "a"), List.of("B", "H2", "b")),
            List.of(List.of("E1", "A", "2"), List.of("E1", "B", "4")));
        expect(ranked.get(0).id().equals("B"), "the more severe problem first");
        System.out.println("All " + passed + " checks passed.");
    }
}
```

```
$ python3 -m unittest test_heuristics.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
$ java -cp out HeuristicsTest
All 5 checks passed.
```

3.10 Memakainya Sungguh-sungguh

Untuk melaksanakan evaluasi heuristik atas produk Anda sendiri, entah halaman pesanan warung, kios puskesmas, atau aplikasi kampus, Anda memerlukan tiga hingga lima evaluator, masing-masing dua jam, dan templat pada **Tabel 3.3**. Evaluator tidak harus perancang profesional; mahasiswa yang sudah membaca bab ini menemukan sebagian besar masalah yang ditemukan seorang profesional, walaupun profesional yang memahami ranahnya menemukan lebih banyak [2].

Tabel 3.3 Templat pencatatan untuk evaluasi heuristik.

Kolom	Yang ditulis	Contoh
Evaluator	Inisial atau kode	E2
Lokasi	Layar dan elemen	Halaman pesanan, kolom tanggal jemput
Masalah	Apa yang salah, hasil pengamatan, dalam satu kalimat	Menerima hari Minggu, padahal toko tutup
Heuristik	Kode dari **Tabel 3.1**	H5
Keparahan	0 sampai 4 dari **Tabel 3.2**	3
Usulan	Kemungkinan perbaikan, tidak wajib	Tawarkan hanya hari buka

Setelah pembahasan, tuliskan masalah yang disepakati ke `problems.csv` dan penilaian setiap evaluator ke `findings.csv`, lalu jalankan programnya. Tiga kebiasaan membuat laporan lebih berguna. Tuliskan masalah sebagai hasil pengamatan, bukan pemecahan (“tidak ada konfirmasi setelah mengirim”, bukan “tambahkan *toast*”), agar pengembang dapat memilih perbaikan terbaik. Catat juga temuan yang positif, agar bagian yang baik tidak ikut “diperbaiki”. Dan ulangi evaluasi setelah perbaikan, karena perbaikan memunculkan masalah baru.

Untuk fitur AI, metode yang sama berlaku dengan daftar yang diperluas. Pedoman interaksi manusia-AI dari Amershi dkk. (Bab 1) dapat dipakai sebagai heuristik tambahan, seperti “jelaskan apa yang dapat dilakukan sistem” dan “dukung koreksi yang efisien”; Bab 13 melakukan persis hal itu.

3.11 IMK dalam Praktik: Profitina

Produk Profitina yang sesungguhnya memperlihatkan beberapa heuristik sebagai fitur yang disengaja. *Visibilitas status sistem* (H1): mengirim tulisan di komunitas bersifat optimistis, sehingga kotak tulis langsung kosong dan tulisan segera muncul, dan jika pengiriman gagal, teksnya dikembalikan dengan pesan yang jelas alih-alih hilang. *Pencegahan kesalahan* (H5) dan *membantu pulih dari kesalahan* (H9) tampak pada Link Safety: setiap tautan yang dikirim di komunitas disaring, dan tautan yang berisiko ditandai beserta penjelasan alasannya, yang mencegah kesalahan mengeklik tautan penipuan dan memberi tahu pengguna apa yang sebaiknya dilakukan. *Kendali dan kebebasan pengguna* (H3) dibangun ke dalam layar akun yang diwajibkan UU PDP, tempat ekspor data, penggantian kata sandi, dan penghapusan akun secara penuh menjadi tindakan swalayan. Dan *kecocokan dengan dunia nyata* (H2) adalah alasan dasbor

menjelaskan setiap sinyal *AI visibility* dengan bahasa sederhana, bukan dengan kosakata teknis data terstruktur dan *crawler*.

Pembuatan kanal memperlihatkan *pencegahan kesalahan* melalui batasan: seorang pengguna boleh membuat paling banyak dua kanal sehari dan memiliki lima kanal aktif, dan aturan itu dinyatakan di dekat tombolnya, sehingga batas itu menjadi batasan yang diketahui, bukan kejutan.

3.12 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

1. **Mengevaluasi bersama-sama sejak awal.** Evaluator yang bekerja berkelompok menemukan lebih sedikit masalah. Evaluasilah secara mandiri, lalu gabungkan.
2. **Memakai satu evaluator.** Satu orang menemukan kira-kira sepertiga hingga separuh masalah. Pakailah tiga hingga lima.
3. **Menuliskan pemecahan alih-alih masalah.** "Tambahkan dialog konfirmasi" menyembunyikan masalahnya dan memutuskan desain terlalu dini. Tuliskan apa yang salah dan di mana.
4. **Menilai keparahan seorang diri.** Penilaian berbeda antarevaluator; rata-ratakan beberapa penilaian mandiri.
5. **Menganggap evaluasi heuristik sebagai uji pengguna.** Evaluasi ini menemukan apa yang dapat diramalkan para ahli; ujilah juga dengan pengguna sungguhan (Bab 7).
6. **Mengikuti *design system* secara membabi buta.** Gaya seperti kaca tembus pandang dapat menurunkan kontras; periksa setiap pilihan terhadap prinsip-prinsipnya.
7. **Memaksakan setiap masalah ke dalam sebuah heuristik.** Sebagian masalah tidak cocok dengan heuristik mana pun; catat saja sebagai "lainnya".
8. **Berhenti setelah satu putaran.** Perbaikan menimbulkan masalah baru; evaluasi lagi setelah perubahan.

3.13 Ringkasan Bab

- Prinsip bersifat umum (mengapa), pedoman khusus untuk sebuah platform (apa), heuristik adalah aturan praktis untuk evaluasi (periksa).
- Prinsip Norman: afordans dan penanda, pemetaan alami (**Gambar 3.2**), batasan dan fungsi pemaksa (ATM mengembalikan kartu sebelum uang), umpan balik dan model konseptual yang jelas, yang bersama-sama menghasilkan ketertemuan.
- Delapan aturan emas Shneiderman, tujuh prinsip interaksi ISO 9241-110:2020, dan sepuluh heuristik Nielsen (**Tabel 3.1**) sebagian besar sejalan.
- Liquid Glass, Material 3 Expressive, dan Fluent 2 menerapkan prinsip-prinsip itu dan kadang meregangkannya; opsi *Tinted* di iOS 26.1 memulihkan keterbacaan dengan memberi pengguna kendali.
- Evaluasi heuristik: evaluasi mandiri, pembahasan, penilaian keparahan 0 sampai 4 (**Tabel 3.2**), laporan berperingkat. Nielsen–Landauer: i evaluator menemukan $1 - (1 - \lambda)^i$ bagian masalah (Persamaan (3.1)).
- Pada halaman pesanan pertama Profitina Laundry, empat evaluator menemukan 14 masalah ($\hat{\lambda} = 0.464$); tiga di antaranya memiliki rata-rata keparahan 3 atau lebih dan menjadi penyebab pesanan ganda serta pertanyaan pada minggu pertama.

3.14 Soal Latihan

1. Tetapkan setiap masalah berikut ke salah satu heuristik Nielsen dan berikan keparahannya dengan alasan satu baris: (a) aplikasi ojek tidak menampilkan posisi pengemudi setelah pesanan diterima; (b) aplikasi bank mengeluarkan pengguna tanpa peringatan setelah dua menit ketika mereka sedang mengetik keterangan transfer; (c) portal kampus memakai kata "Submit" di satu halaman dan "Kirim" di halaman berikutnya untuk tindakan yang sama; (d) halaman QRIS sebuah warung menampilkan nominal dalam teks abu-abu kecil di bawah iklan besar.
2. Dengan Persamaan (3.3), berapa evaluator yang diperlukan untuk menemukan 75% dan 90% masalah bila $\lambda = 0.31$? Tunjukkan perhitungannya.
3. Evaluator kelima, E5, meninjau halaman pesanan pertama dan melaporkan P01 (keparahan 4), P05 (3), P11 (3), P13 (2), dan P09 (2). Tambahkan temuan ini ke salinan folder data dan jalankan programnya. Baris mana dari laporan yang berubah, dan mengapa $\hat{\lambda}$ turun padahal E5 tidak menemukan masalah baru?
4. ATM di Subbab 3.3.3 mencegah kesalahan pascapenyelesaian dengan fungsi pemaksa. Carilah kesalahan pascapenyelesaian pada sistem sehari-hari lain di Indonesia (misalnya pompa bensin, mesin fotokopi di tempat fotokopi kampus, atau pengisian pulsa) dan usulkan fungsi pemaksa atau perbaikan lain.
5. Jelaskan, dengan istilah-istilah di Subbab 3.5, bagaimana Liquid Glass melayani satu heuristik dan meregangkan heuristik lain, dan mengapa opsi *Tinted* dari Apple merupakan penerapan H3 dan H7.
6. Bandingkan kedua halaman pada **Gambar 3.1**. Untuk setiap masalah dari keempat belas masalah pada **Kode 3.1**, sebutkan apakah halaman yang diperbaiki membereskannya, dan sebutkan masalah yang tidak dapat dipastikan hanya dari tangkapan layar.
7. Kembangkan `heuristics.py` agar juga mencetak, untuk setiap evaluator, berapa masalah yang ditemukannya dan bagiannya dari seluruh masalah. Apa yang dicetaknya untuk data aslinya?

3.15 Penyelesaian Soal Latihan

1. (a) H1, visibilitas status sistem; keparahan 3, karena penumpang tidak dapat mengetahui apakah pengemudi sedang datang dan mungkin membatalkan atau memesan dua kali. (b) H3, kendali dan kebebasan pengguna (juga H1, karena tidak ada peringatan); keparahan 3, karena keterangan transfer yang sudah diketik hilang dan tugas harus diulang; peringatan dengan pilihan untuk tetap masuk membereskannya. (c) H4, konsistensi dan standar; keparahan 1 atau 2, inkonsistensi kosmetik yang dapat membuat pengguna bertanya-tanya apakah tindakannya berbeda. (d) H8, desain estetis dan minimalis (juga H1, karena nominal adalah status yang penting); keparahan 3, karena pembayar yang tidak dapat membaca nominal tidak dapat memeriksanya sebelum membayar.

2. Untuk 75%: $i \geq \ln(1-0.75)/\ln(1-0.31) = \ln 0.25/\ln 0.69 = (-1.386)/(-0.371) = 3.74$, jadi 4 evaluator. Untuk 90%: $i \geq \ln 0.10/\ln 0.69 = (-2.303)/(-0.371) = 6.21$, jadi 7 evaluator.

3. Salin folder itu lalu tambahkan lima baris `E5,P01,4`, `E5,P05,3`, `E5,P11,3`, `E5,P13,2`, `E5,P09,2` ke `findings.csv`. Program mencetak:

```
Heuristic evaluation: 5 evaluators, 14 problems, 31 findings
rank id   heur found mean max location
   1 P01  H1   4/5   3.8   4  Send button
   2 P11  H1   5/5   3.4   4  Total price
   3 P08  H3   2/5   3.0   3  After sending
   4 P02  H2   3/5   2.7   3  Service list
   5 P03  H5   2/5   2.5   3  Pickup date
   6 P04  H9   2/5   2.5   3  Error message
   7 P05  H6   2/5   2.5   3  Prices
   8 P14  H8   2/5   2.5   3  Small print
   9 P09  H7   2/5   2.0   2  Regular customers
  10 P12  H5   2/5   2.0   2  Phone number
  11 P13  H4   2/5   2.0   2  Date format
  12 P07  H8   1/5   2.0   2  Promo banner
  13 P06  H4   1/5   1.0   1  Buttons
  14 P10  H10  1/5   1.0   1  Weight
Fix before launch (mean severity 3 or more): P01, P11, P08
Problems per heuristic: H1 2, H2 1, H3 1, H4 2, H5 2, H6 1, H7 1, H8 2, H9 1, H10 1
Each evaluator found 44.3% of the problems on average (lambda).
Predicted share found by i evaluators:
  1: 44%  2: 69%  3: 83%  4: 90%  5: 95%  6: 97%
Evaluators needed for 90%: 4
```

Setiap kolom “found” kini dihitung dari 5; rata-rata P01 naik menjadi 3,8, rata-rata P11 turun menjadi 3,4, dan P05 naik dari peringkat 9 ke peringkat 7 dengan rata-rata 2,5. $\hat{\lambda}$ turun dari 0,464 menjadi $31/(5 \times 14) = 0.443$ karena E5 menemukan lima dari empat belas masalah (36%), lebih sedikit daripada rata-rata evaluator lain. Banyaknya evaluator yang diperlukan untuk 90% tetap empat. E5 tidak menemukan masalah baru, dan hal itu sendiri merupakan bukti bahwa daftar empat belas masalah itu sudah hampir lengkap.

4. Contoh: mesin fotokopi swalayan di tempat fotokopi kampus. Tujuannya adalah salinan; setelah mengambilnya, orang sering meninggalkan dokumen asli di atas kaca, sebuah kesalahan pascapenyelesaian. Fungsi pemaksa: mesin tidak mengeluarkan salinan terakhir sampai tutupnya diangkat, atau mesin memindai dari pengumpan yang mengeluarkan dokumen asli bersama salinannya. Contoh lain adalah pompa bensin yang dibayar setelah pengisian sementara tutup tangki dibiarkan terbuka; mobil yang tidak dapat dinyalakan ketika tutup tangki terbuka akan menjadi fungsi pemaksa, dan tutup tangki yang tertambat ke mobil adalah batasan yang mencegahnya hilang.

5. Liquid Glass melayani H8, desain estetis dan minimalis, karena kontrol mundur ke belakang dan konten menjadi pusat perhatian. Ia meregangkan keterbacaan dan H1: teks dan kontrol di atas lapisan tembus pandang di depan konten yang ramai dapat kehilangan kontras, sehingga status yang ditampilkannya lebih sulit dibaca, terutama bagi mata yang menua. Opsi *Tinted* membiarkan pengguna memilih versi yang lebih pekat dan berkontras lebih tinggi: itulah kendali pengguna (H3), sekaligus fleksibilitas (H7), karena pengguna yang memerlukan keterbacaan dapat menyesuaikan antarmuka tanpa memengaruhi mereka yang lebih menyukai gaya bening.

6. Dibereskan dan terlihat: P02 (nama layanan yang jelas), P03 dan P13 (nama hari dengan jam buka), P05 (harga pada ubin), P06 (satu tombol kirim), P07 (tidak ada promo), P10 (penimbangan dan batas minimal dijelaskan), P11 (perkiraan total), P12 (contoh nomor), P14 (tidak ada cetakan kecil abu-abu), P01 (konfirmasi dengan nomor pesanan). P08 dibereskan sebagian: konfirmasi menyebutkan bahwa pesanan dapat diubah sampai pukul 07.00, tetapi tangkapan layar tidak dapat menunjukkan apakah fungsi pengubahannya bekerja. Tangkapan layar juga tidak dapat memastikan batasan pada P03 (apakah hari Minggu benar-benar tidak dapat dipilih), P04 (seperti apa pesan galat sekarang; tidak ada galat yang ditampilkan), atau P09

(jalan pintas ulangi pesanan bagi pelanggan tetap, yang tidak terlihat dan kemungkinan besar belum ditambahkan). Semua itu memerlukan penelusuran halaman yang benar-benar berjalan.

7. Tambahkan ke `main`, setelah membaca temuan:

```
    for e in evaluators:
        k = sum(1 for f in findings if f["evaluator"] == e)
        print(f"{e}: {k} problems ({100 * k / len(problems):.0f}%)")
```

Untuk data aslinya, program mencetak `E1: 8 problems (57%)`, `E2: 6 problems (43%)`, `E3: 5 problems (36%)`, dan `E4: 7 problems (50%)`. Rata-rata keempat bagian itu adalah 46,4%, yaitu $\hat{\lambda}$.

Daftar Pustaka

[1] J. Nielsen, “10 Usability Heuristics for User Interface Design,” Nielsen Norman Group, 1994, updated. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/

[2] J. Nielsen and R. Molich, “Heuristic evaluation of user interfaces,” in *Proc. ACM CHI ’90 Conference on Human Factors in Computing Systems*, Seattle, WA, USA, 1990, pp. 249–256.

[3] Nielsen Norman Group, “How to Conduct a Heuristic Evaluation,” updated, accessed Sep. 2026. [Online]. Available: https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

[4] D. A. Norman, *The Design of Everyday Things*, revised and expanded ed. New York, NY, USA: Basic Books, 2013.

[5] M. D. Byrne and S. Bovair, “A working memory model of a common procedural error,” *Cognitive Science*, vol. 21, no. 1, pp. 31–61, 1997.

[6] B. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 6th ed. Boston, MA, USA: Pearson, 2016.

[7] International Organization for Standardization, *ISO 9241-110:2020 Ergonomics of human-system interaction — Part 110: Interaction principles*. Geneva, Switzerland: ISO, 2020. [Online]. Available: https://www.iso.org/standard/75258.html

[8] Apple Inc., “Apple introduces a delightful and elegant new software design,” Apple Newsroom, Jun. 9, 2025. [Online]. Available: https://www.apple.com/newsroom/2025/06/apple-introduces-a-delightful-and-elegant-new-software-design/

[9] MacRumors, “Here’s How the iOS 26.1 Transparency Toggle Changes Liquid Glass,” Oct. 20, 2025. [Online]. Available: https://www.macrumors.com/2025/10/20/ios-26-1-transparency-option-liquid-glass/

[10] Google Design, “Expressive Design: Google’s UX Research,” accessed Sep. 2026. [Online]. Available: https://design.google/library/expressive-material-design-google-research

[11] Microsoft, “Fluent 2 Design System,” accessed Sep. 2026. [Online]. Available: https://fluent2.microsoft.design/

[12] J. Nielsen, “Severity Ratings for Usability Problems,” Nielsen Norman Group, 1994. [Online]. Available: https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/

[13] J. Nielsen and T. K. Landauer, “A mathematical model of the finding of usability problems,” in *Proc. INTERCHI ’93 Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, 1993, pp. 206–213.

[14] M. Hertzum and N. E. Jacobsen, “The evaluator effect: A chilling fact about usability evaluation methods,” *International Journal of Human-Computer Interaction*, vol. 13, no. 4, pp. 421–443, 2001.

Bab 4

Proses Berpusat pada Pengguna dan Design Thinking

Mengetahui prinsip desain yang baik (Bab 3) dan batas-batas orang yang akan memakainya (Bab 2) belum cukup untuk membangun hal yang tepat. Banyak produk yang mengikuti setiap heuristik tetap gagal karena memecahkan masalah yang tidak dimiliki penggunanya, atau memecahkannya dengan cara yang tidak cocok dengan kehidupan mereka. Desain berpusat pada pengguna adalah disiplin untuk menata sebuah proyek agar hal itu tidak terjadi: memahami orang dan konteksnya lebih dulu, mengubah pemahaman itu menjadi kebutuhan, merancang, lalu menguji rancangan itu bersama orang-orang tersebut, berulang-ulang. Bab ini menjelaskan proses desain berpusat manusia menurut ISO 9241-210, *Double Diamond* dari UK Design Council, dan lima mode *design thinking*; memberikan templat untuk alat-alat kerja utamanya, mulai dari diagram afinitas dan persona hingga peta perjalanan, pertanyaan "bagaimana kita bisa", dan matriks dampak-upaya; membahas apa yang dapat dan tidak dapat dilakukan alat AI dalam proses ini pada 2026; serta membangun program kecil dalam Java dan Python yang mengubah setumpuk catatan wawancara menjadi tema, perjalanan beserta titik-titik keluhannya, pertanyaan desain, dan daftar gagasan yang diprioritaskan. Di akhir bab ini Anda akan mampu merencanakan proyek berpusat pada pengguna, menjalankan langkah sintesisnya dengan catatan nyata, dan menulis kebutuhan pengguna yang terukur.

4.1 Pangkalan Ojek Ingin Punya Aplikasi

Di tepi pasar sebuah kota kecil di Jawa Timur ada sebuah pangkalan ojek: sebuah naungan, sebuah bangku, nomor telepon yang dicat di papan, dan sekitar dua puluh pengemudi yang bergiliran. Pak Darto, ketua koperasi para pengemudi, menyaksikan aplikasi transportasi daring yang besar merebut penumpang muda dan perjalanan jauh. Penumpang tetapnya masih setia: pedagang pasar, anak sekolah, perawat sif malam, dan banyak orang tua yang menelepon pangkalan, atau meminta cucunya menelepon, ketika mereka perlu diantar ke pasar, puskesmas, atau masjid.

Pak Darto meminta sekelompok mahasiswa informatika membuatkan "aplikasi seperti yang besar-besar itu, tetapi milik kita". Naluri pertama para mahasiswa adalah langsung menggambar layar: peta, ikon kendaraan, tombol pesan, penilaian bintang. Dosen mereka menghentikannya dengan sebuah pertanyaan: *masalah siapa yang akan dipecahkan oleh peta itu?* Tidak ada yang tahu. Aplikasi besar punya peta karena penumpangnya berada di tempat yang asing dengan ponsel ber-GPS dan pengemudinya tidak mengenal mereka; penumpang pangkalan ini kebanyakan pelanggan tetap, banyak di antaranya tidak dapat membaca peta, dan para pengemudi mengenal kota lewat masjid dan warungnya, bukan lewat nama jalan.

Maka tim itu melakukan sesuatu yang lebih lambat dan jauh lebih berguna. Mereka duduk di pangkalan selama tiga pagi, berbincang dengan enam penumpang dan empat pengemudi, menuliskan setiap pengamatan pada kertas tempel, dan baru setelah itu mulai bertanya apa yang harus dibangun. Temuan mereka (Subbab 4.8) adalah bahwa saat-saat yang paling menyakitkan sama sekali bukan ketika mencari pengemudi, melainkan ketika *menunggu tanpa kepastian* dan

ketika *menyepakati harga*, dan bahwa kekhawatiran terbesar para pengemudi sendiri bukan aplikasinya, melainkan keadilan urutan giliran di pangkalan. Produk yang akhirnya mereka usulkan jauh lebih kecil daripada "aplikasi seperti yang besar-besar": tabel tarif tetap yang ditampilkan sebelum memesan, pesan satu ketukan "pengemudi sedang menuju" dengan nama dan foto pengemudi, tautan berbagi perjalanan untuk perjalanan malam, dan kemudian antrean giliran otomatis yang adil. Biayanya hanya sebagian kecil dari gagasan semula, dan ia menjawab apa yang benar-benar dikatakan orang-orang itu.

Bab ini menguraikan proses yang mereka tempuh. Program pendamping di akhir bab mengerjakan pembukuan pada langkah sintesis mereka: mengelompokkan catatan menjadi tema, mengukur keluhan di setiap tahap perjalanan, merumuskan pertanyaan desain, dan mengurutkan gagasan menurut dampak dan upayanya.

4.2 Gagasannya dengan Kata-Kata Sederhana

Gagasan desain berpusat pada pengguna muat dalam tiga kalimat yang ditulis Gould dan Lewis pada 1985, dan belum pernah diperbaiki sejak itu: *fokuslah sejak awal pada pengguna dan tugasnya*, *ukurlah secara empiris* bagaimana orang memakai simulasi dan prototipe, dan *rancanglah secara iteratif*, memperbaiki apa yang diungkap pengukuran lalu mengukur lagi [1]. Setiap proses modern, apa pun namanya, adalah cara menata ketiga kebiasaan itu.

> **DEFINISI.** *Desain berpusat pada pengguna* (*user-centered design*, UCD), yang dalam standar ISO disebut *desain berpusat manusia* (*human-centred design*), adalah pendekatan pengembangan sistem interaktif yang bertujuan membuatnya mudah dipakai dan berguna dengan berfokus pada pengguna, kebutuhan, dan persyaratan mereka, serta dengan menerapkan faktor manusia, pengetahuan *usability*, dan teknik-tekniknya [2]. Intinya adalah siklus memahami konteks penggunaan, menetapkan kebutuhan pengguna, menghasilkan solusi desain, dan mengevaluasinya terhadap kebutuhan itu, yang diulang sampai desainnya memenuhi kebutuhan.

Lawan dari desain berpusat pada pengguna bukan "desain yang buruk", melainkan *merancang dari dalam ke luar*: berangkat dari teknologi ("kita punya API peta"), dari pesaing ("aplikasi besar punya penilaian bintang"), atau dari keinginan pihak yang membayar ("aplikasi seperti yang besar-besar"). Titik-titik berangkat itu tidak terlarang, tetapi harus diperiksa terhadap orang-orang yang akan memakai hasilnya. Pak Darto mengenal penumpangnya jauh lebih baik daripada para mahasiswa, dan ia adalah peserta yang sangat penting; tetapi *permintaannya* adalah usulan solusi, dan proses ini dimulai dengan memahami masalah di baliknya.

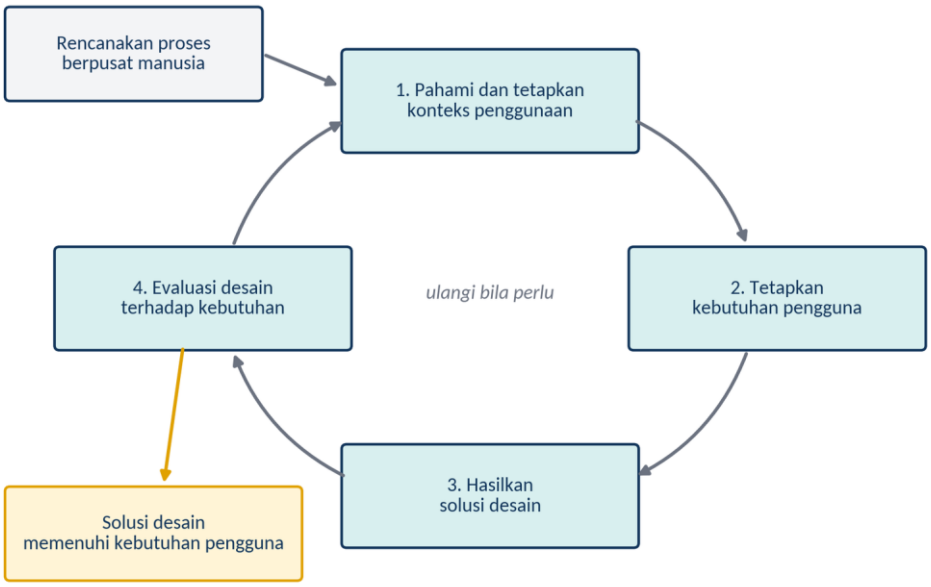
4.3 Proses Desain Berpusat Manusia menurut ISO 9241-210

Standar internasional ISO 9241-210:2019 menggambarkan desain berpusat manusia sebagai enam prinsip dan empat aktivitas [2]. Prinsip-prinsipnya, pada **Tabel 4.1**, menyatakan apa yang harus berlaku pada sebuah proyek; aktivitas-aktivitasnya menyatakan apa yang dikerjakan tim.

Tabel 4.1 Enam prinsip desain berpusat manusia dalam ISO 9241-210:2019, beserta artinya di pangkalan ojek.

Prinsip	Artinya dalam praktik	Di pangkalan ojek
Desain didasarkan pada pemahaman eksplisit tentang pengguna, tugas, dan lingkungan	Tuliskan siapa penggunanya, apa yang mereka kerjakan, dan di mana	Penumpang dari segala usia, banyak yang tidak terbiasa dengan peta; pengemudi di pangkalan; jalan pasar
Pengguna dilibatkan di sepanjang perancangan dan pengembangan	Bukan sekali di awal, tetapi di setiap tahap	Penumpang dan pengemudi mengomentari sketsa dan menguji prototipe
Desain digerakkan dan diperhalus oleh evaluasi berpusat pengguna	Keputusan diuji bersama pengguna, bukan diperdebatkan	Tabel tarif dicoba pada lima penumpang tetap sebelum dicetak
Prosesnya iteratif	Siap untuk kembali; rencanakan hal itu	Pesan "pengemudi sedang menuju" yang pertama terlalu panjang lalu dipersingkat
Desain mencakup seluruh pengalaman pengguna	Sebelum, selama, dan sesudah penggunaan, termasuk perasaan	Rasa aman pada perjalanan malam sama pentingnya dengan kecepatan memesan
Tim desain mencakup keahlian dan sudut pandang multidisiplin	Bukan hanya pemrogram	Seorang pengemudi dan seorang penumpang ikut sesi desain; seorang perawat memberi masukan tentang perjalanan malam

Keempat aktivitas membentuk siklus pada **Gambar 4.1**: memahami dan menetapkan konteks penggunaan; menetapkan kebutuhan pengguna; menghasilkan solusi desain; dan mengevaluasi desain terhadap kebutuhan. Ketika evaluasi menunjukkan bahwa kebutuhan terpenuhi, siklus berakhir; ketika tidak, tim kembali ke aktivitas yang ditunjuk oleh evaluasi, yang bisa saja aktivitas pertama, karena evaluasi dapat mengungkap bahwa konteksnya disalahpahami.



Gambar 4.1 Aktivitas desain berpusat manusia dalam ISO 9241-210: rencanakan, lalu berputar melalui konteks, kebutuhan, desain, dan evaluasi hingga solusinya memenuhi kebutuhan pengguna.

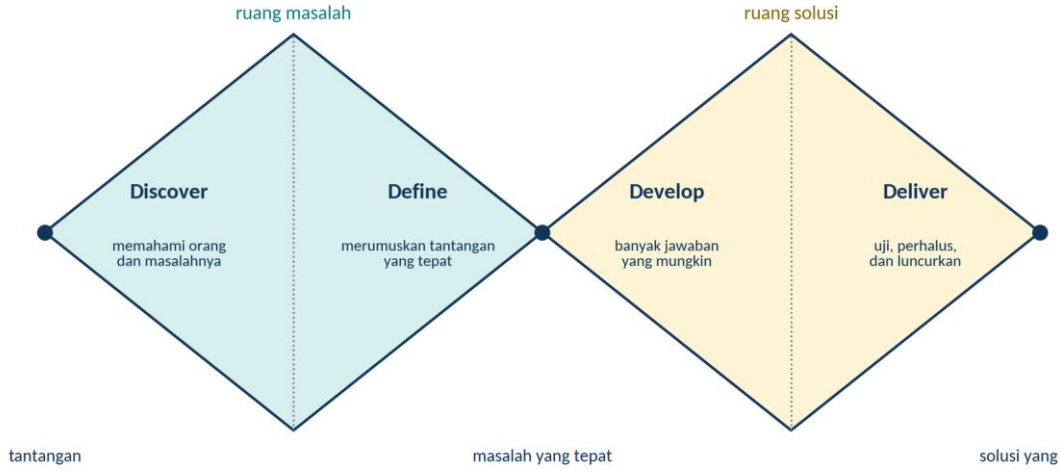
Konteks penggunaan adalah gabungan pengguna, tujuan, tugas, sumber daya, dan lingkungan tempat sebuah sistem dipakai, yaitu konteks "tertentu" dalam definisi *usability* di Bab 1. Untuk pangkalan ojek, konteks itu mencakup penumpang yang berdiri di depan pagar di bawah terik matahari, pengemudi yang berkendara dengan satu tangan di ponsel yang seharusnya tidak disentuh saat berkendara, ponsel Android murah dengan kuota terbatas, dan kebisingan pasar.

Kebutuhan pengguna (*user requirements*) adalah pernyataan tentang apa yang harus dapat dilakukan pengguna dengan sistem dan seberapa baik, ditulis sedemikian rupa sehingga dapat diuji (Subbab 4.5.7).

4.4 Double Diamond dan Design Thinking

4.4.1 Double Diamond

Pada 2004 UK Design Council mempelajari cara perusahaan-perusahaan terkemuka merancang dan merangkum praktik mereka sebagai *Double Diamond* (berlian ganda), seperti tampak pada **Gambar 4.2** [3]. Bentuknya menyandikan gagasan penting: proyek desain berpikir dua kali ke dua arah. Pada berlian pertama, tim *menyebar* (*diverge*) untuk menemukan sebanyak mungkin tentang orang dan masalahnya, lalu *memusat* (*converge*) untuk merumuskan masalah yang tepat. Pada berlian kedua, tim menyebar lagi untuk mengembangkan banyak kemungkinan solusi, lalu memusat untuk menghantarkan solusi yang berhasil, setelah diuji dan diperhalus.



Gambar 4.2 Double Diamond: discover dan define di ruang masalah, develop dan deliver di ruang solusi; setiap berlian mula-mula melebar, lalu menyempit.

Kesalahan paling umum yang diperingatkan berlian ini adalah melompati berlian pertama: melompat dari permintaan awal ("aplikasi seperti yang besar-besar") langsung ke pengembangan solusi. Itulah persisnya yang dicegah dosen para mahasiswa ketika ia menghentikan gambar-gambar layar mereka. Pada 2019 Design Council menerbitkan ulang berlian ini sebagai *Framework for Innovation*, dengan menambahkan empat prinsip (dahulukan manusia; berkomunikasi secara visual dan inklusif; berkolaborasi dan berkreasi bersama; ulangi, ulangi, ulangi) serta bank metode, dan pada 2021 memperluas pendekatannya dengan *Systemic Design Framework* untuk masalah seperti iklim dan kesehatan masyarakat yang tidak dapat dipecahkan oleh satu produk saja [3], [4].

4.4.2 Design thinking

Design thinking adalah nama yang dipakai ketika gagasan-gagasan ini menyebar dari studio desain ke dunia bisnis dan pendidikan, terutama melalui IDEO dan Hasso Plattner Institute of Design di Stanford (d.school). Tim Brown dari IDEO menggambarkannya sebagai membawa metode perancang ke seluruh proses inovasi, mempertemukan kebutuhan manusia dengan apa yang layak secara teknologi dan menguntungkan secara ekonomi [5]. d.school menguraikan lima *mode*: **empathize** (memahami orang melalui pengamatan dan percakapan), **define** (merumuskan

inti masalah sebagai sebuah sudut pandang), **ideate** (membangkitkan banyak gagasan), **prototype** (membuat benda cepat dan murah yang dapat dicoba), dan **test** (mencobanya bersama orang lalu belajar) [6].

Mode-mode itu bukan urutan satu arah. Pengujian sering mengirim tim kembali ke tahap empati; prototipe sering memperjelas rumusan masalah. Ungkapan d.school "*a bias toward action*" (kecenderungan untuk bertindak) menamai kekuatan nyata dari pendekatan ini: buatlah sesuatu yang kecil lalu belajarlah darinya, alih-alih berdiskusi secara abstrak.

4.4.3 Lean UX dan tim *agile*

Sebagian besar perangkat lunak saat ini dibangun oleh tim *agile* dalam siklus pendek, yang mula-mula tampak tidak menyisakan ruang untuk riset. *Lean UX* mendamaikan keduanya: setiap keputusan desain diperlakukan sebagai hipotesis ("kami yakin bahwa menampilkan tarif sebelum memesan akan mengurangi tawar-menawar; kami akan tahu bahwa kami benar bila makin sedikit penumpang yang menanyakan harga lewat telepon"), dibuat hal terkecil yang dapat mengujinya, lalu hasilnya diukur [7]. Banyak tim menjalankan pengembangan *dual-track* (jalur ganda), yang di dalamnya jalur penemuan berupa riset dan pembuatan prototipe berjalan satu atau dua siklus di depan jalur penghantaran yang membangun perangkat lunaknya.

Tabel 4.2 menjajarkan ketiga uraian itu. Ketiganya bukan pesaing, melainkan kosakata yang berbeda untuk siklus dasar yang sama; tim sebaiknya memilih satu kosakata agar semua orang berbicara dengan bahasa yang sama, dan mengetahui bahwa yang lain adalah terjemahannya.

Tabel 4.2 Siklus yang sama dalam tiga kosakata, beserta metode yang lazim untuk setiap langkah.

Aktivitas ISO 9241-210	Double Diamond	Mode design thinking	Metode yang lazim
Memahami dan menetapkan konteks penggunaan	Discover	Empathize	Wawancara, pengamatan, studi buku harian (Bab 5)
Menetapkan kebutuhan pengguna	Define	Define	Diagram afinitas, persona, peta perjalanan, pernyataan masalah, kebutuhan terukur
Menghasilkan solusi desain	Develop	Ideate, prototype	Curah gagasan, sketsa, prototipe kertas dan interaktif (Bab 6)
Mengevaluasi desain terhadap kebutuhan	Deliver (uji dan perhalus)	Test	Evaluasi heuristik (Bab 3), uji *usability* (Bab 7), uji A/B (Bab 14)

4.5 Alat-Alat Kerja, dengan Templat

Subbab ini menguraikan alat-alat yang dipakai tim ojek di antara wawancara dan prototipe pertama. Bab 5 membahas wawancaranya sendiri; di sini kita berangkat dari catatan.

4.5.1 Diagram afinitas

Diagram afinitas (*affinity diagram*) menata banyak pengamatan ke dalam kelompok berdasarkan kemiripan, dari bawah ke atas. Setiap pengamatan ditulis pada satu catatan sebagai fakta dengan istilah partisipan sendiri ("Menunggu di depan pagar tanpa tahu apakah ada yang datang"), bukan sebagai tafsiran ("butuh pelacakan"). Tim lalu mendekatkan catatan-catatan yang tampak sejenis, tanpa menentukan kategorinya lebih dulu, dan akhirnya menamai setiap kelompok dengan label singkat [8]. Label yang muncul adalah *tema* riset itu. Ke-37 catatan tim ojek membentuk delapan tema, dari "Waktu tunggu tak pasti" hingga "Giliran yang adil".

Dua angka menggambarkan bobot sebuah tema: berapa banyak catatan di dalamnya, dan berapa banyak *orang yang berbeda* yang menyumbangkannya. Angka kedua lebih penting. Sepuluh catatan dari satu penumpang yang banyak bicara adalah pandangan satu orang; lima catatan dari lima orang adalah sebuah pola. *Jangkauan* (*reach*) sebuah tema t adalah bagian dari seluruh partisipan yang menyebutnya,

$$r_t = \frac{|P_t|}{|P|} \tag{4.1}$$

dengan P_t himpunan partisipan yang punya setidaknya satu catatan dalam tema t, P himpunan semua partisipan, dan $|\cdot|$ banyaknya anggota.

4.5.2 Persona

Persona adalah uraian singkat dan konkret tentang pengguna yang khas, dibangun dari riset, yang dapat terus diingat tim selama merancang: nama, foto atau gambar, tujuan, konteks, kekesalan, dan satu kutipan yang khas [9]. Persona mencegah "pengguna karet" yang dapat dilenturkan untuk membenarkan fitur apa pun; pertanyaan "akankah Mbah Sri memakai ini?" punya jawaban yang konkret, sedangkan "akankah pengguna memakai ini?" tidak. **Tabel 4.3** memperlihatkan salah satu persona tim ojek, yang dibangun dari catatan partisipan P4.

Tabel 4.3 Persona yang dibangun dari catatan riset, beserta implikasi desain yang disarankannya.

Kolom	Isi
Nama dan konteks	Mbah Sri, 72 tahun, berjualan sayur di pasar pada hari kerja; tinggal bersama keluarga anaknya dua kilometer dari pasar
Tujuan	Sampai di pasar pukul 05.30 dan pulang pada tengah hari tanpa bergantung pada anaknya
Perilaku	Anaknya biasanya menelepon pangkalan untuknya; mengenal tempat hanya lewat patokan; lebih suka uang tunai
Kekesalan	Berdiri di bawah terik tanpa tahu berapa lama pengemudi akan datang; tidak dapat membaca peta
Kutipan	"Cukup kasih tahu saya siapa yang datang dan kapan. Saya kenal semua tukang ojek di sini."
Implikasi desain	Tidak perlu peta; pesan yang jelas dengan nama pengemudi dan waktu tiba; patokan, bukan alamat

Persona hanya sebaik buktinya. Persona yang dikarang di belakang meja, atau yang dibangkitkan alat AI dari pengetahuan umum (Subbab 4.6), menggambarkan stereotipe, bukan penggunanya.

4.5.3 Peta perjalanan

Peta perjalanan (*journey map*) mengikuti seorang pengguna melalui tahap-tahap sebuah tugas dari waktu ke waktu dan mencatat, untuk setiap tahap, apa yang dilakukan, dipikirkan, dan dirasakan orang itu, di mana keluhannya, dan *titik sentuh* (*touchpoint*) mana yang terlibat (telepon, pesan, naungan, pembayaran). Untuk pangkalan ojek, tahap-tahapnya adalah *Mencari* pengemudi (*Find*), *Sepakat* soal tempat dan harga (*Agree*), *Menunggu* pengemudi (*Wait*), *Naik* (*Ride*), dan *Bayar* (*Pay*). Garis emosi pada peta itu, yang biasanya digambar sebagai kurva melintasi tahap-tahap, memperlihatkan sekilas di mana pengalamannya menyakitkan. Tim memberi setiap catatan skor keluhan dari 1 (ringan) sampai 5 (berat); rata-rata keluhan pada tahap s adalah

$$\bar{p}_s = \frac{1}{|N_s|} \sum_{n \in N_s} p_n \tag{4.2}$$

dengan N_s himpunan catatan tentang tahap s dan p_n skor keluhan catatan n. Subbab 4.8 menggambar kurvanya.

4.5.4 Peta empati dan *jobs to be done*

Dua alat lain sering dipakai berdampingan. *Peta empati* (*empathy map*) menata apa yang *dikatakan*, *dipikirkan*, *dilakukan*, dan *dirasakan* sebuah persona dalam empat kuadran, yang membantu tim melihat pertentangan: Mbah Sri *mengatakan* tidak keberatan menunggu, tetapi *melakukan* hal lain, yaitu berdiri di depan pagar dua puluh menit lebih awal. Sudut pandang *jobs to be done* bertanya "pekerjaan" apa yang "disewakan" seseorang kepada sebuah produk: tidak ada orang yang menginginkan aplikasi ojek; orang ingin "sampai di pasar sebelum sayur yang bagus habis" [10]. Merumuskan kebutuhan sebagai pekerjaan menjaga tim agar tidak mencampuradukkan kebutuhan dengan satu solusi tertentu.

4.5.5 Pernyataan masalah dan pertanyaan "bagaimana kita bisa"

Langkah perumusan diakhiri dengan *pernyataan masalah* yang menyebut penggunanya, kebutuhannya, dan wawasannya: "Penumpang tetap pangkalan, banyak di antaranya orang tua, perlu tahu kapan pengemudinya tiba, karena menunggu tanpa kepastian membuat mereka berangkat dari rumah terlalu awal dan berdiri di bawah terik." Dari situ tim menurunkan pertanyaan *how might we* (HMW, "bagaimana kita bisa"), yang mengubah masalah menjadi undangan untuk bergagasan: "Bagaimana kita bisa menunjukkan kepada penumpang kapan pengemudinya tiba?" Pertanyaan HMW yang baik cukup luas untuk memungkinkan banyak jawaban dan cukup sempit untuk memberi arah. "Bagaimana kita bisa membangun aplikasi?" terlalu sempit (menyebut solusi) dan tidak mengatakan apa pun tentang masalahnya; "Bagaimana kita bisa memperbaiki transportasi?" terlalu luas untuk ditindaklanjuti.

4.5.6 Ideasi dan matriks dampak-upaya

Dalam ideasi, tim membangkitkan banyak jawaban untuk setiap pertanyaan HMW tanpa menilainya, lalu memilih. Alat pemilihan yang sederhana dan banyak dipakai adalah *matriks dampak-upaya* (*impact-effort matrix*). Setiap gagasan diberi skor dampak (seberapa besar ia mengurangi keluhan yang ditemukan dalam riset, 1 sampai 5) dan skor upaya (seberapa banyak pekerjaan yang diperlukan, 1 sampai 5) yang disepakati tim, lalu ditempatkan di salah satu dari empat kuadran: *quick win* atau menang cepat (dampak tinggi, upaya rendah), *big bet* atau taruhan besar (dampak tinggi, upaya tinggi), *fill-in* atau pengisi (dampak rendah, upaya rendah), dan gagasan yang *ditunda dulu* (dampak rendah, upaya tinggi). Di dalam satu kuadran, nilai per upaya

$$v = \frac{I}{E} \tag{4.3}$$

dengan I skor dampak dan E skor upaya, mengurutkan gagasan-gagasan itu, dan suara tim memecah hasil seri. Skor-skor ini adalah pertimbangan, bukan pengukuran; matriks tidak memutuskan bagi tim, tetapi membuat penalaran tim terlihat dan dapat didiskusikan.

4.5.7 Kebutuhan pengguna yang terukur

Kebutuhan yang menutup langkah perumusan harus dapat diuji, dalam pengertian *usability* di Bab 1: pengguna, tugas, ukuran, dan target. Contoh untuk pangkalan ojek: "Penumpang tetap berusia 60 tahun ke atas memesan perjalanan dari tautan tersimpan dalam waktu kurang dari 30 detik tanpa bantuan, pada 4 dari 5 percobaan"; "setelah memesan, 9 dari 10 penumpang dapat menyebutkan nama pengemudi dan perkiraan waktu tibanya"; "dalam sebulan pemakaian, telepon ke pangkalan yang menanyakan di mana pengemudinya turun separuh". Setiap kebutuhan dapat diperiksa dalam uji *usability* (Bab 7) atau dengan data (Bab 14), dan itulah yang memungkinkan langkah evaluasi pada **Gambar 4.1**.

4.6 AI dalam Proses, 2026

Alat AI mengubah pekerjaan sehari-hari tim desain pada 2025 dan 2026. Di Config 2026, Figma mengumumkan agen AI di dalam kanvas desainnya dengan pencarian web, *skill* khusus, dan sambungan ke alat eksternal, di samping *code layers* yang membawa kode yang berjalan ke atas kanvas [11]; alat bantu serupa ada di aplikasi-aplikasi lain. Alat-alat itu sungguh berguna untuk separuh *develop* dari Double Diamond: membangkitkan varian tata letak, menyusun draf teks, mengubah sketsa menjadi prototipe yang dapat diklik, meringkas transkrip yang panjang. Bab 6 memakainya untuk membuat prototipe.

Alat-alat itu jauh lebih lemah, dan kadang merugikan, pada separuh *discover*. Alat yang menawarkan "pengguna sintetis" (*synthetic users*), yaitu persona atau jawaban wawancara hasil bangkitan AI, menghasilkan pernyataan yang masuk akal yang mencerminkan apa yang lazim dalam data latihnya, bukan apa yang benar tentang sekelompok orang tertentu. Tinjauan Nielsen Norman Group menyimpulkan bahwa keluaran semacam itu dapat membantu menyiapkan riset, misalnya dengan menyarankan pertanyaan yang perlu diajukan, tetapi tidak dapat menggantikan riset bersama orang sungguhan [12]. Tidak ada model bahasa yang akan memberi tahu tim ojek bahwa kekhawatiran utama para pengemudi adalah keadilan urutan giliran di pangkalan mereka, atau bahwa Mbah Sri mengenal setiap pengemudi dengan namanya. Aturan buku ini sederhana: AI boleh membantu Anda *mengolah* apa yang dikatakan orang sungguhan; AI tidak boleh *mengarang* apa yang akan mereka katakan.

Bekerja dengan orang sungguhan juga berarti bekerja dengan data pribadi mereka. Catatan wawancara, rekaman, dan foto partisipan adalah data pribadi menurut Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi, dan pemrosesannya memerlukan dasar yang sah seperti persetujuan eksplisit untuk tujuan tertentu [13]; peraturan pelaksananya, Peraturan Pemerintah Nomor 33 Tahun 2026, berlaku mulai Januari 2027 [14]. Dalam praktik: beri tahu partisipan untuk apa riset itu, mintalah persetujuan untuk mencatat atau merekam, pakailah kode (P1, D1) alih-alih nama dalam berkas yang dibagikan, dan hapus rekaman setelah analisis selesai. Menempelkan transkrip mentah lengkap dengan nama ke alat AI berbasis awan berarti mengalihkan data pribadi ke pihak ketiga dan memerlukan kehati-hatian yang sama.

4.7 Membangun Alat Sintesis

Diagram afinitas dibuat oleh orang-orang yang memindah-mindahkan catatan di dinding atau di papan tulis digital; bagian itu tetap pekerjaan manusia, karena memutuskan catatan mana yang sejenis adalah penafsiran. Apa yang datang sesudahnya, yaitu menghitung catatan dan orang per tema, merata-ratakan keluhan per tahap, menuliskan pertanyaan HMW untuk tema-tema terkuat,

dan mengurutkan gagasan, adalah pembukuan, dan program di subbab ini mengerjakannya agar angka-angkanya benar dan dapat dihitung ulang setiap kali sebuah catatan dipindahkan.

4.7.1 Datanya

Tiga berkas CSV menyimpan hasil kerja tim. **Kode 4.1** mendaftar delapan tema dengan label singkat dan frasa "bagaimana kita bisa" yang disepakati untuk masing-masing. (Seperti di Bab 3, isi berkas data ditulis dalam bahasa Inggris agar sama dengan edisi bahasa Inggris dan keluaran program.)

Kode 4.1 `data/themes.csv`: tema-tema diagram afinitas beserta frasa "bagaimana kita bisa".

```
theme,label,hmw
T1,Unclear price,make the fare clear before the ride starts
T2,Unknown waiting time,show passengers when their driver will arrive
T3,Calling the base,let passengers book without a phone call
T4,Explaining the place,help passengers say where they are without street names
T5,Safety at night,help passengers feel safe on night rides
T6,Big apps feel hard,keep booking as simple as asking a neighbour
T7,Cash and change,make paying easy without exact change
T8,Fair turns,keep the base's turn order fair for every driver
```

Kode 4.2 menyimpan ke-37 catatan: pengenal, kode partisipan (P untuk penumpang, D untuk pengemudi), peran, tahap perjalanan, tema tempat catatan itu diletakkan, skor keluhan, dan catatannya sendiri.

Kode 4.2 `data/notes.csv`: 37 catatan dari enam penumpang dan empat pengemudi, masing-masing dengan tahap, tema, dan skor keluhan.

```
id,participant,role,stage,theme,pain,note
N01,P1,passenger,Find,T3,4,Calls the base but the phone is busy at 7 am
N02,P1,passenger,Agree,T1,4,Never sure the price is the same as last time
N03,P1,passenger,Wait,T2,5,Waits at the gate not knowing if anyone is coming
N04,P2,passenger,Find,T3,3,Does not have the base number saved on the new phone
N05,P2,passenger,Agree,T4,4,"Explains ""after the mosque, green fence"" twice"
N06,P2,passenger,Pay,T7,3,Driver has no change for a 50 thousand note
N07,P3,passenger,Find,T6,3,Tried a big app; too many screens and promotions
N08,P3,passenger,Wait,T2,4,Asks a neighbour to call again after 15 minutes
N09,P3,passenger,Ride,T5,4,Coming home from the night shift feels unsafe
N10,P3,passenger,Agree,T1,3,Price for the night differs and nobody explains why
N11,P4,passenger,Find,T3,2,Her son usually calls the base for her
N12,P4,passenger,Agree,T4,5,Cannot read a map; knows only landmarks
N13,P4,passenger,Wait,T2,4,Stands in the sun because she does not know how long
N14,P4,passenger,Pay,T7,2,Prefers cash; does not trust QRIS yet
N15,P5,passenger,Find,T6,3,Uninstalled a big app after a surprise fee
N16,P5,passenger,Ride,T5,3,Wants to know the driver's name before getting on
N17,P5,passenger,Pay,T7,2,Would pay by QRIS if the amount is shown first
N18,P5,passenger,Agree,T1,4,Haggles every time; finds it tiring
N19,P6,passenger,Find,T3,3,Sends a WhatsApp voice note; reply comes late
N20,P6,passenger,Wait,T2,3,Wants a message when the driver leaves the base
N21,P6,passenger,Ride,T5,2,Likes riding with drivers she knows
N22,P6,passenger,Agree,T4,3,"Shares location, but the pin is on the wrong street"
N23,D1,driver,Find,T8,4,Arguments about whose turn it is when a call comes
N24,D1,driver,Agree,T4,4,Loses time finding houses without numbers
N25,D1,driver,Pay,T7,3,Needs to keep small notes for change
N26,D2,driver,Find,T6,4,Commission of big apps is too high for short trips
N27,D2,driver,Agree,T1,3,Passengers think the fare is made up on the spot
N28,D2,driver,Find,T8,3,Whoever holds the base phone gets the good trips
N29,D3,driver,Wait,T2,2,Would send a message if it were one tap
N30,D3,driver,Find,T3,3,Misses calls while riding
N31,D3,driver,Ride,T5,2,Wants passengers to know who he is
N32,D4,driver,Find,T8,5,Young drivers skip the queue with their own apps
N33,D4,driver,Agree,T4,3,Old customers explain places by old names
N34,D4,driver,Pay,T7,2,Would accept QRIS if the cooperative helps set it up
N35,P5,passenger,Wait,T2,4,Leaves the house far too early to be safe
N36,P2,passenger,Wait,T2,3,Keeps calling the base to ask where the driver is
N37,D3,driver,Agree,T1,3,Explains the fare again at almost every trip
```

Kode 4.3 mendaftar sepuluh gagasan dari sesi ideasi beserta skor dampak dan upaya dari tim serta banyaknya suara yang diterima masing-masing dari penumpang dan pengemudi dalam sesi umpan balik.

Kode 4.3 `data/ideas.csv`: sepuluh gagasan dengan dampak, upaya, dan suara.

```
idea,theme,impact,effort,votes
Fixed fare table by zone shown before booking,T1,5,2,9
Driver-on-the-way message with name and photo,T2,5,2,8
One-tap booking page from a saved WhatsApp link,T3,4,3,7
Pick the pickup point from photos of landmarks,T4,4,4,5
Share-my-ride link for family on night trips,T5,4,2,6
Automatic fair queue for the base,T8,5,3,7
Cooperative QRIS with the amount shown,T7,3,2,4
Own full ride-hailing app like the big ones,T6,3,5,2
Loyalty points for regular passengers,T6,2,3,3
Voice booking by phone with a menu,T3,3,4,4
```

4.7.2 Programnya

Kode 4.4 adalah versi Python-nya.

Kode 4.4 `synthesis.py`: tema beserta jangkauannya, keluhan per tahap perjalanan, pertanyaan “bagaimana kita bisa”, dan gagasan yang diprioritaskan.

```
"""Dari catatan wawancara ke tema, perjalanan, pertanyaan, dan prioritas."""
import csv
import sys

STAGES = ["Find", "Agree", "Wait", "Ride", "Pay"]
QUADRANTS = ["Quick win", "Big bet", "Fill-in", "Avoid for now"]


def read_csv(path):
    with open(path, newline="", encoding="utf-8") as f:
        return list(csv.DictReader(f))


def affinity(notes):
    """Kelompokkan catatan per tema: jumlah catatan, orang berbeda, peran."""
    board = {}
    for n in notes:
        t = board.setdefault(n["theme"], {"notes": 0, "people": set(),
                                          "roles": set()})
        t["notes"] += 1
        t["people"].add(n["participant"])
        t["roles"].add(n["role"])
    return sorted(board.items(),
                  key=lambda kv: (-len(kv[1]["people"]), -kv[1]["notes"], kv[0]))


def journey(notes):
    """Rata-rata keluhan (1 ringan-5 berat) dan jumlah catatan per tahap."""
    result = []
    for stage in STAGES:
        pains = [int(n["pain"]) for n in notes if n["stage"] == stage]
        result.append((stage, len(pains), sum(pains) / len(pains)))
    return result


def quadrant(impact, effort):
    """Matriks dampak-upaya: dampak tinggi 4 atau 5, upaya rendah 1 atau 2."""
    if impact >= 4:
        return "Quick win" if effort <= 2 else "Big bet"
    return "Fill-in" if effort <= 2 else "Avoid for now"
```

Kode 4.4 (Lanjutan) *synthesis.py: tema beserta jangkauannya, keluhan per tahap perjalanan, pertanyaan "bagaimana kita bisa", dan gagasan yang diprioritaskan.*

```
def prioritise(ideas):
    """Kuadran dulu, lalu nilai per upaya, lalu suara."""
    rows = []
    for i in ideas:
        impact, effort = int(i["impact"]), int(i["effort"])
        rows.append((quadrant(impact, effort), impact, effort,
                     impact / effort, int(i["votes"]), i["idea"]))
    return sorted(rows, key=lambda r: (QUADRANTS.index(r[0]), -r[3], -r[4]))


def main():
    folder = sys.argv[1] if len(sys.argv) > 1 else "data"
    notes = read_csv(f"{folder}/notes.csv")
    themes = {t["theme"]: t for t in read_csv(f"{folder}/themes.csv")}
    people = {n["participant"]: n["role"] for n in notes}
    roles = list(people.values())
    print(f"Affinity board: {len(notes)} notes from {len(people)} people "
          f"({roles.count('passenger')} passengers, "
          f"{roles.count('driver')} drivers)")
    print("theme label                  notes people roles")
    board = affinity(notes)
    for theme, t in board:
        who = "+".join(sorted(t["roles"], reverse=True))
        print(f"{theme:5} {themes[theme]['label']:22}{t['notes']:6}"
              f"{len(t['people']):7}  {who}")
    print("Journey: mean pain per stage (1 mild, 5 severe)")
    for stage, count, pain in journey(notes):
        print(f"  {stage:6}{count:3} notes  {pain:.1f}  "
              + "#" * round(2 * pain))
    print("How might we ...")
    for k, (theme, t) in enumerate(board[:3], 1):
        print(f"  {k}. ... {themes[theme]['hmw']}? "
              f"({theme}, {len(t['people'])} people)")
    print("Ideas by impact and effort")
    print("quadrant      imp eff ratio votes idea")
    for q, imp, eff, ratio, votes, idea in prioritise(
            read_csv(f"{folder}/ideas.csv")):
        print(f"{q:14}{imp:3}{eff:4}{ratio:6.2f}{votes:6}  {idea}")


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

`affinity` membangun, untuk setiap tema, hitungan catatan dan dua himpunan, yaitu himpunan partisipan dan himpunan peran; himpunan menyimpan setiap anggota sekali saja, sehingga ukurannya adalah banyaknya orang yang *berbeda*, yaitu $|P_t|$ pada Persamaan (4.1). Tema diurutkan menurut banyaknya orang, lalu banyaknya catatan, lalu pengenalnya. `journey` menghitung Persamaan (4.2) untuk kelima tahap dalam urutan tetapnya. `quadrant` menempatkan sebuah gagasan dalam matriks dampak-upaya, dengan 4 atau 5 sebagai dampak tinggi dan 1 atau 2 sebagai upaya rendah, dan `prioritise` mengurutkan gagasan menurut kuadran, lalu nilai per upaya dari Persamaan (4.3), lalu suara. `main` mencetak empat blok: papan afinitas, perjalanan dengan batang kecil dari karakter `#` (dua per poin keluhan) sebagai versi teks dari kurva emosi, pertanyaan HMW untuk tiga tema terkuat, dan gagasan yang diprioritaskan.

Kode 4.5 adalah versi Java-nya.

Kode 4.5 `Synthesis.java`: sintesis yang sama dalam Java.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.TreeSet;

/** Dari catatan wawancara ke tema, perjalanan, pertanyaan, dan prioritas. */
public class Synthesis {
    static final List<String> STAGES =
        List.of("Find", "Agree", "Wait", "Ride", "Pay");
    static final List<String> QUADRANTS =
        List.of("Quick win", "Big bet", "Fill-in", "Avoid for now");

    record Theme(String id, int notes, TreeSet<String> people,
                 TreeSet<String> roles) {}

    record Idea(String quadrant, int impact, int effort, double ratio, int votes,
                String text) {}

    /** Berkas CSV sebagai peta nama kolom ke teks; "teks, berkutip" boleh. */
    static List<Map<String, String>> readCsv(String path) throws IOException {
        List<String> lines = Files.readAllLines(Path.of(path));
        String[] head = lines.get(0).split(",");
        List<Map<String, String>> rows = new ArrayList<>();
        for (String line : lines.subList(1, lines.size())) {
            List<String> cells = new ArrayList<>();
            StringBuilder cell = new StringBuilder();
            boolean quoted = false;
            for (int i = 0; i < line.length(); i++) {
                char c = line.charAt(i);
                if (c == '"' && quoted && i + 1 < line.length()
                        && line.charAt(i + 1) == '"') {
                    cell.append('"');
                    i++;
                } else if (c == '"') {
                    quoted = !quoted;
                } else if (c == ',' && !quoted) {
                    cells.add(cell.toString());
                    cell.setLength(0);
                } else {
                    cell.append(c);
```

Kode 4.5 (Lanjutan) *Synthesis.java: sintesis yang sama dalam Java.*

```
                }
            }
            cells.add(cell.toString());
            Map<String, String> row = new HashMap<>();
            for (int i = 0; i < head.length; i++) row.put(head[i], cells.get(i));
            rows.add(row);
        }
        return rows;
    }

    /** Kelompokkan catatan per tema: jumlah catatan, orang berbeda, peran. */
    static List<Theme> affinity(List<Map<String, String>> notes) {
        Map<String, Theme> board = new LinkedHashMap<>();
        for (var n : notes) {
            Theme t = board.computeIfAbsent(n.get("theme"),
                k -> new Theme(k, 0, new TreeSet<>(), new TreeSet<>()));
            t.people().add(n.get("participant"));
            t.roles().add(n.get("role"));
            board.put(t.id(),
                      new Theme(t.id(), t.notes() + 1, t.people(), t.roles()));
        }
        List<Theme> list = new ArrayList<>(board.values());
        list.sort(Comparator.comparingInt((Theme t) -> -t.people().size())
                  .thenComparingInt(t -> -t.notes()).thenComparing(Theme::id));
        return list;
    }

    /** Matriks dampak-upaya: dampak tinggi 4 atau 5, upaya rendah 1 atau 2. */
    static String quadrant(int impact, int effort) {
        if (impact >= 4) return effort <= 2 ? "Quick win" : "Big bet";
        return effort <= 2 ? "Fill-in" : "Avoid for now";
    }

    /** Kuadran dulu, lalu nilai per upaya, lalu suara. */
    static List<Idea> prioritise(List<Map<String, String>> ideas) {
        List<Idea> rows = new ArrayList<>();
        for (var i : ideas) {
            int impact = Integer.parseInt(i.get("impact"));
            int effort = Integer.parseInt(i.get("effort"));
            rows.add(new Idea(quadrant(impact, effort), impact, effort,
                              (double) impact / effort,
                              Integer.parseInt(i.get("votes")), i.get("idea")));
        }
        rows.sort(Comparator
                  .comparingInt((Idea r) -> QUADRANTS.indexOf(r.quadrant()))
                  .thenComparingDouble(r -> -r.ratio())
```

Kode 4.5 (Lanjutan) *Synthesis.java: sintesis yang sama dalam Java.*

```
                    .thenComparingInt(r -> -r.votes()));
        return rows;
    }

    public static void main(String[] args) throws IOException {
        String folder = args.length > 0 ? args[0] : "data";
        var notes = readCsv(folder + "/notes.csv");
        Map<String, Map<String, String>> themes = new HashMap<>();
        for (var t : readCsv(folder + "/themes.csv")) themes.put(t.get("theme"), t);
        Map<String, String> people = new LinkedHashMap<>();
        for (var n : notes) people.put(n.get("participant"), n.get("role"));
        long passengers = people.values().stream()
                                .filter(r -> r.equals("passenger")).count();
        System.out.printf("Affinity board: %d notes from %d people "
                          + "(%d passengers, %d drivers)%n", notes.size(),
                          people.size(), passengers, people.size() - passengers);
        System.out.println("theme label                  notes people roles");
        List<Theme> board = affinity(notes);
        for (Theme t : board) {
            String who = String.join("+", t.roles().descendingSet());
            System.out.printf("%-5s %-22s%6d%7d  %s%n", t.id(),
                              themes.get(t.id()).get("label"), t.notes(),
                              t.people().size(), who);
        }
        System.out.println("Journey: mean pain per stage (1 mild, 5 severe)");
        for (String stage : STAGES) {
            int count = 0, sum = 0;
            for (var n : notes) {
                if (n.get("stage").equals(stage)) {
                    count++;
                    sum += Integer.parseInt(n.get("pain"));
                }
            }
            double pain = (double) sum / count;
            System.out.printf("  %-6s%3d notes  %.1f  %s%n", stage, count, pain,
                              "#".repeat((int) Math.round(2 * pain)));
        }
        System.out.println("How might we ...");
        for (int k = 0; k < 3; k++) {
            Theme t = board.get(k);
            System.out.printf("  %d. ... %s? (%s, %d people)%n", k + 1,
                              themes.get(t.id()).get("hmw"), t.id(),
                              t.people().size());
        }
        System.out.println("Ideas by impact and effort");
        System.out.println("quadrant      imp eff ratio votes idea");
```

Kode 4.5 (Lanjutan) *Synthesis.java: sintesis yang sama dalam Java.*

```
        for (Idea r : prioritise(readCsv(folder + "/ideas.csv"))) {
            System.out.printf("%-14s%3d%4d%6.2f%6d  %s%n", r.quadrant(), r.impact(),
                              r.effort(), r.ratio(), r.votes(), r.text());
        }
    }
}
```

Versi Java memakai dua *record*, `Theme` dan `Idea`. Karena *field* sebuah *record* tidak dapat diubah, `affinity` mengganti *record* sebuah tema dengan *record* baru yang hitungan catatannya lebih besar satu, sementara dua objek `TreeSet` di dalamnya dipakai bersama dan tumbuh di tempat; `TreeSet` juga menjaga peran tetap terurut, sehingga `descendingSet` memberikan “passenger” sebelum “driver”, seperti `sorted(..., reverse=True)` di Python. Pembaca CSV-nya menangani kolom berkutip, karena dua catatan mengandung koma di dalam tanda kutip.

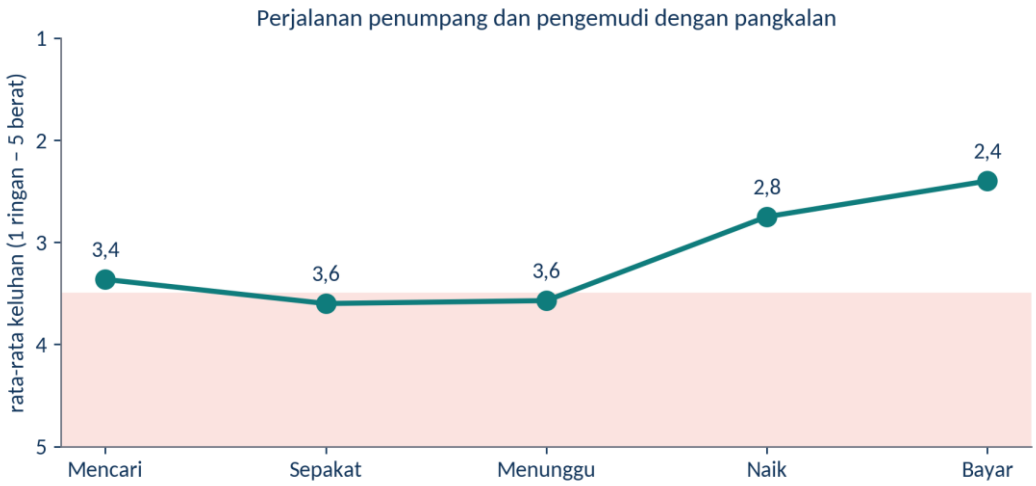
4.8 Menjalankannya

```
$ python3 synthesis.py
$ java -cp out Synthesis
```

```
Affinity board: 37 notes from 10 people (6 passengers, 4 drivers)
theme label                 notes people roles
T2    Unknown waiting time      7      7  passenger+driver
T1    Unclear price             5      5  passenger+driver
T3    Calling the base          5      5  passenger+driver
T4    Explaining the place      5      5  passenger+driver
T7    Cash and change           5      5  passenger+driver
T5    Safety at night           4      4  passenger+driver
T6    Big apps feel hard        3      3  passenger+driver
T8    Fair turns                3      3  driver
Journey: mean pain per stage (1 mild, 5 severe)
  Find   11 notes  3.4  #######
  Agree  10 notes  3.6  #######
  Wait    7 notes  3.6  #######
  Ride    4 notes  2.8  ######
  Pay     5 notes  2.4  #####
How might we ...
  1. ... show passengers when their driver will arrive? (T2, 7 people)
  2. ... make the fare clear before the ride starts? (T1, 5 people)
  3. ... let passengers book without a phone call? (T3, 5 people)
Ideas by impact and effort
quadrant      imp eff ratio votes idea
Quick win       5   2  2.50     9  Fixed fare table by zone shown before booking
Quick win       5   2  2.50     8  Driver-on-the-way message with name and photo
Quick win       4   2  2.00     6  Share-my-ride link for family on night trips
Big bet         5   3  1.67     7  Automatic fair queue for the base
Big bet         4   3  1.33     7  One-tap booking page from a saved WhatsApp link
Big bet         4   4  1.00     5  Pick the pickup point from photos of landmarks
Fill-in         3   2  1.50     4  Cooperative QRIS with the amount shown
Avoid for now   3   4  0.75     4  Voice booking by phone with a menu
Avoid for now   2   3  0.67     3  Loyalty points for regular passengers
Avoid for now   3   5  0.60     2  Own full ride-hailing app like the big ones
```

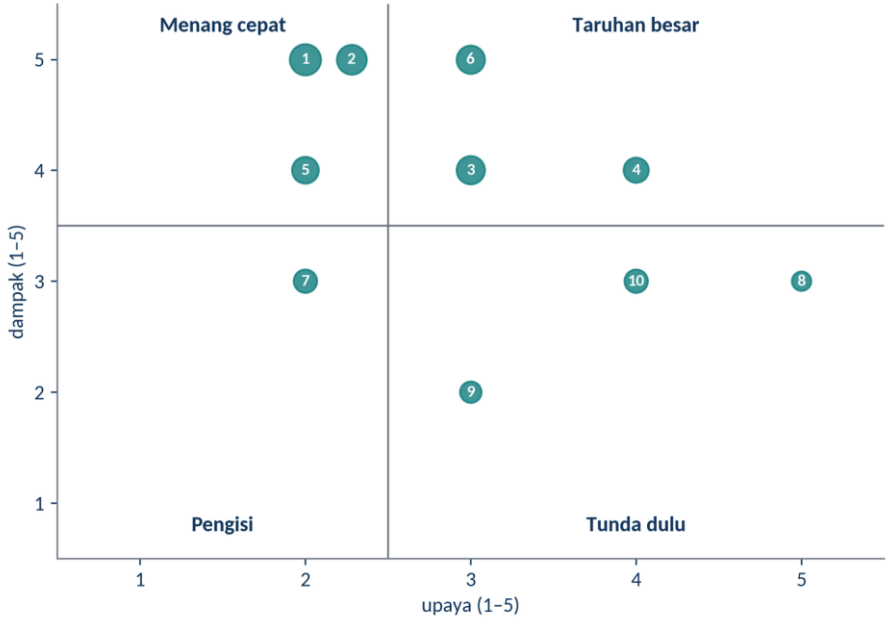
Papan afinitas memperlihatkan satu tema yang jelas di depan: *waktu tunggu tak pasti* (*Unknown waiting time*) diangkat oleh tujuh dari sepuluh orang, jangkauan $r = 7/10 = 70\%$ menurut Persamaan (4.1), baik oleh pengemudi maupun penumpang. Empat tema menyusul dengan masing-masing lima orang (50%). *Giliran yang adil* (*Fair turns*) hanya diangkat oleh pengemudi, dan justru karena itulah tema ini akan terlewat seandainya tim hanya mewawancarai penumpang.

Blok perjalanan, yang digambar sebagai kurva pada **Gambar 4.3**, menempatkan keluhan di paruh pertama perjalanan. Menyepakati tempat dan harga serta menunggu pengemudi sama-sama berrata-rata 3,6, dan mencari pengemudi 3,4; perjalanannya sendiri (2,8) dan pembayaran (2,4) paling sedikit menimbulkan keluhan. Untuk tahap *Agree*, misalnya, kesepuluh skor keluhannya adalah 4, 4, 3, 5, 4, 3, 4, 3, 3, dan 3, yang jumlahnya 36, sehingga $\bar{p} = 36/10 = 3.6$. Temuan ini layak disampaikan terus terang kepada Pak Darto: perjalanannya, bagian yang dibanggakan para pengemudi, sudah baik; yang menyakitkan adalah semua yang terjadi sebelumnya.



Gambar 4.3 Garis emosi peta perjalanan: rata-rata keluhan per tahap dari 37 catatan (sumbunya dibalik, sehingga kurva turun di tempat yang menyakitkan; pita berarsir menandai keluhan berat).

Tiga pertanyaan HMW memberi arah bagi sesi-sesi desain: menunjukkan kepada penumpang kapan pengemudi tiba, membuat tarif jelas sebelum perjalanan, dan memungkinkan penumpang memesan tanpa menelepon. Blok gagasan dan **Gambar 4.4** lalu mengurutkan kesepuluh gagasan. Tiga di antaranya menang cepat: tabel tarif tetap per zona (dampak 5, upaya 2, sembilan suara), pesan pengemudi sedang menuju dengan nama dan foto, dan tautan berbagi perjalanan untuk perjalanan malam. Tiga adalah taruhan besar yang layak direncanakan: antrean giliran otomatis yang adil, yang menjawab tema para pengemudi; halaman pemesanan satu ketukan; dan memilih titik jemput dari foto-foto patokan, yang menjawab "saya hanya kenal patokan" dari Mbah Sri. Adapun permintaan semula, "aplikasi transportasi lengkap milik sendiri seperti yang besar-besar", berakhir di baris terakhir, dengan nilai per upaya terendah dan dua suara dari seluruh kelompok.



Gambar 4.4 Kesepuluh gagasan dalam matriks dampak-upaya, dinomori sesuai urutan pada **Kode 4.3**; ukuran setiap lingkaran menunjukkan banyaknya suara.

Rekomendasi tim kepada Pak Darto mengalir langsung dari situ: mulailah dengan tiga gagasan menang cepat, yang bersama-sama menjawab tiga tema terkuat; rencanakan antrean yang adil

bersama para pengemudi; buat prototipe penjemputan dengan foto patokan bersama penumpang lanjut usia; dan jangan membangun aplikasi transportasi yang lengkap. Setiap rekomendasi disertai kebutuhan terukur seperti di Subbab 4.5.7, agar siklus berikutnya pada **Gambar 4.1** dapat memeriksa apakah rekomendasi itu berhasil.

4.9 Mengujinya

Kode 4.6 memeriksa keempat kuadran, memeriksa bahwa tema yang disebut dua orang berperingkat di atas tema dengan jumlah catatan yang sama dari satu orang, dan memeriksa rata-rata perjalanan pada sebuah contoh kecil.

Kode 4.6 `test_synthesis.py`: uji kuadran, peringkat tema, dan rata-rata perjalanan.

```
"""Uji sintesis riset: python -m unittest test_synthesis.py"""
import unittest
from synthesis import affinity, quadrant, journey


class SynthesisTests(unittest.TestCase):
    def test_quadrants(self):
        self.assertEqual(quadrant(5, 2), "Quick win")
        self.assertEqual(quadrant(4, 3), "Big bet")
        self.assertEqual(quadrant(3, 2), "Fill-in")
        self.assertEqual(quadrant(2, 5), "Avoid for now")

    def test_people_count_more_than_notes(self):
        notes = [dict(theme="A", participant="P1", role="passenger"),
                 dict(theme="A", participant="P1", role="passenger"),
                 dict(theme="B", participant="P1", role="passenger"),
                 dict(theme="B", participant="D1", role="driver")]
        self.assertEqual(affinity(notes)[0][0], "B")

    def test_journey_mean(self):
        notes = [dict(stage=s, pain=p) for s, p in
                 [("Find", "4"), ("Find", "2"), ("Agree", "3"), ("Wait", "5"),
                  ("Ride", "1"), ("Pay", "2")]]
        self.assertEqual(journey(notes)[0], ("Find", 2, 3.0))


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

Kode 4.7 menjalankan pemeriksaan yang bersesuaian dalam Java.

Kode 4.7 `SynthesisTest.java`: pemeriksaan yang sama dalam Java.

```java
import java.util.List;
import java.util.Map;

/** Uji sintesis riset: java -cp out SynthesisTest */
public class SynthesisTest {
    static int passed = 0;

    static void expect(boolean ok, String what) {
        if (!ok) throw new AssertionError(what);
        passed++;
    }

    public static void main(String[] args) {
        expect(Synthesis.quadrant(5, 2).equals("Quick win"), "quick win");
        expect(Synthesis.quadrant(4, 3).equals("Big bet"), "big bet");
        expect(Synthesis.quadrant(3, 2).equals("Fill-in"), "fill-in");
        expect(Synthesis.quadrant(2, 5).equals("Avoid for now"), "avoid");
        var notes = List.of(
            Map.of("theme", "A", "participant", "P1", "role", "passenger"),
            Map.of("theme", "A", "participant", "P1", "role", "passenger"),
            Map.of("theme", "B", "participant", "P1", "role", "passenger"),
            Map.of("theme", "B", "participant", "D1", "role", "driver"));
        var board = Synthesis.affinity(notes);
        expect(board.get(0).id().equals("B"), "two people beat two notes");
        System.out.println("All " + passed + " checks passed.");
    }
}
```

```
$ python3 -m unittest test_synthesis.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
$ java -cp out SynthesisTest
All 5 checks passed.
```

4.10 Memakainya Sungguh-sungguh

Tim kecil dapat menjalankan seluruh langkah perumusan sebuah proyek dalam satu hari dengan rencana pada **Tabel 4.4**. Rencana ini mengandaikan bahwa wawancara atau pengamatan sudah dilakukan dan setiap pengamatan sudah tertulis pada catatan.

Tabel 4.4 Rencana lokakarya sintesis satu hari beserta keluarannya.

Waktu	Kegiatan	Keluaran
08.00–09.30	Bacakan semua catatan; kelompokkan tanpa berbicara di dinding atau papan tulis	Kelompok afinitas
09.30–10.00	Namai kelompoknya; beri setiap catatan tahap perjalanan dan skor keluhan	Tema, `notes.csv`, `themes.csv`
10.00–10.30	Jalankan program sintesis; diskusikan jangkauan setiap tema dan kurva perjalanan	Tema berperingkat, peta perjalanan
10.30–12.00	Tulis persona dan pernyataan masalah untuk tema-tema terkuat; sepakati pertanyaan HMW	Persona, pernyataan masalah
13.00–14.30	Ideasi: banyak gagasan per pertanyaan HMW, tanpa menilai	Daftar gagasan
14.30–15.30	Beri skor dampak dan upaya; tunjukkan gagasan kepada partisipan dan kumpulkan suara	`ideas.csv`
15.30–16.30	Jalankan program lagi; pilih apa yang akan dibuat prototipenya; tulis kebutuhan terukur	Rencana prototipe, kebutuhan

Berkas dan programnya sengaja dibuat umum. Ganti tahap-tahapnya dengan tahap perjalanan Anda sendiri (untuk laundry: *pesan*, *jemput*, *cuci*, *antar*, *bayar*; untuk puskesmas: *datang*, *daftar*, *menunggu*, *periksa*, *apotek*), dan peran-perannya dengan peran Anda sendiri (pasien dan petugas, pelanggan dan kurir). Ada dua kehati-hatian. Pertama, angka-angka itu merangkum pertimbangan: skor keluhan 4 adalah bacaan tim atas sebuah catatan, dan peringkatnya hanya sebaik bacaan itu; diskusikan perbedaan pendapat alih-alih merata-ratakannya begitu saja. Kedua, tema dengan jangkauan rendah bisa saja kritis, misalnya masalah keselamatan yang disebut oleh satu perawat sif malam; angka mengarahkan perhatian, bukan menggantikannya.

4.11 IMK dalam Praktik: Profitina

Profitina menerapkan siklus pada **Gambar 4.1** pada produknya sendiri. *Alur umpan balik yang transparan* adalah sebuah lingkar berpusat pada pengguna yang dibuat terlihat: pengguna mengirim umpan balik atas jawaban AI, tim menyaringnya, butir yang tersaring menjadi dataset, model Qwen2.5 lokal dilatih dan diukur dengannya, lalu versi baru dirilis, dan siklus dimulai lagi. Pengguna dapat melihat sampai di mana umpan baliknya diproses, yang menjadikan mereka peserta dalam desain, bukan objeknya, sesuai prinsip kedua ISO 9241-210.

Keputusan produk juga mengikuti bukti tentang pemakaian. Paket langganan Agency dihapus pada Agustus 2026, sehingga tersisa satu langganan Pro 30 hari, sebuah penyederhanaan dari jenis yang sering direkomendasikan matriks dampak-upaya di Subbab 4.5.6: lebih sedikit pilihan, masing-masing dipahami dengan jelas. Dan aturan produk bahwa aplikasi memberi saran dan pengguna yang memutuskan adalah sebuah kebutuhan dalam pengertian Subbab 4.5.7: setiap saran hasil AI harus disajikan sedemikian rupa sehingga pemilik merek dapat menilai dan menolaknya, dan ketika model lokal tidak tersedia, aplikasi mengatakannya terus terang alih-alih diam-diam menggantinya dengan templat.

4.12 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

1. **Berangkat dari solusi.** “Aplikasi seperti yang besar-besar” adalah solusi. Mulailah dengan berlian pertama: temukan dan rumuskan sebelum mengembangkan.
2. **Menganggap permintaan klien sebagai kebutuhan.** Permintaan adalah hipotesis; orang yang akan memakai hasilnya yang menentukan kebutuhan.
3. **Mewawancarai hanya satu jenis pengguna.** Tema giliran yang adil dari para pengemudi tidak terlihat dalam wawancara penumpang. Libatkan setiap peran yang tersentuh sistem.
4. **Menghitung catatan alih-alih orang.** Satu partisipan yang banyak bicara dapat mendominasi sebuah tema; urutkan tema menurut jangkauannya (Persamaan (4.1)).
5. **Mengarang persona atau pengguna.** Persona tanpa bukti, atau pengguna sintetis dari alat AI, menggambarkan stereotipe. Bangunlah persona dari catatan.
6. **Menulis kebutuhan yang tidak dapat diuji.** “Mudah dipakai” adalah keinginan; “memesan dalam kurang dari 30 detik tanpa bantuan pada 4 dari 5 percobaan” adalah kebutuhan.
7. **Menjalankan prosesnya sekali saja.** Desain berpusat pada pengguna adalah siklus; rencanakan iterasi kedua sebelum iterasi pertama selesai.
8. **Salah menangani data partisipan.** Catatan wawancara adalah data pribadi menurut UU PDP; mintalah persetujuan, pakailah kode, dan hapus rekaman setelah analisis.

4.13 Ringkasan Bab

- Desain berpusat pada pengguna bertumpu pada tiga kebiasaan Gould dan Lewis: fokus awal pada pengguna dan tugas, pengukuran empiris, dan desain iteratif.
- ISO 9241-210:2019 memberikan enam prinsip (**Tabel 4.1**) dan empat aktivitas dalam sebuah siklus: konteks penggunaan, kebutuhan pengguna, solusi desain, dan evaluasi (**Gambar 4.1**).
- Double Diamond (*discover*, *define*, *develop*, *deliver*; **Gambar 4.2**) dan lima mode *design thinking* (*empathize*, *define*, *ideate*, *prototype*, *test*) menggambarkan siklus yang sama (**Tabel 4.2**); Lean UX menyesuaikannya dengan tim *agile*.
- Alat-alat kerjanya: diagram afinitas dan jangkauan tema (Persamaan (4.1)), persona (**Tabel 4.3**), peta perjalanan dengan rata-rata keluhan per tahap (Persamaan (4.2)), peta empati, *jobs to be done*, pernyataan masalah, pertanyaan HMW, matriks dampak-upaya dengan nilai per upaya (Persamaan (4.3)), dan kebutuhan yang terukur.
- Alat AI membantu mengembangkan dan membuat prototipe; alat itu tidak dapat menggantikan riset bersama orang sungguhan. Data riset adalah data pribadi menurut UU PDP dan PP 33/2026.
- Untuk pangkalan ojek, 37 catatan dari sepuluh orang menunjukkan bahwa menunggu tanpa kepastian (jangkauan 70%) dan menyepakati harga paling menyakitkan, dan mengubah “aplikasi seperti yang besar-besar” menjadi tiga gagasan menang cepat dan tiga taruhan yang direncanakan.

4.14 Soal Latihan

1. Dengan catatan partisipan P1 pada **Kode 4.2**, tuliskan sebuah persona dalam format **Tabel 4.3**. Kolom mana yang dapat Anda isi dari bukti, dan mana yang perlu Anda tanyakan?
2. Hitunglah dengan tangan, memakai Persamaan (4.2), rata-rata keluhan tahap *Wait* dari tujuh catatan pada **Kode 4.2**, dan jangkauan tema T5 dengan Persamaan (4.1).
3. Dalam sesi ideasi kedua, seorang pengemudi mengusulkan “kartu tarif tercetak di pangkalan dan di toko-toko”, yang diberi skor oleh tim dengan dampak 4, upaya 1, dan lima suara. Tambahkan ke salinan `ideas.csv`, jalankan programnya, lalu sebutkan di mana gagasan itu mendarat dan mengapa.
4. Perbaiki pertanyaan-pertanyaan “bagaimana kita bisa” berikut dan sebutkan apa yang salah pada masing-masing: (a) “Bagaimana kita bisa membangun aplikasi pemesanan untuk pangkalan?”; (b) “Bagaimana kita bisa memperbaiki transportasi di kota ini?”; (c) “Bagaimana kita bisa menambahkan peta dengan GPS?”.
5. Uraikan proyek ojek di bab ini dengan kosakata Double Diamond dan dengan kosakata ISO 9241-210. Di mana tim menyebar dan memusat, dan ke aktivitas mana evaluasi berikutnya kemungkinan besar akan mengirim mereka kembali?
6. Keponakan Pak Darto mengusulkan agar wawancara dilewati saja: “Minta AI membangkitkan dua puluh persona penumpang; lebih cepat.” Tuliskan balasan lima atau enam kalimat yang menjelaskan apa yang akan hilang dan apa yang dapat dikerjakan AI dengan berguna sebagai gantinya.
7. Kembangkan `synthesis.py` agar juga mencetak jangkauan setiap tema dalam persen dan menandai tema yang hanya diangkat oleh satu peran. Tunjukkan baris-baris barunya untuk data asli.

4.15 Penyelesaian Soal Latihan

1. Dari catatan P1: *konteks*: penumpang tetap yang menelepon pangkalan pagi-pagi (teleponnya sibuk pukul 7 pagi); *kekesalan*: telepon sibuk, harga tidak selalu sama, dan menunggu di depan pagar tanpa tahu apakah ada yang datang (keluhan 5, tertinggi dari semua catatan); *tujuan* (disimpulkan, perlu dikonfirmasi): sampai di tempat kerja atau sekolah tepat waktu pada pagi hari. Kutipan harus diambil dari wawancara itu sendiri, bukan dari khayalan; catatan-catatannya menunjuk pada kutipan tentang menunggu di depan pagar. Usia, pekerjaan, tujuan perjalanan, ponsel, dan kebiasaan membayar tidak ada dalam catatan dan harus ditanyakan; menuliskannya dari khayalan akan mengubah persona menjadi stereotipe. *Implikasi desain*: pemesanan yang tidak bergantung pada telepon yang sibuk dan konfirmasi tentang siapa yang datang dan kapan.

2. Catatan *Wait*: N03 (5), N08 (4), N13 (4), N20 (3), N29 (2), N35 (4), N36 (3). Jumlahnya 25, sehingga $\bar{p} = 25/7 = 3.57$, dicetak sebagai 3.6. Tema T5 punya catatan dari P3, P5, P6, dan D3, yaitu empat orang berbeda dari sepuluh, sehingga $r = 4/10 = 40\%$.

3. Setelah menambahkan baris `Printed fare card at the base and in shops,T1,4,1,5` dan menjalankan program, blok gagasannya menjadi:

```
quadrant      imp eff ratio votes idea
Quick win       4   1  4.00     5  Printed fare card at the base and in shops
Quick win       5   2  2.50     9  Fixed fare table by zone shown before booking
Quick win       5   2  2.50     8  Driver-on-the-way message with name and photo
Quick win       4   2  2.00     6  Share-my-ride link for family on night trips
Big bet         5   3  1.67     7  Automatic fair queue for the base
Big bet         4   3  1.33     7  One-tap booking page from a saved WhatsApp link
Big bet         4   4  1.00     5  Pick the pickup point from photos of landmarks
Fill-in         3   2  1.50     4  Cooperative QRIS with the amount shown
Avoid for now   3   4  0.75     4  Voice booking by phone with a menu
Avoid for now   2   3  0.67     3  Loyalty points for regular passengers
Avoid for now   3   5  0.60     2  Own full ride-hailing app like the big ones
```

Kartu itu mendarat di puncak: ia menang cepat (dampak 4 tergolong tinggi, upaya 1 tergolong rendah), dan nilai per upayanya, $4/1 = 4.00$, adalah yang tertinggi dari semua gagasan. Kartu itu juga melayani tema *harga tidak jelas* bagi penumpang tanpa ponsel pintar, yang tidak dijangkau oleh tabel tarif sebelum memesan, sehingga ia melengkapi, bukan menggantikan, gagasan itu.

4. (a) Menyebut solusi ("aplikasi pemesanan") tanpa kebutuhan; lebih baik: "Bagaimana kita bisa memungkinkan penumpang tetap memesan perjalanan tanpa menelepon pangkalan?". (b) Terlalu luas untuk ditindaklanjuti tim; lebih baik: "Bagaimana kita bisa membantu penumpang lanjut usia sampai di pasar sebelum pukul 05.30 tanpa bergantung pada keluarganya?". (c) Menyebut teknologi dan mengabaikan temuan bahwa banyak penumpang tidak dapat membaca peta; lebih baik: "Bagaimana kita bisa membantu penumpang menyebutkan di mana mereka berada tanpa nama jalan?".

5. Double Diamond: *discover*, tiga pagi di pangkalan dan sepuluh percakapan (menyebar); *define*, diagram afinitas, persona, peta perjalanan, dan tiga pertanyaan HMW (memusat pada menunggu, harga, dan pemesanan); *develop*, sesi ideasi dengan sepuluh gagasan (menyebar); *deliver*, memilih tiga gagasan menang cepat untuk dibuat prototipenya dan diuji (memusat). ISO 9241-210: memahami konteks penggunaan (pangkalan, ponsel, penumpang), menetapkan kebutuhan (pernyataan terukur di Subbab 4.5.7), menghasilkan solusi desain (prototipe yang akan dibuat), dan mengevaluasinya. Evaluasi pertama kemungkinan besar akan mengirim tim kembali ke *menghasilkan solusi desain* untuk rincian pesan-pesannya, dan mungkin ke *konteks penggunaan* bila prototipe mengungkap, misalnya, bahwa penumpang lanjut usia tidak membaca pesan WhatsApp ketika berdiri di bawah terik.

6. Persona bangkitan AI menggambarkan apa yang lazim bagi penumpang pada umumnya, sebagaimana tercermin dalam data latih model, bukan para pelanggan tetap pangkalan ini. Persona semacam itu tidak mungkin mengungkap bahwa kekhawatiran utama para pengemudi adalah keadilan urutan giliran, bahwa penumpang lanjut usia hanya mengenal tempat lewat patokan, atau bahwa menunggu lebih menyakitkan daripada memesan; temuan-temuan itu hanya datang dari mendengarkan orang-orang ini, dan temuan itulah yang mengubah seluruh rencana. Membangun di atas pengguna karangan berisiko menghabiskan uang koperasi untuk fitur yang tidak diperlukan siapa pun. AI dapat membantu dengan berguna menyiapkan panduan wawancara, meringkas catatan tim sendiri yang sudah dianonimkan, atau menyusun varian pesan setelah kebutuhannya diketahui. Riset itu sendiri, tiga pagi di pangkalan, murah dibandingkan dengan membangun aplikasi yang salah.

7. Di `main`, setelah `board = affinity(notes)`, ubah baris tema untuk menambahkan jangkauan dan menandai tema dengan satu peran:

```
    for theme, t in board:
        who = "+".join(sorted(t["roles"], reverse=True))
        reach = 100 * len(t["people"]) / len(people)
        mark = "  <- one role only" if len(t["roles"]) == 1 else ""
        print(f"{theme:5} {themes[theme]['label']:22}{t['notes']:6}"
              f"{len(t['people']):7}  {reach:3.0f}%  {who}{mark}")
```

Untuk data asli, baris-barisnya kemudian menunjukkan 70% untuk T2; 50% untuk T1, T3, T4, dan T7; 40% untuk T5; 30% untuk T6; dan 30% untuk T8, satu-satunya tema yang ditandai “one role only” (pengemudi).

Daftar Pustaka

[1] J. D. Gould and C. Lewis, “Designing for usability: Key principles and what designers think,” *Communications of the ACM*, vol. 28, no. 3, pp. 300–311, 1985.

[2] International Organization for Standardization, *ISO 9241-210:2019 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. Geneva, Switzerland: ISO, 2019. [Online]. Available: https://www.iso.org/standard/77520.html

[3] Design Council, “Framework for Innovation: Design Council’s evolved Double Diamond,” accessed Sep. 2026. [Online]. Available: https://www.designcouncil.org.uk/resources/framework-for-innovation/

[4] Design Council, “Systemic Design Framework,” 2021, accessed Sep. 2026. [Online]. Available: https://www.designcouncil.org.uk/resources/systemic-design-framework/

[5] T. Brown, “Design Thinking,” *Harvard Business Review*, vol. 86, no. 6, pp. 84–92, Jun. 2008.

[6] Hasso Plattner Institute of Design at Stanford, *An Introduction to Design Thinking: Process Guide*. Stanford, CA, USA: d.school, 2010.

[7] J. Gothelf and J. Seiden, *Lean UX: Creating Great Products with Agile Teams*, 3rd ed. Sebastopol, CA, USA: O’Reilly Media, 2021.

[8] K. Holtzblatt and H. Beyer, *Contextual Design: Design for Life*, 2nd ed. Cambridge, MA, USA: Morgan Kaufmann, 2017.

[9] A. Cooper, R. Reimann, D. Cronin, and C. Noessel, *About Face: The Essentials of Interaction Design*, 4th ed. Indianapolis, IN, USA: Wiley, 2014.

[10] C. M. Christensen, T. Hall, K. Dillon, and D. S. Duncan, “Know Your Customers’ ‘Jobs to Be Done’,” *Harvard Business Review*, vol. 94, no. 9, pp. 54–62, Sep. 2016.

[11] Figma, “What’s new from Config 2026,” Figma Help Center, Jun. 2026. [Online]. Available: https://help.figma.com/hc/en-us/articles/39582753756695-What-s-new-from-Config-2026

[12] Nielsen Norman Group, "Synthetic Users: If, When, and How to Use AI-Generated 'Research'," accessed Sep. 2026. [Online]. Available: https://www.nngroup.com/articles/synthetic-users/

[13] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. JDIH BPK RI. [Online]. Available: https://peraturan.bpk.go.id/Details/229798/uu-no-27-tahun-2022

[14] Veritask, "Pengaturan Teknis Pelindungan Data Pribadi dan Kewajiban Pengendali serta Prosesor Akhirnya Terbit Lewat PP 33/2026," 2026. [Online]. Available: https://veritask.ai/id/artikel/pengaturan-teknis-pelindungan-data-pribadi-dan-kewajiban-pengendali-serta-prosesor-akhirnya-terbit-lewat-pp-33-2026

Bab 5

Metode Riset Pengguna

Sebuah laundry kecil di dekat kampus kehilangan pelanggan, dan tidak ada yang tahu sebabnya. Keponakan pemiliknya ingin membuat aplikasi; pemiliknya ingin menurunkan harga; seorang teman bilang toko itu butuh papan nama yang lebih besar. Semuanya menebak. Bab ini membahas cara mengganti tebakan dengan bukti, secara murah dan jujur. Di akhir bab Anda akan mampu memilih metode riset berdasarkan pertanyaan yang perlu dijawab; merencanakan studi kecil lengkap dengan rencana riset, *screener* rekrutmen, dan teks persetujuan yang menghormati UU PDP 27/2022 beserta peraturan pelaksananya yang baru, PP 33/2026; melakukan wawancara dan inkuiri kontekstual tanpa menggiring partisipan; menyusun survei singkat yang butirnya benar-benar mengukur apa yang Anda kira diukurnya; merangkum survei dengan Java dan Python lengkap dengan rata-rata, selang kepercayaan, dan proporsi *top-2-box*; mengodekan catatan wawancara, menyusun diagram afinitas, dan memutuskan dengan aturan yang jelas kapan wawancara sudah cukup; serta mengubah temuan menjadi pernyataan wawasan yang dapat ditindaklanjuti perancang. Anda juga akan melihat di mana AI generatif membantu peneliti pada 2026 dan di mana ia diam-diam merugikan.

5.1 Pelanggan Bu Rina yang Menghilang

Bu Rina mengelola *Laundry Bersih*, usaha laundry dengan dua mesin cuci, dua pengering, dan satu meja setrika panjang, di sebuah gang dekat kampus di Surabaya. Pelanggannya mahasiswa dari rumah-rumah kos di sekitar kampus, pekerja kantoran muda, beberapa keluarga, dan pemilik kos sendiri yang setiap Senin membawa seprai dua puluh kamar. Sebagian besar pelanggan memesan lewat WhatsApp: mereka mengirim pesan, Dimas, karyawan Bu Rina, datang menjemput dengan sepeda motor, pakaian ditimbang di toko, lalu pelanggan menerima pesan berisi berat dan harganya. Dua hari kemudian pakaian kembali, terlipat rapi dan terbungkus plastik.

Selama tiga tahun usaha itu tumbuh. Tahun ini jumlah pesanan turun empat bulan berturut-turut, dan di buku pesanannya Bu Rina menghitung pelanggan tetap berkurang sekitar sepertiga dibandingkan setahun sebelumnya. Ia punya dugaan tentang penyebabnya, begitu pula orang-orang di sekitarnya. Keponakannya, mahasiswa informatika, yakin pelanggan menginginkan aplikasi dengan peta dan layar pelacakan seperti aplikasi transportasi dan pesan-antar makanan yang besar. Seorang tetangga menduga laundry baru di ujung jalan lebih murah. Dimas mengira pelanggan kesal karena ia kadang datang larut malam. Bu Rina sendiri curiga mahasiswa pulang kampung saat libur panjang dan belum semuanya kembali.

Setiap penjelasan itu masuk akal, dan setiap penjelasan membawa ke investasi yang berbeda. Aplikasi menghabiskan waktu berbulan-bulan dan biaya akun pengembang serta server. Harga yang lebih rendah memotong margin setiap kilogram. Sepeda motor kedua berarti satu gaji tambahan. Jika penyebab sebenarnya lain lagi, setiap investasi itu sia-sia, dan pelanggan tetap pergi. Yang dibutuhkan Bu Rina sebelum mengeluarkan uang adalah *mencari tahu*. Itulah riset pengguna: cara yang disiplin untuk mengetahui apa yang dilakukan orang, apa yang mereka butuhkan, dan mengapa, sebelum dan selama kita merancang untuk mereka.

Kebutuhan ini tidak kecil. Dalam survei yang dirilis Mei 2026, Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) memperkirakan 81,7% penduduk Indonesia, sekitar 235 juta dari 287 juta jiwa, menggunakan internet, naik dari 80,66% setahun sebelumnya; penetrasinya 84,75% di perkotaan dan 78,18% di perdesaan, dan kegiatan yang paling umum adalah komunikasi dan media sosial, hiburan, pencarian informasi, serta perdagangan daring [1]. Usaha sekecil milik Bu Rina kini bertemu pelanggannya lewat layar, dan setiap keputusan tentang layar itu adalah keputusan tentang orang-orang yang tidak dapat ia lihat.

Angka kedua menunjukkan mengapa riset tetap perlu hati-hati meskipun data berlimpah. StatCounter, yang menghitung tayangan halaman di situs-situs yang memasang kode pelacaknya, melaporkan untuk Agustus 2026 bahwa 60,59% tayangan halaman yang dilihatnya dari Indonesia berasal dari komputer desktop dan 38,98% dari ponsel [2]. Sekilas ini bertentangan dengan semua yang kita ketahui tentang negeri tempat kebanyakan orang pertama kali mengenal internet lewat ponsel. Sebenarnya tidak bertentangan; ini pengukuran yang berbeda. APJII bertanya kepada *orang*; StatCounter menghitung *tayangan halaman* di sekumpulan situs tertentu, dan tayangan dari komputer kantor, dari lalu lintas otomatis, dan dari segelintir pengguna berat berbobot sama dengan milik siapa pun. Tidak ada angka yang salah. Masing-masing menjawab pertanyaannya sendiri, dan peneliti yang mengutip salah satunya harus tahu pertanyaan mana yang dijawab. Dalam arti tertentu, seluruh bab ini membahas pelajaran itu.

5.2 Gagasannya dalam Bahasa Sederhana

5.2.1 Mencari tahu sebelum membangun

Standar perancangan berpusat pada manusia, ISO 9241-210:2019, menempatkan gagasan ini di awal prosesnya: sebelum merancang solusi, kita harus *memahami dan merumuskan konteks penggunaan*, yaitu pengguna, tujuan dan tugas mereka, serta lingkungan fisik, sosial, dan teknis tempat mereka bekerja [3]. Bab 4 menempatkan kegiatan ini dalam siklus *design thinking* sebagai berlian pertama, bagian ketika tim memperluas pandangannya tentang masalah sebelum menyempit ke sebuah rumusan. Bab ini menyediakan alat untuk bagian tersebut.

> **DEFINISI.** *Riset pengguna* (*user research*) adalah kajian sistematis tentang orang-orang yang menggunakan (atau mungkin menggunakan) sebuah produk: perilaku, kebutuhan, tujuan, konteks, dan sikap mereka. *Pertanyaan riset* adalah hal yang perlu diketahui tim, misalnya "mengapa pelanggan tetap berhenti memesan?". *Metode* adalah prosedur untuk menjawabnya, misalnya wawancara atau survei. *Partisipan* adalah orang yang ikut serta dalam studi, dan *temuan* adalah sesuatu yang ditunjukkan studi, dinyatakan bersama buktinya.

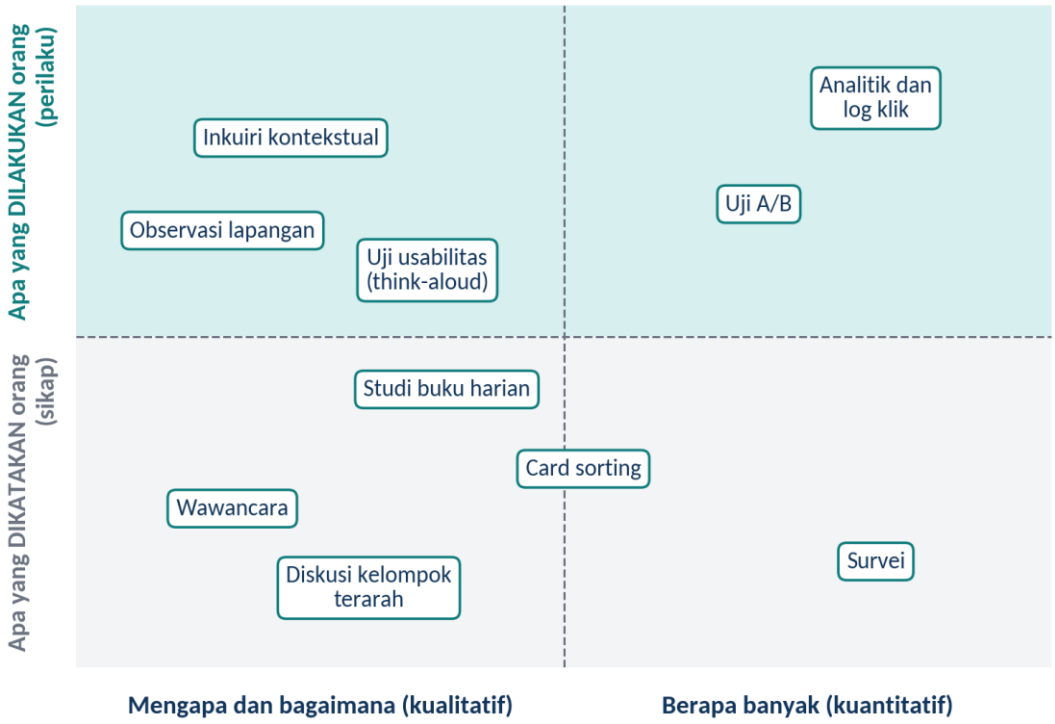
Sebuah studi dimulai dari pertanyaan riset, tidak pernah dari metode. "Ayo kita survei" bukan rencana; "kita perlu tahu mana dari empat dugaan penyebab yang paling menjelaskan hilangnya pelanggan tetap" adalah rencana. Pertanyaan itulah yang kemudian menentukan metode, partisipan, dan seberapa banyak bukti yang cukup.

5.2.2 Apa yang dikatakan dan apa yang dilakukan

Manusia adalah saksi yang buruk atas perilakunya sendiri. Tanyakan kepada seorang pelanggan seberapa sering ia mencucikan pakaian, dan ia akan menjawab "tiap minggu"; riwayat pesanannya mungkin menunjukkan tiap sepuluh hari, dengan jeda sebulan pada bulan Juli. Tanyakan apakah ia akan memakai aplikasi dengan pelacakan pesanan, dan ia akan dengan sopan menjawab ya, karena kedengarannya modern dan menjawab tidak terasa kurang enak. Keduanya bukan kebohongan. Orang mengingat kasus yang lazim, bukan kasus yang sebenarnya, dan

mereka buruk dalam meramalkan perilakunya sendiri, terutama tentang hal yang belum pernah mereka coba.

Itulah sebabnya peta metode riset yang paling berguna memiliki dua sumbu [4]. Sumbu pertama memisahkan apa yang *dikatakan* orang (metode sikap atau *attitudinal*: wawancara, survei, diskusi kelompok terarah) dari apa yang *dilakukan* orang (metode perilaku atau *behavioral*: observasi, uji usabilitas, analitik). Sumbu kedua memisahkan pertanyaan *mengapa dan bagaimana* (metode kualitatif, sedikit partisipan yang dikaji mendalam) dari pertanyaan *berapa banyak* (metode kuantitatif, banyak partisipan yang diukur dengan cara yang sama). **Gambar 5.1** menempatkan metode-metode yang umum pada peta ini.



Gambar 5.1 Lanskap metode riset pengguna: metode perilaku (atas) mengamati apa yang dilakukan orang, metode sikap (bawah) mencatat apa yang dikatakan orang; metode kualitatif (kiri) menjelaskan mengapa, metode kuantitatif (kanan) menghitung berapa banyak.

Tidak ada kuadran yang lebih baik daripada yang lain. Metode perilaku kualitatif, seperti ikut Dimas berkeliling menjemput cucian dan mengamati pelanggan menyerahkan kantongnya, memperlihatkan masalah yang tidak akan terpikir untuk disebutkan siapa pun. Metode sikap kuantitatif, seperti survei terhadap enam puluh pelanggan, memberi tahu seberapa luas sebuah masalah, tetapi tidak menjelaskan mengapa masalah itu ada. Studi yang baik biasanya memadukan dua kuadran, dan paduan ini disebut *triangulasi*: ketika wawancara, survei, dan buku pesanan menunjuk ke penyebab yang sama, kita dapat bertindak dengan yakin.

5.2.3 Empat pertanyaan untuk memilih metode

Sebelum memilih metode, jawablah empat pertanyaan secara tertulis.

1. **Keputusan apa yang akan dibantu oleh riset ini?** Bu Rina harus memutuskan ke mana uangnya dibelanjakan dalam tiga bulan ke depan. Riset yang tidak dapat mengubah keputusan tidak layak dilakukan.

2. **Apa yang sudah kita ketahui, dan dari mana kita mengetahuinya?** Buku pesanan menunjukkan pelanggan tetap berkurang; itu data perilaku dan kuantitatif, tetapi tidak berkata apa pun tentang sebabnya.
3. **Apakah pertanyaannya tentang mengapa atau tentang berapa banyak?** "Mengapa pelanggan pergi?" bersifat kualitatif dan membutuhkan beberapa percakapan mendalam. "Berapa banyak pelanggan yang terdampak penjemputan yang terlambat?" bersifat kuantitatif dan membutuhkan hitungan.
4. **Apa yang sanggup kita lakukan?** Bu Rina punya beberapa malam, daftar nomor telepon pelanggan, dan anggaran untuk voucer ucapan terima kasih. Itu tidak cukup untuk panel besar, tetapi cukup untuk selusin wawancara dan survei singkat.

Tabel 5.1 mengubah jawaban-jawaban itu menjadi pilihan pertama. Tabel ini bukan hukum; ini jenis tabel yang biasa ditempel tim riset di dinding.

Tabel 5.1 Memilih metode pertama berdasarkan pertanyaan riset, dengan ukuran sampel dan durasi yang lazim untuk tim kecil.

Jika pertanyaannya…	Mulailah dengan	Ukuran lazim	Waktu untuk tim kecil
Mengapa orang melakukan ini? Seperti apa hari-hari mereka?	Inkuiri kontekstual, wawancara	6–15 orang	1–2 minggu
Seberapa luas sebuah masalah atau pendapat?	Survei	50–400 jawaban	1–3 minggu
Dapatkah orang menyelesaikan tugas dengan rancangan kita?	Uji usabilitas (Bab 7)	5–8 per putaran	beberapa hari
Bagaimana orang mengelompokkan konten kita?	*Card sorting* (Bab 8)	15–30 orang	1 minggu
Apa yang sebenarnya terjadi selama berhari-hari atau berminggu-minggu?	Studi buku harian	8–15 orang	2–4 minggu
Mana dari dua rancangan yang berkinerja lebih baik?	Uji A/B (Bab 7)	ratusan hingga ribuan	1–4 minggu

Bagi Bu Rina, pertanyaan "mengapa pelanggan tetap berhenti memesan?" adalah pertanyaan mengapa, sehingga studi dimulai dengan wawancara dan satu pagi inkuiri kontekstual. Setelah wawancara mengarah ke penyebab yang paling mungkin, survei singkat akan mengukur seberapa luas masing-masing penyebab. Urutan ini, kualitatif dahulu untuk menemukan lalu kuantitatif untuk mengukur, adalah bentuk studi kecil yang paling umum, dan bentuk itulah yang diikuti bab ini.

5.3 Merencanakan Studi

5.3.1 Rencana riset satu halaman

Rencana riset adalah satu halaman yang dapat dibaca siapa pun di tim dalam dua menit. Nilainya bukan pada dokumennya, melainkan pada percakapan yang dipaksanya terjadi sebelum ada yang berbicara dengan pelanggan. **Tabel 5.2** adalah rencana untuk Laundry Bersih.

Tabel 5.2 Rencana riset satu halaman untuk studi Laundry Bersih.

Bagian	Studi Laundry Bersih
Keputusan	Ke mana berinvestasi dalam tiga bulan ke depan: harga, kapasitas penjemputan, komunikasi, atau aplikasi
Pertanyaan riset	Mengapa pelanggan tetap berhenti memesan? Apa yang terjadi antara pesan pemesanan dan kembalinya pakaian? Apa yang ingin diketahui atau dilakukan pelanggan?
Metode	12 wawancara semi-terstruktur (30–40 menit); satu pagi inkuiri kontekstual pada rute penjemputan; lalu survei 6 butir yang dikirim ke semua pelanggan di daftar WhatsApp
Partisipan	Pelanggan yang baru-baru ini berhenti memesan dan pelanggan tetap saat ini; campuran mahasiswa, pekerja, keluarga, pemilik kos; sekurang-kurangnya dua orang berusia 60 tahun ke atas
Rekrutmen	Pesan WhatsApp dari Bu Rina; *screener* empat pertanyaan; voucer cuci gratis 3 kg
Etika dan data	Teks persetujuan dibacakan dan disetujui sebelum perekaman; rekaman dihapus setelah catatan diperiksa; catatan hanya memakai kode partisipan
Keluaran	Catatan berkode, diagram afinitas, ringkasan survei, lima pernyataan wawasan, rekomendasi satu halaman
Jadwal	Minggu 1 rekrutmen dan wawancara; minggu 2 menyelesaikan wawancara dan analisis; minggu 3 survei dan laporan

5.3.2 Siapa yang diajak bicara, dan berapa banyak

Orang yang paling ingin kita dengar sering kali justru paling sulit dijangkau. Pelanggan yang sudah pergi tidak ada di toko, dan sebagian dari mereka kesal. Namun studi yang hanya berbicara dengan pelanggan setia yang puas akan belajar mengapa orang bertahan, bukan mengapa mereka pergi. Karena itu rencana Bu Rina sengaja meminta kedua kelompok. *Screener*, daftar pertanyaan singkat yang dikirim sebelum mengundang seseorang, memastikan campurannya tepat: kapan terakhir Anda memesan, seberapa sering Anda memesan sebelumnya, apa kegiatan Anda (mahasiswa, pekerja, rumah tangga, usaha), dan apakah Anda bersedia berbincang sekitar setengah jam. Orang yang menjawab “saya belum pernah memesan” diberi ucapan terima kasih dan tidak diundang.

Libatkan kelompok di pinggiran. Laundry di dekat kampus tidak hanya dipakai anak berusia dua puluhan. Tetangga yang sudah pensiun, yang WhatsApp-nya dipasangkan oleh anak mereka, mungkin kesulitan membaca daftar harga berhuruf kecil atau lebih nyaman mengirim pesan suara daripada mengetik. Pelanggan penyandang disabilitas dilindungi di Indonesia oleh UU 8/2016 tentang Penyandang Disabilitas, yang menjamin hak mereka atas layanan dan informasi yang aksesibel [5]; studi yang tidak melibatkan mereka akan merancang tanpa mereka. Bab 11 kembali ke soal aksesibilitas; di sini aturannya sederhana: rekrut dengan sengaja orang-orang yang pengalamannya berbeda dari pelanggan pada umumnya.

Berapa wawancara yang cukup? Untuk studi terhadap satu kelompok yang cukup seragam, sebuah analisis terhadap studi-studi wawancara yang banyak dikutip menemukan bahwa sebagian besar tema muncul dalam enam sampai dua belas wawancara pertama, dan analisis itu mengusulkan aturan sederhana untuk memutuskan kapan berhenti, yang di Subbab 5.7 kita ubah menjadi program [6]. Untuk survei, jawabannya bergantung pada ketelitian yang dibutuhkan, yang dihitung di Subbab 5.5. Untuk uji usabilitas, Bab 7 menunjukkan mengapa lima sampai delapan orang per putaran biasanya sudah mengungkap sebagian besar masalah serius.

5.3.3 Persetujuan dan data pribadi

Riset dengan manusia menangani data pribadi: nama, nomor telepon, suara dalam rekaman, kadang foto rumah. Di Indonesia, UU 27/2022 tentang Pelindungan Data Pribadi (UU PDP) mensyaratkan dasar pemrosesan yang sah, dan persetujuan eksplisit adalah dasar yang lazim dipakai studi riset [7]. Peraturan pelaksananya, Peraturan Pemerintah (PP) 33/2026, merinci seperti apa persetujuan yang sah: harus diberikan secara bebas, sadar, spesifik, dan tidak ambigu; informasi harus diberikan *sebelum* persetujuan diminta, secara ringkas dan akurat; persetujuan tidak boleh digabungkan dengan syarat dan ketentuan umum atau diperoleh dengan cara yang menipu; orang yang menarik persetujuannya harus melihat pemrosesan dihentikan dalam 72 jam setelah permintaan yang terverifikasi; dan menolak memberi persetujuan tidak boleh menurunkan mutu layanan yang tidak membutuhkan data tersebut [8]. Kewajiban rincinya berlaku mulai 16 Januari 2027 [9], tetapi studi yang dijalankan hari ini sebaiknya sudah bekerja dengan cara ini, karena memang begitulah cara yang pantas memperlakukan partisipan.

Dalam praktik, teks persetujuan untuk wawancara terdiri atas enam bagian, dibacakan atau dikirim sebagai pesan singkat dan disetujui sebelum perekaman dimulai:

1. siapa yang melakukan studi dan untuk apa (“untuk memperbaiki layanan pemesanan dan penjemputan Laundry Bersih”);
2. apa yang akan terjadi (“percakapan sekitar tiga puluh menit tentang cara Anda memakai laundry”);
3. apa yang direkam dan siapa yang melihatnya (“hanya suara, didengar Bu Rina dan satu pembantu, dihapus setelah catatan diperiksa; catatan memakai kode, bukan nama Anda”);
4. bahwa keikutsertaan bersifat sukarela dan dapat dihentikan kapan saja, tanpa pengaruh apa pun terhadap layanan;
5. ucapan terima kasih (“voucer cuci tiga kilogram, baik Anda menyelesaikan wawancara maupun tidak”);
6. sebuah pertanyaan: “Apakah Anda setuju ikut serta, dan apakah Anda setuju suara Anda direkam?”, dengan kedua jawaban dicatat terpisah.

Butir terakhir lebih penting daripada kelihatannya. Setuju untuk berbincang dan setuju untuk direkam adalah dua persetujuan yang berbeda; seseorang bisa dengan senang hati berbincang tetapi tidak mau direkam. Syarat PP 33/2026 bahwa persetujuan harus spesifik berarti keduanya harus ditanyakan terpisah, tidak dilipat menjadi satu jawaban ya [8].

> **PERINGATAN:** Jangan mengumpulkan apa yang tidak Anda butuhkan. Studi tentang pengalaman memesan tidak membutuhkan alamat, rincian keluarga, atau penghasilan partisipan. Setiap isian tambahan adalah risiko bagi partisipan dan kewajiban bagi Anda. Gunakan kode partisipan (P01, P02, …) sejak awal, dan simpan daftar yang menghubungkan kode dengan nomor telepon di satu tempat, terpisah dari catatan.

5.4 Wawancara dan Inkuiri Kontekstual

5.4.1 Wawancara semi-terstruktur

Wawancara dalam riset pengguna bukanlah kuesioner yang dibacakan. Ia adalah percakapan terarah: peneliti memegang daftar topik dan beberapa pertanyaan pembuka, tetapi mengikuti ke mana pun pengalaman partisipan membawa. Bentuk ini disebut *semi-terstruktur*. Nasihat Portigal, yang disarikan dari pengalaman bertahun-tahun mewawancarai, adalah bahwa tugas

peneliti menjadikan partisipan sebagai ahlinya dan jauh lebih banyak mendengar daripada berbicara [10].

Panduan wawancara yang baik untuk Bu Rina terdiri atas empat bagian: pemanasan ("ceritakan sedikit tentang minggu Anda, tempat tinggal Anda, dan bagaimana urusan cucian masuk ke dalamnya"), kisah kali terakhir ("ceritakan saat terakhir Anda mencucikan pakaian di sini, sejak Anda memutuskannya"), topik-topik khusus (penjemputan, harga, menunggu, menerima kembali pakaian, membayar, masalah), dan penutup ("kalau Anda boleh mengubah satu hal tentang mencucikan pakaian, apa itu?", "adakah yang seharusnya saya tanyakan?").

Jantung wawancara adalah kisah sebuah kejadian *spesifik* yang baru terjadi. Hal umum menghasilkan jawaban umum; hari Selasa tertentu menghasilkan detail. Ketika partisipan berkata "biasanya sih lancar", peneliti bertanya "ceritakan saat terakhir yang tidak lancar".

5.4.2 Bertanya tanpa menggiring

Pertanyaan itu sendiri adalah tempat kebanyakan wawancara keliru. **Tabel 5.3** memuat jebakan yang umum dan bentuk yang lebih baik untuk masing-masing.

Tabel 5.3 Jebakan wawancara dan pertanyaan yang lebih baik untuk studi laundry.

Jebakan	Pertanyaan buruk	Pertanyaan yang lebih baik
Menggiring	"Jam penjemputannya terlalu sempit, kan?"	"Ceritakan bagaimana Anda mengatur penjemputan kemarin."
Meminta ramalan	"Apakah Anda akan memakai aplikasi dengan pelacakan?"	"Apa yang Anda lakukan sambil menunggu cucian kemarin?"
Meminta rancangan	"Fitur apa yang harus ada di aplikasi kami?"	"Apa yang menjengkelkan atau mencemaskan pada pesanan terakhir?"
Pertanyaan ganda	"Apakah harganya jelas dan wajar?"	"Bagaimana Anda tahu harganya kemarin?" lalu "Apa pendapat Anda tentang harga itu?"
Ya/tidak yang menutup	"Apakah pengantarannya tepat waktu?"	"Kapan pakaian Anda kembali, dan bagaimana itu cocok dengan hari Anda?"
Jargon	"Bagaimana alur *order fulfillment*-nya?"	"Apa yang terjadi setelah Anda mengirim pesan?"

Tiga kebiasaan membantu lebih daripada daftar apa pun. Pertama, *diam*: setelah sebuah jawaban, hitung sampai lima sebelum berbicara; orang cenderung mengisi keheningan dengan bagian yang paling menarik. Kedua, *gema*: ulangi kata-kata terakhir partisipan sebagai pertanyaan ("harganya datang belakangan?") untuk mengundang cerita lebih banyak tanpa menyarankan apa pun. Ketiga, *minta artefaknya*: "boleh saya lihat chat-nya?" Percakapan WhatsApp di ponsel partisipan sendiri memperlihatkan jam, kata-kata, dan saat kekesalan yang persis, jauh lebih baik daripada ingatan.

5.4.3 Inkuiri kontekstual: guru dan murid magang

Inkuiri kontekstual (*contextual inquiry*) adalah wawancara yang berlangsung di tempat pekerjaan terjadi, ketika pekerjaan itu terjadi. Holtzblatt dan Beyer menggambarkan hubungannya seperti *guru dan murid magang*: partisipan adalah ahli pekerjaannya, peneliti adalah murid yang mengamati, bertanya, dan memeriksa pemahamannya, di tempat yang sebenarnya [11]. Empat prinsip menuntunnya: *konteks* (datangi tempat kerjanya), *kemitraan* (partisipan memimpin), *interpretasi* (sampaikan tafsiran Anda dan biarkan partisipan meluruskannya), dan *fokus* (ketahui bagian pekerjaan mana yang ingin Anda pelajari).

Untuk Laundry Bersih, inkuiri itu berupa satu pagi membonceng sepeda motor Dimas dan satu sore di meja kasir. Pagi itu memperlihatkan sesuatu yang tidak disebut dalam wawancara mana pun: di tiga rumah kos, pelanggannya sedang tidak di rumah, dan kantong cucian dititipkan kepada penjaga kos, yang tidak bisa memberi tahu kapan cucian kembali. Sore itu memperlihatkan ujung yang lain: pesan harga baru dikirim setelah penimbangan, sering tiga atau empat jam setelah penjemputan, dan pesan "sudah selesai" baru dikirim ketika Dimas hendak berangkat, kadang setelah pukul sembilan malam. Tidak ada yang mengeluhkan hal ini dengan kata-kata, tetapi riwayat chat di ponsel pelanggan penuh dengan "Bu, sudah selesai belum?".

5.4.4 Catatan lapangan dan studi buku harian

Catatan dibuat selama atau segera setelah setiap sesi, dalam format tetap agar dapat dibandingkan: kode partisipan, tanggal, tempat, lalu satu pengamatan per baris, masing-masing ditandai sebagai *kutipan* (kata-kata persis partisipan), *pengamatan* (apa yang Anda lihat), atau *tafsiran* (apa yang Anda pikir artinya). Memisahkan ketiganya adalah kebiasaan terpenting seorang peneliti, karena tafsiran kelak akan dipertanyakan, dan kutipan serta pengamatanlah buktinya.

Ketika perilaku yang dikaji berlangsung berhari-hari, *studi buku harian* (*diary study*) meminta partisipan mencatat entri singkat setiap kali sesuatu terjadi: setiap pesanan, setiap penantian, setiap masalah, misalnya berupa pesan suara atau foto di grup WhatsApp yang dibuat untuk studi. Studi buku harian mengungkap irama sehari-hari yang dipadatkan oleh wawancara satu jam: para pemilik kos pelanggan Bu Rina, misalnya, mencatat bahwa mereka memesan setiap Senin tetapi memutuskannya sehari sebelumnya, ketika melihat tumpukan seprai, sehingga pesan pengingat pada Minggu malam tampaknya akan disambut baik.

5.5 Survei dan Kuesioner

5.5.1 Kapan survei menjadi alat yang tepat

Setelah dua belas wawancara, tim Bu Rina memiliki daftar calon masalah: tidak tahu kapan cucian selesai, harga baru diketahui setelah penimbangan, balasan yang lambat pada malam hari, jam penjemputan yang tidak cocok dengan jam kerja. Wawancara tidak dapat mengatakan seberapa umum setiap masalah di antara semua pelanggan; survei dapat. Survei itu dibuat *sesudah* wawancara, dan butir-butirnya ditulis dengan kata-kata pelanggan sendiri dari wawancara. Survei yang ditulis sebelum wawancara apa pun mengajukan pertanyaan tim dengan kata-kata tim, sehingga yang diukurnya adalah asumsi tim.

5.5.2 Menulis butir yang baik

Butir survei adalah pernyataan atau pertanyaan dengan sekumpulan jawaban tetap. Bentuk yang paling umum dalam riset pengguna adalah *butir Likert*: sebuah pernyataan ("Harga jelas sebelum saya membayar") dengan lima atau tujuh jawaban dari "sangat tidak setuju" sampai "sangat setuju". Butir yang baik mengikuti beberapa aturan [12]:

- **Satu gagasan per butir.** "Harganya jelas dan wajar" mencampur dua gagasan; orang yang menganggapnya jelas tetapi tidak wajar tidak dapat menjawab.
- **Konkret dan baru terjadi.** "Saya tahu kapan cucian saya selesai" menyangkut sesuatu yang dialami pelanggan setiap kali; "Saya puas dengan layanannya" terlalu kabur untuk ditindaklanjuti.

- **Kata-kata netral.** "Bukankah menurut Anda penjemputannya praktis?" mendorong ke jawaban ya. Orang cenderung menyetujui pernyataan yang disodorkan kepada mereka, yang disebut *bias persetujuan* (*acquiescence bias*); jangan menambahnya.
- **Skala yang sama di mana-mana,** dengan arah yang sama (5 = sangat setuju) untuk setiap butir.
- **Sedikit butir.** Pelanggan yang menjawab lewat ponsel memberi dua menit, bukan dua puluh. Enam butir yang baik mengalahkan tiga puluh butir yang biasa-biasa saja.
- **Uji coba dahulu.** Sebelum mengirim, mintalah tiga orang menjawab sambil bersuara dan mengatakan apa yang mereka pahami dari setiap butir.

Survei Laundry Bersih memuat enam butir, tercantum di `items.csv`, dan dua pertanyaan latar belakang: kelompok usia dan apakah pelanggan biasanya memesan lewat WhatsApp atau datang langsung.

5.5.3 Seberapa teliti sebuah survei?

Survei adalah sampel: enam puluh pelanggan mewakili semua pelanggan Bu Rina. Setiap angka yang dihitung dari sampel, misalnya "15% setuju", adalah taksiran yang mengandung ketidakpastian. Untuk proporsi p yang ditaksir dari n jawaban, *margin galat* (*margin of error*) 95% adalah

$$E = z\sqrt{\frac{p(1-p)}{n}} \tag{5.1}$$

dengan $z = 1.96$, nilai yang menyisakan 2,5% distribusi normal di setiap ekornya. Margin ini terbesar ketika $p = 0.5$, sehingga peneliti yang merencanakan survei memakai $p = 0.5$ sebagai kasus terburuk. Untuk $n = 60$: $E = 1.96 \times \sqrt{0.25/60} = 1.96 \times 0.0645 = 0.127$, yaitu plus-minus 12,7 poin persentase. Untuk $n = 400$: $E = 1.96 \times \sqrt{0.25/400} = 1.96 \times 0.025 = 0.049$, sekitar 4,9 poin. Sampel empat kali lipat membuat margin menjadi separuhnya, karena n berada di bawah akar kuadrat.

Menyelesaikan Persamaan (5.1) untuk n memberi tahu berapa jawaban yang dibutuhkan untuk margin yang diinginkan: $n = z^2 p(1-p)/E^2$. Dengan $E = 0.10$ dan $p = 0.5$, $n = 3.8416 \times 0.25/0.01 = 96.04$, jadi 97 jawaban. Enam puluh jawaban Bu Rina cukup untuk membedakan masalah yang menimpa sebagian besar pelanggan dari masalah yang menimpa sedikit pelanggan, tetapi tidak cukup untuk membedakan 40% dari 50%.

Untuk rata-rata butir Likert, ketidakpastian dinyatakan sebagai *selang kepercayaan* (*confidence interval*). Jika butir itu mempunyai rata-rata $\bar{x}$, simpangan baku s, dan n jawaban, selang kepercayaan 95%-nya adalah

$$\bar{x} \pm t_{0.975,\,n-1} \cdot \frac{s}{\sqrt{n}} \tag{5.2}$$

dengan $t_{0.975,\,n-1}$ nilai distribusi t Student berderajat bebas $n-1$ yang menyisakan 2,5% di ekor atas. Untuk n besar nilainya mendekati 1,96; untuk $n = 60$ nilainya sekitar 2,001. Tabel cetak memberikannya, tetapi program membutuhkan rumus. Sebuah ekspansi deret klasik di sekitar nilai normal z teliti sampai sekitar tiga desimal mulai lima derajat bebas ke atas:

$$t \approx z + \frac{z^3 + z}{4\nu} + \frac{5z^5 + 16z^3 + 3z}{96\nu^2} \tag{5.3}$$

dengan $\nu = n - 1$ banyaknya derajat bebas. Untuk $\nu = 59$: $z^3 + z = 7.529 + 1.960 = 9.489$, dibagi $4 \times 59 = 236$ menghasilkan 0.0402; suku kedua $(5 \times 28.925 + 16 \times 7.529 + 3 \times 1.960)/(96 \times 3481) = 270.97/334176 = 0.0008$; jadi $t \approx 1.960 + 0.040 + 0.001 = 2.001$, sama dengan tabel.

Ringkasan ketiga yang dilaporkan peneliti adalah proporsi *top-2-box*: bagian jawaban pada dua kategori paling positif (4 dan 5). Angka ini lebih mudah dijelaskan kepada pemilik toko daripada rata-rata ("dua dari tiga pelanggan setuju bahwa memesan lewat WhatsApp itu mudah") dan merupakan proporsi, sehingga Persamaan (5.1) memberikan margin galatnya.

5.6 Membangun Perangkum Survei

5.6.1 Datanya

Jawaban survei tersimpan di `data/survey.csv`, satu baris per responden: pengenal, kelompok usia, saluran yang biasa dipakai, dan enam jawaban dari 1 sampai 5. Teks butir tersimpan di `data/items.csv`. Baris-baris pertamanya seperti ini:

```
id,age,channel,Q1,Q2,Q3,Q4,Q5,Q6
R01,30-44,walkin,2,2,5,5,4,5
R02,18-29,wa,5,2,2,4,4,4
```

5.6.2 Programnya dalam Python

Kode 5.1 membaca kedua berkas, menghitung untuk setiap butir rata-rata, selang kepercayaan Persamaan (5.2) dengan nilai t dari Persamaan (5.3), dan proporsi *top-2-box*, lalu merinci butir yang paling mengkhawatirkan menurut kelompok usia dan mencetak margin galat Persamaan (5.1) untuk seluruh sampel.

Kode 5.1 `survey_stats.py`: merangkum survei Likert dengan rata-rata, selang kepercayaan 95%, dan proporsi *top-2-box*.

```
"""Summarize a Likert survey: mean, 95% confidence interval, top-2-box.
Usage: python survey_stats.py [survey.csv] [items.csv]"""
import csv
import math
import sys


def t_critical(df):
    """Two-sided 95% t value; series expansion, accurate for df >= 5."""
    z = 1.959964
    return (z + (z**3 + z) / (4 * df)
            + (5 * z**5 + 16 * z**3 + 3 * z) / (96 * df * df))


def item_stats(values):
    """Return n, mean, sd, half-width of the 95% CI, top-2-box share."""
    n = len(values)
    mean = sum(values) / n
    var = sum((v - mean) ** 2 for v in values) / (n - 1)
    sd = math.sqrt(var)
    half = t_critical(n - 1) * sd / math.sqrt(n)
    top2 = sum(1 for v in values if v >= 4) / n
    return n, mean, sd, half, top2


def margin_of_error(n, p=0.5):
    """Worst-case 95% margin of error of a proportion, in points."""
    return 100 * 1.959964 * math.sqrt(p * (1 - p) / n)


def load(survey_path, items_path):
    with open(items_path, newline="", encoding="utf-8") as f:
        items = [(r["item"], r["text"]) for r in csv.DictReader(f)]
    with open(survey_path, newline="", encoding="utf-8") as f:
        rows = list(csv.DictReader(f))
    return items, rows
```

Kode 5.1 (Lanjutan) *survey_stats.py: merangkum survei Likert dengan rata-rata, selang kepercayaan 95%, dan proporsi top-2-box.*

```
def main(args):
    survey = args[0] if len(args) > 0 else "../data/survey.csv"
    names = args[1] if len(args) > 1 else "../data/items.csv"
    items, rows = load(survey, names)
    print(f"{len(rows)} respondents, 1 = strongly disagree ... 5 = strongly agree")
    print(f"{'item':<5}{'mean':>6}{'95% CI':>15}{'top-2':>7}  statement")
    for key, text in items:
        n, mean, sd, half, top2 = item_stats([int(r[key]) for r in rows])
        ci = f"[{mean - half:.2f}, {mean + half:.2f}]"
        print(f"{key:<5}{mean:>6.2f}{ci:>15}{top2 * 100:>6.0f}%  {text}")
    print("\nQ2 'I know when my laundry will be ready' by age group:")
    for group in ("18-29", "30-44", "45+"):
        vals = [int(r["Q2"]) for r in rows if r["age"] == group]
        n, mean, sd, half, top2 = item_stats(vals)
        print(f"  {group:<6} n={n:<3} mean {mean:.2f} +/- {half:.2f}")
    print(f"\nMargin of error of a percentage at n={len(rows)}: "
          f"+/-{margin_of_error(len(rows)):.1f} points")


if __name__ == "__main__":
    main(sys.argv[1:])
```

Tiga hal perlu dicatat. `item_stats` membagi dengan $n - 1$, bukan n, ketika menghitung variansi: inilah simpangan baku *sampel*, yang tepat ketika enam puluh jawaban mewakili populasi yang

lebih besar. `t_critical` adalah Persamaan (5.3) yang ditulis suku demi suku. Program juga membaca teks butir dari berkas alih-alih menuliskannya langsung di kode, sehingga program yang sama dapat merangkum survei berikutnya, dengan butir yang berbeda, tanpa diubah.

5.6.3 Programnya dalam Java

Kode 5.2 adalah program yang sama dalam Java. Program ini membaca berkas CSV dengan `Files.readAllLines` dan memecah setiap baris pada tanda koma, yang cukup untuk berkas seperti ini yang tidak memuat koma di dalam tanda kutip; survei yang diekspor dari alat formulir dengan jawaban teks bebas membutuhkan pengurai CSV yang sesungguhnya.

Kode 5.2 `SurveyStats.java`: ringkasan survei yang sama dalam Java.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

/** Summarize a Likert survey: mean, 95% confidence interval, top-2-box.
 *  Usage: java SurveyStats [survey.csv] [items.csv] */
public class SurveyStats {
    /** Two-sided 95% t value; series expansion, accurate for df >= 5. */
    static double tCritical(int df) {
        double z = 1.959964;
        return z + (Math.pow(z, 3) + z) / (4.0 * df)
            + (5 * Math.pow(z, 5) + 16 * Math.pow(z, 3) + 3 * z) / (96.0 * df * df);
    }

    /** Returns {n, mean, sd, half-width of the 95% CI, top-2-box share}. */
    static double[] itemStats(List<Integer> values) {
        int n = values.size();
        double sum = 0;
        for (int v : values) sum += v;
        double mean = sum / n, ss = 0;
        int top2 = 0;
        for (int v : values) {
            ss += (v - mean) * (v - mean);
            if (v >= 4) top2++;
        }
        double sd = Math.sqrt(ss / (n - 1));
        double half = tCritical(n - 1) * sd / Math.sqrt(n);
        return new double[] {n, mean, sd, half, (double) top2 / n};
    }

    /** Worst-case 95% margin of error of a proportion, in points. */
    static double marginOfError(int n) {
        return 100 * 1.959964 * Math.sqrt(0.25 / n);
    }

    static List<String[]> readCsv(String path) throws IOException {
        List<String[]> rows = new ArrayList<>();
        List<String> lines = Files.readAllLines(Path.of(path));
        for (String line : lines.subList(1, lines.size()))
            if (!line.isBlank()) rows.add(line.split(",", -1));
        return rows;
    }
```

Kode 5.2 (Lanjutan) *SurveyStats.java: ringkasan survei yang sama dalam Java.*

```
    static List<Integer> column(List<String[]> rows, int col, String age) {
        List<Integer> out = new ArrayList<>();
        for (String[] r : rows)
            if (age == null || r[1].equals(age)) out.add(Integer.parseInt(r[col]));
        return out;
    }

    public static void main(String[] args) throws IOException {
        Locale.setDefault(Locale.US);
        String survey = args.length > 0 ? args[0] : "../data/survey.csv";
        String names = args.length > 1 ? args[1] : "../data/items.csv";
        List<String[]> items = readCsv(names), rows = readCsv(survey);
        System.out.println(rows.size()
            + " respondents, 1 = strongly disagree ... 5 = strongly agree");
        System.out.printf("%-5s%6s%15s%7s  statement%n", "item", "mean", "95% CI",
            "top-2");
        for (int i = 0; i < items.size(); i++) {
            double[] s = itemStats(column(rows, 3 + i, null));
            String ci = String.format("[%.2f, %.2f]", s[1] - s[3], s[1] + s[3]);
            System.out.printf("%-5s%6.2f%15s%6.0f%%  %s%n", items.get(i)[0], s[1],
                ci, s[4] * 100, items.get(i)[1]);
        }
        System.out.println(
            "\nQ2 'I know when my laundry will be ready' by age group:");
        for (String group : new String[] {"18-29", "30-44", "45+"}) {
            double[] s = itemStats(column(rows, 4, group));
            System.out.printf("  %-6s n=%-3d mean %.2f +/- %.2f%n", group,
                (int) s[0], s[1], s[3]);
        }
        System.out.printf("%nMargin of error of a percentage at n=%d: "
            + "+/-%.1f points%n", rows.size(), marginOfError(rows.size()));
    }
}
```

`Locale.setDefault(Locale.US)` membuat `printf` menulis titik desimal, bukan koma desimal yang akan dihasilkan *locale* sistem berbahasa Indonesia, sehingga kedua program mencetak keluaran yang identik di mana pun.

5.6.4 Menjalankannya

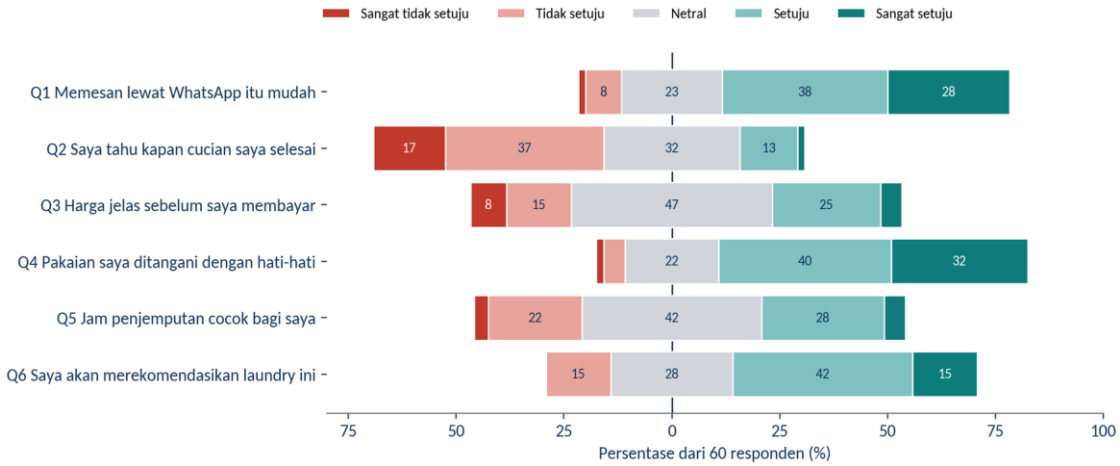
Dari folder `python`, jalankan `python3 survey_stats.py`; dari folder `java`, kompilasi dengan `javac -d out *.java` lalu jalankan `java -cp out SurveyStats`. Keduanya mencetak:

```
60 respondents, 1 = strongly disagree ... 5 = strongly agree
item   mean         95% CI  top-2  statement
Q1     3.83   [3.58, 4.09]    67%  Ordering by WhatsApp is easy
Q2     2.47   [2.21, 2.72]    15%  I know when my laundry will be ready
Q3     3.03   [2.78, 3.28]    30%  The price is clear before I pay
Q4     3.95   [3.71, 4.19]    72%  I trust my clothes are handled with care
Q5     3.10   [2.86, 3.34]    33%  The pickup times suit me
Q6     3.57   [3.33, 3.81]    57%  I would recommend the laundry to a friend

Q2 'I know when my laundry will be ready' by age group:
  18-29  n=22  mean 2.68 +/- 0.42
  30-44  n=24  mean 2.46 +/- 0.39
  45+    n=14  mean 2.14 +/- 0.63

Margin of error of a percentage at n=60: +/-12.7 points
```

Gambar 5.2 menggambar jawaban yang sama sebagai *diagram batang bertumpuk divergen*, gambaran data Likert yang paling jelas: ketidaksetujuan memanjang ke kiri garis tengah, persetujuan ke kanan, dan jawaban netral mengangkangi garis itu.



Gambar 5.2 Enam butir survei sebagai diagram batang bertumpuk divergen (60 responden); angka menunjukkan persentase responden, dan butir Q2 paling condong ke sisi tidak setuju.

5.6.5 Membaca hasilnya

Survei menegaskan apa yang diisyaratkan wawancara dan menambahkan proporsinya. Pelanggan merasa memesan lewat WhatsApp itu mudah (rata-rata 3,83, dua pertiga setuju) dan percaya pakaian mereka ditangani dengan hati-hati (3,95, 72% setuju): salurannya dan layanan intinya bukan masalah. Butir terlemah jauh adalah Q2, "Saya tahu kapan cucian saya selesai", dengan rata-rata 2,47 dan hanya 15% yang setuju; selang kepercayaannya, 2,21 sampai 2,72, seluruhnya berada di bawah titik tengah netral 3, sehingga hasil ini bukan kebetulan pengambilan sampel. Kejelasan harga (Q3) dan jam penjemputan (Q5) berada di sekitar netral dengan banyak jawaban netral, yang biasanya berarti "kadang ya, kadang tidak".

Rincian Q2 menurut usia menunjukkan kelompok tertua pada 2,14 dan termuda pada 2,68. Selang-selangnya banyak bertumpang tindih (kelompok tertua hanya empat belas orang, sehingga selangnya ±0,63), jadi survei ini saja tidak dapat membuktikan bahwa pelanggan yang lebih tua lebih menderita. Ini sebuah petunjuk, sejalan dengan dua wawancara ketika pelanggan yang lebih tua bercerita harus meminta cucunya memeriksa chat, dan sebagai petunjuklah ia harus dilaporkan.

Terakhir, margin galat: dengan enam puluh jawaban, setiap persentase di tabel tidak pasti sampai sekitar 13 poin. "15% setuju" berarti "antara beberapa persen sampai sekitar seperempat". Itu cukup teliti untuk keputusan Bu Rina, karena butir-butir lain jauh lebih tinggi, tetapi tidak cukup teliti untuk memantau perbaikan kecil dari bulan ke bulan.

> **WAWASAN KUNCI:** Wawancara memberi tahu Bu Rina *apa* masalahnya dan *mengapa* masalah itu penting; survei memberi tahu *seberapa luas* masing-masing. Salah satunya saja tidak akan cukup: wawancara bisa terlalu memberatkan satu cerita yang berkesan, dan survei yang ditulis tanpa wawancara tidak akan memuat butir yang ternyata paling penting.

5.7 Memaknai Data Kualitatif

5.7.1 Dari catatan ke kode

Dua belas wawancara menghasilkan barangkali enam puluh halaman catatan. Langkah pertama untuk memaknainya adalah *pengodean* (*coding*): membaca catatan baris demi baris dan menempelkan label pendek, *kode*, pada setiap bagian yang mengatakan sesuatu yang relevan. "Kalau sore sering nanya sudah selesai belum, dibalasnya malam" mendapat kode `ready-time-`

`unknown` dan `reply-slow-evening`. Kode diberi nama dalam bentuk pendek yang konsisten, dan daftar kode beserta definisi satu baris untuk masing-masing, yaitu *buku kode* (*codebook*), tumbuh seiring berjalannya analisis.

Analisis tematik Braun dan Clarke menggambarkan prosesnya dalam enam fase: mengakrabi data, membuat kode awal, mencari tema (kelompok kode yang saling terkait), meninjau tema terhadap data, mendefinisikan dan menamai tema, lalu menuliskannya dengan kutipan sebagai bukti [13]. Metode ini luwes, dan disiplin utamanya adalah kembali ke data pada setiap langkah: tema yang tidak didukung beberapa bagian dari beberapa partisipan adalah opini, bukan temuan.

5.7.2 Diagram afinitas

Cara klasik untuk mencari tema bersama tim adalah *diagram afinitas*. Setiap pengamatan atau kutipan ditulis pada kertas tempel, satu gagasan per kertas; tim menempelkan semua kertas di dinding dan, awalnya tanpa berbicara, memindahkannya ke dalam kelompok-kelompok yang terasa berkaitan; lalu setiap kelompok diberi nama berupa kalimat dari sudut pandang pelanggan. Untuk Laundry Bersih, dinding itu berakhir dengan lima kelompok:

- “Saya tidak pernah tahu kapan pakaian saya kembali” (`ready-time-unknown`, `reply-slow-evening`);
- “Saya baru tahu harganya ketika sudah terlambat untuk berubah pikiran” (`price-after-weighing`);
- “Penjemputan harus menyesuaikan jam kerja saya” (`pickup-window-short`, `bulk-order-weekly`);
- “Saya ingin yakin barang saya aman” (`wants-photo-proof`, `separate-items-worry`, `lost-sock-story`);
- “Memesan harus bisa dengan cara saya berbicara” (`voice-note-order`, `small-text-hard`, `child-helps-order`, `pays-by-qris`).

Rumusan nama kelompok itu penting. “Masalah komunikasi” hanyalah label; “Saya tidak pernah tahu kapan pakaian saya kembali” adalah temuan yang dapat dirancangkan solusinya.

5.7.3 Kapan wawancara sudah cukup?

Setiap studi kualitatif harus memutuskan kapan berhenti. Jawaban yang lazim adalah *saturasi*: berhenti ketika wawancara baru tidak lagi menghasilkan kode baru. Dirumuskan begitu, aturannya kabur. Guest, Namey, dan Chen mengusulkan cara untuk menjadikannya tepat dan dapat dilaporkan [6]. Pilih *ukuran dasar* b, banyaknya wawancara pertama yang kodenya menjadi dasar (mereka menyarankan 4); *panjang deret* r, banyaknya wawancara berurutan berikutnya yang ditinjau bersama (mereka menyarankan 2); dan *ambang* τ untuk informasi baru (mereka menyarankan 5%). Setelah dasar, tinjau setiap deret r wawancara. Jika B adalah banyaknya kode berbeda yang ditemukan dalam wawancara dasar dan N_{run} banyaknya kode yang pertama kali muncul dalam deret, saturasi tercapai di akhir deret pertama yang memenuhi

$$\frac{N_{\text{run}}}{B} \le \tau \tag{5.4}$$

Rasio ini membandingkan apa yang ditambahkan deret dengan apa yang ditemukan dasar. Deret yang tidak menambah apa-apa, atau hanya sedikit sekali, mengisyaratkan bahwa wawancara sejenis berikutnya juga hanya akan menambah sedikit.

Catatan berkode dari studi laundry tersimpan di `data/codes.csv`, satu baris untuk setiap kode yang disebut dalam sebuah wawancara:

```
interview,participant,code
1,P01 student,ready-time-unknown
1,P01 student,price-after-weighing
1,P01 student,voice-note-order
```

5.7.4 Program saturasi

Kode 5.3 menghitung, untuk setiap kode, dalam berapa wawancara kode itu muncul; mendaftar kode yang pertama kali ditambahkan setiap wawancara; dan menerapkan Persamaan (5.4) dengan $b = 4$, $r = 2$, dan $\tau = 0.05$.

Kode 5.3 `saturation.py`: frekuensi kode, kode baru per wawancara, dan aturan saturasi Guest, Namey, dan Chen.

```python
"""Count interview codes and test for thematic saturation
(base size 4, run length 2, new-information threshold 5%).
Usage: python saturation.py [codes.csv]"""
import csv
import sys


def load(path):
    """Return {interview number: set of codes} in interview order."""
    by_interview = {}
    with open(path, newline="", encoding="utf-8") as f:
        for r in csv.DictReader(f):
            by_interview.setdefault(int(r["interview"]), set()).add(r["code"])
    return dict(sorted(by_interview.items()))


def new_codes_per_interview(by_interview):
    seen, result = set(), []
    for number, codes in by_interview.items():
        fresh = codes - seen
        result.append((number, len(fresh), sorted(fresh)))
        seen |= codes
    return result


def saturation_point(by_interview, base=4, run=2, threshold=0.05):
    """First interview that ends a run whose new codes are <= threshold
    of the codes found in the base interviews; None if never reached."""
    numbers = list(by_interview)
    base_codes = set().union(*(by_interview[n] for n in numbers[:base]))
    seen = set(base_codes)
    fresh_counts = []
    for n in numbers[base:]:
        fresh_counts.append(len(by_interview[n] - seen))
        seen |= by_interview[n]
    for i in range(len(fresh_counts) - run + 1):
        if sum(fresh_counts[i:i + run]) / len(base_codes) <= threshold:
            return numbers[base + i + run - 1], len(base_codes)
    return None, len(base_codes)
```

Kode 5.3 (Lanjutan) *saturation.py: frekuensi kode, kode baru per wawancara, dan aturan saturasi Guest, Namey, dan Chen.*

```
def main(args):
    path = args[0] if args else "../data/codes.csv"
    by_interview = load(path)
    counts = {}
    for codes in by_interview.values():
        for c in codes:
            counts[c] = counts.get(c, 0) + 1
    total = len(by_interview)
    print(f"{total} interviews, {len(counts)} distinct codes")
    for code, k in sorted(counts.items(), key=lambda kv: (-kv[1], kv[0])):
        print(f"  {code:<22}{k:>3} of {total}  {'#' * k}")
    print("\nNew codes per interview:")
    for number, k, fresh in new_codes_per_interview(by_interview):
        print(f"  interview {number:>2}: {k}  {' '.join(fresh)}".rstrip())
    point, base = saturation_point(by_interview)
    if point is None:
        print("\nNo saturation yet: keep interviewing.")
    else:
        print(f"\nBase: {base} codes in interviews 1-4. Saturation reached "
              f"at interview {point} (run of 2 with <= 5% new codes).")


if __name__ == "__main__":
    main(sys.argv[1:])
```

Kode 5.4 adalah versi Java-nya. `TreeMap` menjaga urutan wawancara dan `TreeSet` menjaga kode setiap wawancara tetap terurut, sehingga kode-kode baru dicetak dalam urutan alfabet yang sama seperti di Python.

Kode 5.4 `Saturation.java`: analisis yang sama dalam Java.

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;
import java.util.TreeSet;

/** Count interview codes and test for thematic saturation
 *  (base size 4, run length 2, new-information threshold 5%).
 *  Usage: java Saturation [codes.csv] */
public class Saturation {
    /** Returns interview number -> set of codes, in interview order. */
    static TreeMap<Integer, Set<String>> load(String path) throws IOException {
        TreeMap<Integer, Set<String>> byInterview = new TreeMap<>();
        List<String> lines = Files.readAllLines(Path.of(path));
        for (String line : lines.subList(1, lines.size())) {
            if (line.isBlank()) continue;
            String[] f = line.split(",");
            byInterview.computeIfAbsent(Integer.parseInt(f[0]),
                k -> new TreeSet<>()).add(f[2]);
        }
        return byInterview;
    }

    /** Returns {interview that ends the first run with <= threshold new
     *  codes (or -1), number of codes in the base interviews}. */
    static int[] saturationPoint(TreeMap<Integer, Set<String>> byInterview,
                                 int base, int run, double threshold) {
        List<Integer> numbers = new ArrayList<>(byInterview.keySet());
        Set<String> seen = new TreeSet<>();
        for (int i = 0; i < base; i++)
            seen.addAll(byInterview.get(numbers.get(i)));
        int baseCount = seen.size();
        List<Integer> fresh = new ArrayList<>();
        for (int i = base; i < numbers.size(); i++) {
            Set<String> codes = byInterview.get(numbers.get(i));
            int k = 0;
            for (String c : codes) if (!seen.contains(c)) k++;
            fresh.add(k);
            seen.addAll(codes);
        }
        for (int i = 0; i + run <= fresh.size(); i++) {
```

Kode 5.4 (Lanjutan) *Saturation.java: analisis yang sama dalam Java.*

```
            int sum = 0;
            for (int j = i; j < i + run; j++) sum += fresh.get(j);
            if ((double) sum / baseCount <= threshold)
                return new int[] {numbers.get(base + i + run - 1), baseCount};
        }
        return new int[] {-1, baseCount};
    }

    public static void main(String[] args) throws IOException {
        String path = args.length > 0 ? args[0] : "../data/codes.csv";
        TreeMap<Integer, Set<String>> byInterview = load(path);
        Map<String, Integer> counts = new HashMap<>();
        for (Set<String> codes : byInterview.values())
            for (String c : codes) counts.merge(c, 1, Integer::sum);
        int total = byInterview.size();
        System.out.println(total + " interviews, " + counts.size()
            + " distinct codes");
        List<Map.Entry<String, Integer>> sorted =
            new ArrayList<>(counts.entrySet());
        sorted.sort((a, b) -> a.getValue().equals(b.getValue())
            ? a.getKey().compareTo(b.getKey()) : b.getValue() - a.getValue());
        for (Map.Entry<String, Integer> e : sorted)
            System.out.printf("  %-22s%3d of %d  %s%n", e.getKey(), e.getValue(),
                total, "#".repeat(e.getValue()));
        System.out.println("\nNew codes per interview:");
        Set<String> seen = new TreeSet<>();
        for (Map.Entry<Integer, Set<String>> e : byInterview.entrySet()) {
            List<String> fresh = new ArrayList<>();
            for (String c : e.getValue()) if (!seen.contains(c)) fresh.add(c);
            seen.addAll(e.getValue());
            String line = String.format("  interview %2d: %d  %s", e.getKey(),
                fresh.size(), String.join(" ", fresh));
            System.out.println(line.stripTrailing());
        }
        int[] sat = saturationPoint(byInterview, 4, 2, 0.05);
        if (sat[0] < 0)
            System.out.println("\nNo saturation yet: keep interviewing.");
        else System.out.printf("%nBase: %d codes in interviews 1-4. Saturation "
            + "reached at interview %d (run of 2 with <= 5%% new codes).%n",
            sat[1], sat[0]);
    }
}
```

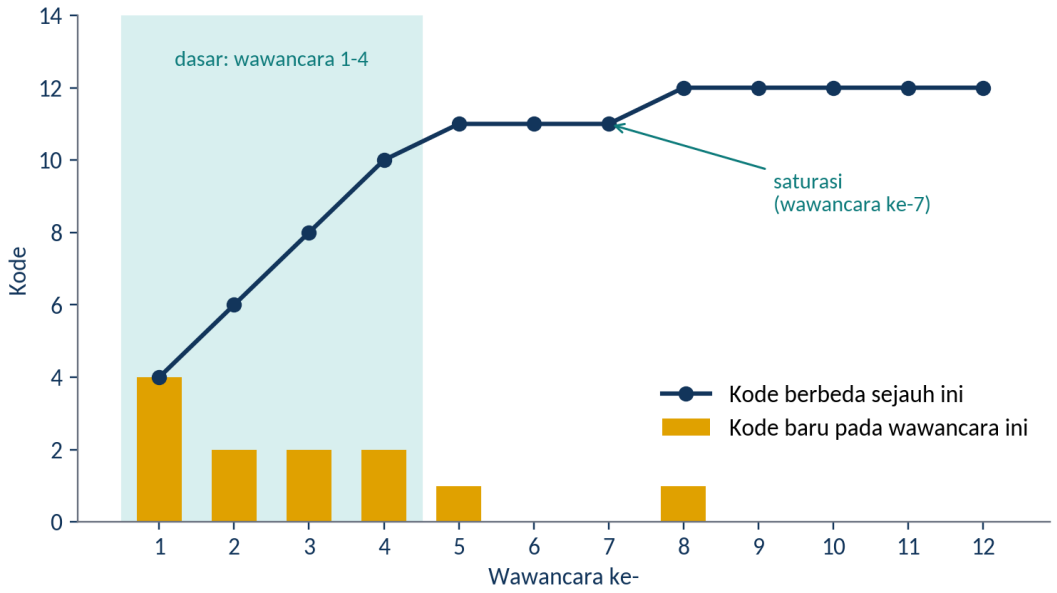
Menjalankan `python3 saturation.py` atau `java -cp out Saturation` mencetak:

```
12 interviews, 12 distinct codes
  ready-time-unknown     11 of 12  ###########
  price-after-weighing    7 of 12  #######
  reply-slow-evening      5 of 12  #####
  voice-note-order        5 of 12  #####
  pickup-window-short     4 of 12  ####
  pays-by-qris            3 of 12  ###
  wants-photo-proof       3 of 12  ###
  child-helps-order       2 of 12  ##
  small-text-hard         2 of 12  ##
  bulk-order-weekly       1 of 12  #
  lost-sock-story         1 of 12  #
  separate-items-worry    1 of 12  #

New codes per interview:
  interview  1: 4  price-after-weighing ready-time-unknown reply-slow-evening voice-note-
order
  interview  2: 2  pickup-window-short wants-photo-proof
  interview  3: 2  pays-by-qris separate-items-worry
  interview  4: 2  child-helps-order small-text-hard
  interview  5: 1  bulk-order-weekly
  interview  6: 0
  interview  7: 0
  interview  8: 1  lost-sock-story
  interview  9: 0
  interview 10: 0
  interview 11: 0
  interview 12: 0

Base: 10 codes in interviews 1-4. Saturation reached at interview 7 (run of 2 with <= 5%
new codes).
```

Wawancara dasar 1 sampai 4 menemukan sepuluh kode. Wawancara 5 menambah satu (`bulk-order-weekly`, dari pemilik kos), sehingga deret wawancara 5 dan 6 menambah $1/10 = 10\%$, di atas ambang. Deret wawancara 6 dan 7 tidak menambah apa pun, $0/10 = 0\%$, sehingga aturan ini menyatakan saturasi pada wawancara 7. **Gambar 5.3** memperlihatkan cerita yang sama: jumlah kumulatif kode naik tajam pada dasar lalu mendatar.



Gambar 5.3 Kode baru per wawancara (batang) dan kode berbeda yang ditemukan sejauh ini (garis) dalam studi laundry; aturan Persamaan (5.4) menyatakan saturasi pada wawancara 7.

Perhatikan wawancara 8. Mahasiswa yang bercerita tentang kaus kaki yang hilang menambahkan kode baru *setelah* saturasi. Itu bukan kegagalan aturan; itu pengingat tentang arti saturasi.

Saturasi berarti tema-tema yang *umum* sudah ditemukan, bukan bahwa setiap cerita yang mungkin sudah terdengar. Kaus kaki yang hilang muncul sekali dalam dua belas wawancara; waktu selesai yang tidak diketahui muncul dalam sebelas. Saturasi memberi tahu tim bahwa ia boleh berhenti merekrut partisipan sejenis dan mulai menganalisis; saturasi tidak menjanjikan bahwa masalah yang jarang tidak ada. Studi yang harus menemukan masalah langka, misalnya studi keselamatan, membutuhkan lebih banyak partisipan dan aturan berhenti yang lain.

Tabel frekuensi juga merupakan peringatan agar tidak berlebihan membaca hitungan dalam data kualitatif. Bahwa `ready-time-unknown` muncul dalam 11 dari 12 wawancara adalah bukti kuat bahwa masalah itu umum di antara *jenis orang yang diwawancarai*. Itu bukan taksiran persentase di antara semua pelanggan; untuk itulah survei ada.

5.8 Menguji Program

Program analisis layak diuji, karena rumus yang salah menghasilkan angka yang tampak sangat masuk akal. **Kode 5.5** memeriksa nilai *t* terhadap tabel cetak untuk 59 dan 9 derajat bebas, rata-rata dan proporsi *top-2-box* yang dihitung dengan tangan, margin galat untuk 400 jawaban (4,9 poin, dihitung di Subbab 5.5.3), titik saturasi data studi, dan kasus ketika setiap wawancara membawa kode baru sehingga saturasi tidak pernah tercapai.

Kode 5.5 `test_research.py`: pengujian statistik survei dan aturan saturasi.

```
"""Tests for survey_stats.py and saturation.py.
Run: python -m unittest test_research.py"""
import unittest
from survey_stats import item_stats, margin_of_error, t_critical
from saturation import load, saturation_point


class ResearchTest(unittest.TestCase):
    def test_t_critical_close_to_table(self):
        self.assertAlmostEqual(t_critical(59), 2.001, places=3)
        self.assertAlmostEqual(t_critical(9), 2.262, places=2)

    def test_item_stats(self):
        n, mean, sd, half, top2 = item_stats([5, 4, 4, 2, 1])
        self.assertEqual(n, 5)
        self.assertAlmostEqual(mean, 3.2)
        self.assertAlmostEqual(top2, 0.6)

    def test_margin_of_error(self):
        self.assertAlmostEqual(margin_of_error(400), 4.9, places=1)

    def test_saturation_on_study_data(self):
        point, base = saturation_point(load("../data/codes.csv"))
        self.assertEqual((point, base), (7, 10))

    def test_no_saturation_when_every_interview_is_new(self):
        data = {i: {f"c{i}a", f"c{i}b"} for i in range(1, 8)}
        self.assertEqual(saturation_point(data)[0], None)


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

```
Ran 5 tests in 0.001s

OK
```

Kode 5.6 melakukan pemeriksaan yang sama dalam Java tanpa pustaka pengujian, sehingga dapat berjalan hanya dengan JDK.

Kode 5.6 `ResearchTest.java`: pemeriksaan yang sama dalam Java.

```java
import java.io.IOException;
import java.util.List;
import java.util.Set;
import java.util.TreeMap;

/** Tests for SurveyStats and Saturation, without a test library.
 *  Run: java -cp out ResearchTest */
public class ResearchTest {
    static int passed = 0;

    static void check(boolean ok, String name) {
        if (!ok) throw new AssertionError("FAILED: " + name);
        passed++;
    }

    public static void main(String[] args) throws IOException {
        check(Math.abs(SurveyStats.tCritical(59) - 2.001) < 0.0005, "t(59)");
        check(Math.abs(SurveyStats.tCritical(9) - 2.262) < 0.005, "t(9)");
        double[] s = SurveyStats.itemStats(List.of(5, 4, 4, 2, 1));
        check(s[0] == 5 && Math.abs(s[1] - 3.2) < 1e-9, "mean");
        check(Math.abs(s[4] - 0.6) < 1e-9, "top-2-box");
        check(Math.abs(SurveyStats.marginOfError(400) - 4.9) < 0.05, "margin");
        int[] sat = Saturation.saturationPoint(
            Saturation.load("../data/codes.csv"), 4, 2, 0.05);
        check(sat[0] == 7 && sat[1] == 10, "saturation on study data");
        TreeMap<Integer, Set<String>> allNew = new TreeMap<>();
        for (int i = 1; i <= 7; i++)
            allNew.put(i, Set.of("c" + i + "a", "c" + i + "b"));
        check(Saturation.saturationPoint(allNew, 4, 2, 0.05)[0] == -1, "never");
        System.out.println(passed + " checks passed");
    }
}
```

```
7 checks passed
```

5.9 Dari Temuan ke Wawasan

5.9.1 Triangulasi bukti

Bu Rina kini memiliki tiga jenis bukti. Buku pesanan (perilaku, kuantitatif) menunjukkan pelanggan tetap berkurang. Wawancara dan pagi di atas sepeda motor (perilaku dan sikap, kualitatif) memperlihatkan pelanggan yang tidak tahu kapan pakaiannya kembali, yang baru tahu harga berjam-jam setelah penjemputan, dan yang menulis “sudah selesai belum?” berkali-kali. Survei (sikap, kuantitatif) menunjukkan bahwa ketidakpastian ini adalah masalah yang paling meluas. Tiga kuadran **Gambar 5.1** menunjuk ke tempat yang sama.

Bagaimana dengan penjelasan yang ditebak semua orang? Pesaing yang lebih murah muncul dalam dua wawancara, keduanya dari mahasiswa, dan tidak seorang pun menyebutnya sebagai alasan utama. Libur panjang menjelaskan sebagian penurunan di kalangan mahasiswa, tetapi tidak di kalangan pekerja dan keluarga. Kunjungan Dimas yang larut malam memang nyata, tetapi itu gejala: ia datang terlambat karena pesan “sudah selesai” dikirim terlambat, dan pelanggan tidak ada di rumah untuk menerimanya. Dan tidak seorang pun, dalam dua belas wawancara, meminta aplikasi; beberapa orang justru mengatakan mereka suka WhatsApp *karena* tidak perlu memasang aplikasi baru.

5.9.2 Menulis pernyataan wawasan

Temuan mengatakan apa yang benar; *wawasan* (*insight*) mengatakan mengapa hal itu penting bagi rancangan. Bentuk yang berguna adalah: *[siapa] membutuhkan [apa] karena [mengapa], tetapi saat ini [hambatannya]*. Lima wawasan lahir dari studi ini:

1. Pelanggan perlu tahu kapan cuciannya kembali *pada saat mereka menyerahkannya*, karena mereka merencanakan pakaian untuk seminggu berdasarkan hal itu, tetapi saat ini mereka baru tahu ketika Dimas hendak berangkat.
2. Pelanggan perlu tahu harga sebelum berkomitmen, karena mahasiswa menganggarkan sampai seribu rupiah, tetapi saat ini harga datang setelah penimbangan, berjam-jam kemudian.
3. Pelanggan yang bekerja membutuhkan jam penjemputan dan pengantaran di luar jam kerja, karena tidak ada orang di rumah pada siang hari, tetapi saat ini jamnya ditentukan oleh rute toko.
4. Pelanggan membutuhkan kepastian bahwa tidak ada yang hilang, karena pakaian bersifat pribadi dan sebagian mahal, tetapi saat ini mereka tidak menerima jumlah potong atau foto.
5. Pelanggan yang lebih tua perlu memesan dengan suara dan membaca pesan yang besar dan sederhana, karena mengetik dan huruf kecil menyulitkan mereka, tetapi saat ini daftar harganya berupa gambar kecil.

Setiap wawasan dapat diubah menjadi pertanyaan *Bagaimana kita bisa…?* (*How might we…?*) untuk langkah pencarian gagasan di Bab 4: “Bagaimana kita bisa memberi tahu waktu selesai saat penjemputan?”, “Bagaimana kita bisa memberi perkiraan harga sebelum penimbangan?”.

5.9.3 Apa yang dilakukan Bu Rina

Rekomendasinya bukan aplikasi. Rekomendasinya empat perubahan pada layanan WhatsApp, cukup murah untuk dicoba minggu berikutnya: pesan saat penjemputan berisi *perkiraan* rentang harga (dari ukuran kantong dan harga per kilogram), janji waktu selesai, dan jumlah potong pakaian; pesan “sudah selesai” yang dikirim paling lambat pukul 16.00 dengan dua pilihan jam pengantaran, salah satunya setelah pukul 18.00; foto cucian yang sudah terlipat beserta jumlah potongnya sebelum diantar; dan daftar harga berhuruf besar yang dikirim sebagai teks, bukan gambar, dengan catatan bahwa pesan suara dipersilakan. Bab 6 membuat prototipe pesan-pesan ini, dan Bab 7 mengujinya. Riset tidak memberi tahu Bu Rina apa yang harus dibangun; riset memberi tahu masalah mana yang harus diselesaikan, dan solusi menggoda mana yang akan meleset.

5.10 AI dalam Riset Pengguna, per 2026

AI generatif masuk ke riset pengguna dengan cepat, dan pada 2026 hampir setiap alat riset menawarkannya: transkripsi rekaman otomatis, ringkasan wawancara, usulan kode, pengelompokan komentar survei, bahkan “wawancara” dengan partisipan simulasi. Sebagian sungguh membantu; sebagian berbahaya dengan cara yang mudah terlewat.

Di mana ia membantu. Transkripsi menghemat berjam-jam per wawancara dan kini cukup akurat dalam bahasa Indonesia untuk analisis, asalkan ada yang memeriksa nama dan angka. Model bahasa dapat mengusulkan kumpulan kode awal untuk transkrip yang panjang, yang kemudian diterima, digabung, atau ditolak peneliti. Model itu dapat mengelompokkan ratusan jawaban teks bebas survei menjadi calon kelompok untuk ditinjau manusia. Dalam sebuah studi pengguna di CHI 2026 tentang model sumber terbuka yang dijalankan secara lokal untuk pengodean kualitatif, para peneliti menghargai alat itu terutama karena kecepatannya dan

kemampuannya mengekstrak makna di permukaan, kurang memercayainya untuk makna tersirat yang membutuhkan konteks lebih luas, dan meminta agar keluaran selalu menunjuk kembali ke bagian sumbernya serta agar buku kode dapat dikunci supaya hasilnya dapat diulang [14]. Itu aturan yang baik untuk setiap analisis berbantuan AI: setiap kode usulan harus tertaut ke bagian asalnya, dan manusialah yang memutuskan.

Di mana ia merugikan: pengguna sintetis. Beberapa produk kini menjual "pengguna sintetis" (*synthetic users*), persona buatan AI yang menjawab pertanyaan wawancara seolah-olah pelanggan sungguhan. Nielsen Norman Group mengulas alat-alat semacam ini dan menyimpulkan bahwa alat itu mungkin membantu riset meja, penyusunan panduan wawancara, atau proto-persona awal yang kelak diperiksa, tetapi tidak boleh menggantikan riset dengan orang sungguhan atau dipakai untuk memvalidasi rancangan: partisipan simulasi terlalu mengiyakan, antusias terhadap segala hal tanpa prioritas, dan tidak punya pengalaman nyata dengan produknya [15]. Tanyakan kepada pelanggan laundry sintetis apakah ia menginginkan aplikasi dengan pelacakan, dan ia akan menjawab ya dengan antusias, persis jawaban yang akan membawa Bu Rina ke arah yang salah.

Privasi. Rekaman wawancara adalah data pribadi. Mengunggahnya ke layanan AI awan berarti menyerahkannya kepada pemroses lain, yang menurut UU PDP dan PP 33/2026 harus dicakup oleh persetujuan partisipan dan oleh perjanjian dengan pemroses tersebut [7], [8]. Studi yang memakai AI hendaknya menyebutkannya dalam teks persetujuan, mengutamakan alat yang memproses data secara lokal, dan tidak pernah menempelkan nama, nomor telepon, atau wajah partisipan ke *chatbot* publik.

> **PERINGATAN:** Ringkasan wawancara buatan AI adalah tafsiran, bukan kutipan. Jangan pernah menaruh kalimat dari ringkasan AI di dalam tanda kutip dalam laporan, dan jangan pernah membiarkan ringkasan menggantikan membaca sendiri catatan dari sekurang-kurangnya beberapa wawancara.

5.11 Menerapkannya dalam Kenyataan

Bentuk studi yang sama, satu halaman rencana, beberapa wawancara dan satu pagi observasi, lalu survei singkat, cocok untuk sebagian besar layanan kecil. Tiga contoh berikut menunjukkan cara menyesuaikannya.

Antrean puskesmas. Pertanyaan risetnya "mengapa pasien datang dua jam sebelum gilirannya?". Inkuiri kontekstual berarti duduk di ruang tunggu pukul 07.00 dengan izin kepala puskesmas; wawancara dilakukan sambil orang menunggu; survei dibuat singkat dan di atas kertas, dibacakan bagi yang lebih suka begitu. Persetujuan harus sangat hati-hati: informasi kesehatan adalah data pribadi yang bersifat *spesifik* menurut UU PDP, sehingga studi tidak menanyakan apa pun tentang penyakit, hanya tentang antrean [7]. Jika partisipan menyebut keluhannya, peneliti mendengarkan dengan sopan, tidak memberi nasihat, dan menyarankan agar ia berbicara dengan perawat atau dokter yang bertugas.

Pemesanan warung lewat WhatsApp. Pertanyaannya mungkin "mengapa pelanggan tetap memesan di meja alih-alih lewat pesan?". Observasi pada jam makan siang, lima wawancara, dan riwayat chat warung itu sendiri (dengan persetujuan pemiliknya, dan hanya jumlah serta jamnya, bukan nama pelanggan) sudah cukup.

Pangkalan ojek. Pengemudi sibuk dan dibayar per perjalanan; wawancarai mereka selama waktu menunggu, batasi lima belas menit, dan bayar waktu mereka. Sikap guru dan murid magang penting di sini: pengemudi mengenal kota lebih baik daripada peneliti mana pun.

Sebelum memulai, periksa daftar berikut:

1. Keputusan dan pertanyaan riset sudah dituliskan.
2. Metode mengikuti pertanyaan, bukan sebaliknya.
3. Partisipan mencakup orang yang sudah pergi, bukan hanya yang bertahan, dan orang-orang di pinggiran (lebih tua, kurang terhubung, penyandang disabilitas).
4. Teks persetujuan memisahkan keikutsertaan dari perekaman, dan menjelaskan apa yang terjadi pada data.
5. Panduan wawancara dimulai dari kejadian spesifik terakhir dan tidak memuat pertanyaan yang menggiring atau meminta ramalan.
6. Butir survei ditulis setelah wawancara, dengan kata-kata partisipan, satu gagasan per butir.
7. Aturan berhenti untuk wawancara dan margin galat untuk survei dinyatakan.
8. Setiap temuan dalam laporan menunjuk ke buktinya: kutipan, pengamatan, angka.

5.12 IMK dalam Praktik: Profitina

Profitina adalah platform *AI Visibility* (GEO): platform ini mengukur seberapa siap situs sebuah merek untuk dikutip mesin jawaban AI dan membantu memperbaiki apa yang ditemukannya. Penggunanya pemilik merek dan pemasar, banyak di antaranya tanpa latar belakang teknis, dan metode bab ini berlaku di tiga tempat.

Membuat skor terbaca. Skor kesiapan hanya berguna jika pemilik yang nonteknis memahami artinya dan tahu langkah berikutnya. Pertanyaan yang diajukan tim di sini persis pertanyaan Subbab 5.4: bukan “apakah Anda suka dasbornya?” melainkan “ceritakan saat terakhir Anda melihat skor Anda; apa yang Anda lakukan sesudahnya?”. Jawaban jujur atas pertanyaan kedua memperlihatkan apakah penjelasan setiap sinyal, yang diberikan produk ini, benar-benar mendorong tindakan.

Persetujuan yang dipahami. Profitina menerapkan hak-hak swalayan UU PDP 27/2022: persetujuan eksplisit, ekspor data, penggantian surel dan kata sandi, serta penghapusan akun secara penuh. Apakah layar persetujuan *dipahami*, bukan sekadar diklik, adalah pertanyaan riset: wawancara singkat setelah pendaftaran (“apa yang barusan Anda setujui?”) mengujinya jauh lebih baik daripada hitungan klik, dan aturan PP 33/2026 bahwa persetujuan harus diinformasikan, spesifik, dan tidak digabungkan menjadikan uji itu bagian dari mengerjakan tugas dengan benar [8].

Alur umpan balik sebagai riset berkelanjutan. Profitina menjalankan alur yang transparan: pengguna mengirim umpan balik atas respons AI, kiriman itu dikurasi menjadi *dataset*, dan model lokal (Qwen2.5, berjalan di server Profitina sendiri) dilatih dan diukur sebelum dirilis. Dari sudut pandang riset, ini saluran yang permanen, berbasis perilaku, sekaligus kualitatif dan kuantitatif: orang memperlihatkan apa yang salah dengan kata-kata mereka sendiri, pada saat kejadian. Disiplin Subbab 5.7 berlaku tanpa perubahan: kodekan umpan baliknya, cari temanya, dan periksa setiap tema terhadap kiriman aslinya. Dan karena pemrosesan AI-nya lokal, kekhawatiran privasi di Subbab 5.10 tentang pengiriman data ke pihak ketiga tidak muncul untuk bagian analisis tersebut.

Profitina memberi saran dan penggunalah yang memutuskan; riset dengan penggunanya adalah cara tim mengetahui apakah saran itu dipahami.

5.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Memulai dari metode, bukan pertanyaan.** "Ayo survei" menghasilkan data yang tidak dipakai siapa pun. Tuliskan keputusan dan pertanyaan riset terlebih dahulu.
- **Meminta orang meramal atau merancang.** "Apakah Anda akan memakai…?" dan "fitur apa yang Anda inginkan?" menghasilkan jawaban yang sopan tetapi tidak berguna. Tanyakan kejadian spesifik terakhir.
- **Hanya berbicara dengan pelanggan yang puas.** Pelanggan setia menjelaskan mengapa orang bertahan. Rekrut orang yang sudah pergi, dan orang-orang di pinggiran.
- **Pertanyaan menggiring dan interupsi yang "membantu".** Menyarankan jawaban, atau menyelesaikan kalimat partisipan, mengganti pengalamannya dengan pengalaman Anda. Gunakan diam dan gema.
- **Mencampur kutipan, pengamatan, dan tafsiran dalam catatan.** Kelak tidak ada yang bisa membedakan apa yang dikatakan partisipan dari apa yang dipikirkan peneliti. Tandai setiap baris.
- **Menulis survei sebelum wawancara.** Survei itu lalu mengukur asumsi tim. Tulis butir dengan kata-kata partisipan.
- **Melaporkan persentase tanpa margin galatnya.** "15% setuju" dari enam puluh jawaban berarti ±13 poin; katakan begitu, dan jangan membandingkan selisih yang lebih kecil daripada margin.
- **Memperlakukan hitungan wawancara sebagai persentase populasi.** "11 dari 12 narasumber" adalah bukti bahwa masalah itu umum dalam kelompok yang dikaji, bukan taksiran untuk semua pelanggan.
- **Menyatakan saturasi tanpa aturan.** Nyatakan dasar, deret, dan ambangnya, seperti Persamaan (5.4), dan ingat bahwa saturasi tidak meniadakan masalah yang langka.
- **Memakai pengguna sintetis sebagai bukti.** Mereka dapat membantu menyiapkan studi; mereka tidak dapat menggantikannya.
- **Satu persetujuan untuk segalanya.** Ikut serta, direkam, dan pemakaian layanan AI adalah persetujuan yang terpisah; tanyakan secara terpisah.

5.14 Ringkasan Bab

- Riset pengguna mencari tahu apa yang dilakukan dan dibutuhkan orang serta mengapa, sebelum dan selama merancang. Riset dimulai dari keputusan dan pertanyaan riset, bukan dari metode.
- Metode berbeda menurut dua sumbu: apa yang dikatakan versus apa yang dilakukan orang, serta mengapa/bagaimana versus berapa banyak (**Gambar 5.1**). Memadukan metode dari kuadran yang berbeda, triangulasi, memberi keyakinan.
- Rencana riset satu halaman (**Tabel 5.2**) menetapkan keputusan, pertanyaan, metode, partisipan, etika, dan keluaran. Persetujuan menurut UU PDP 27/2022 dan PP 33/2026 harus diinformasikan, spesifik, tidak digabungkan, dan dapat ditarik.
- Wawancara semi-terstruktur dan inkuiri kontekstual mengungkap perilaku dan alasannya; tanyakan kejadian spesifik yang baru terjadi dan hindari pertanyaan yang menggiring, meminta ramalan, dan ganda (**Tabel 5.3**).

- Survei mengukur seberapa luas sesuatu. Margin galat proporsi (Persamaan (5.1)) dan selang kepercayaan rata-rata (Persamaan (5.2), dengan nilai t dari Persamaan (5.3)) menyatakan seberapa teliti taksirannya.
- Data kualitatif dikodekan, dikelompokkan menjadi tema dengan diagram afinitas, dan dianalisis sampai saturasi, yang dibuat eksplisit oleh aturan Persamaan (5.4).
- Dalam studi laundry, tidak tahu kapan cucian selesai adalah masalah yang paling umum (rata-rata Q2 2,47, persetujuan 15%, dan 11 dari 12 wawancara); tidak ada yang meminta aplikasi.
- AI membantu transkripsi dan pengodean tahap awal, selalu dengan tautan kembali ke sumber dan manusia yang memutuskan; pengguna sintetis tidak menggantikan partisipan sungguhan.

5.15 Soal Latihan

1. Untuk setiap pertanyaan berikut, katakan apakah pertanyaan itu tentang apa yang dikatakan atau dilakukan orang, dan apakah kualitatif atau kuantitatif, lalu sebutkan metode yang cocok: (a) Berapa bagian pelanggan Bu Rina yang membayar dengan QRIS? (b) Mengapa pemilik kos memesan pada hari Senin? (c) Dapatkah pelanggan berusia 65 tahun menemukan daftar harga di chat? (d) Mana dari dua rumusan pesan "sudah selesai" yang menghasilkan lebih banyak pengambilan di hari yang sama?
2. Tulis ulang setiap pertanyaan wawancara berikut agar tidak menggiring atau meminta ramalan: (a) "Anda ingin dapat foto cucian Anda, kan?" (b) "Apakah Anda mau membayar lebih untuk layanan kilat?" (c) "Apakah penjemputan kami praktis dan cepat?"
3. Dengan Persamaan (5.1), hitung dengan tangan margin galat 95% untuk proporsi di sekitar 50% dengan $n = 150$, dan banyaknya jawaban yang dibutuhkan untuk margin 5 poin.
4. Dengan Persamaan (5.3), hitung t untuk 9 derajat bebas dan bandingkan dengan nilai tabel 2,262. Lalu hitung selang kepercayaan 95% untuk sepuluh jawaban 4, 5, 3, 4, 4, 2, 5, 4, 3, 4.
5. Kembangkan `survey_stats` (dalam Python atau Java) sehingga juga mencetak, untuk setiap butir, rata-rata pelanggan WhatsApp dan pelanggan yang datang langsung. Jalankan, lalu katakan apakah ada selisih yang lebih besar daripada ketidakpastiannya.
6. Misalkan wawancara 6 dalam studi laundry juga menyebut `lost-sock-story`. Terapkan Persamaan (5.4) dengan tangan dengan $b = 4$, $r = 2$, $\tau = 5\%$: di mana saturasi tercapai sekarang? Periksa dengan menyunting salinan `codes.csv` dan menjalankan `saturation.py` padanya.
7. Tulis teks persetujuan enam bagian untuk studi buku harian yang meminta pasien puskesmas mengirim pesan suara setiap kali berkunjung. Persetujuan mana yang harus ditanyakan terpisah, dan apa yang harus terjadi jika seorang pasien menarik persetujuannya setelah dua minggu?
8. Ubah temuan "Pelanggan yang lebih tua meminta cucunya membacakan pesan harga" menjadi pernyataan wawasan dengan bentuk di Subbab 5.9.2 dan menjadi dua pertanyaan *Bagaimana kita bisa*.

5.16 Penyelesaian Soal Latihan

1. (a) Dilakukan, kuantitatif: hitung cara pembayaran di buku pesanan atau catatan pembayaran; survei tidak perlu, karena perilakunya sudah tercatat. (b) Dikatakan dan dilakukan, kualitatif: wawancara dengan pemilik kos, idealnya studi buku harian selama beberapa minggu. (c) Dilakukan, kualitatif: uji usabilitas dengan partisipan yang lebih tua (Bab 7). (d) Dilakukan,

kuantitatif: uji A/B, mengirim setiap rumusan kepada separuh pelanggan secara acak dan menghitung pengambilan di hari yang sama (Bab 7).

2. (a) "Ceritakan saat Anda menerima kembali cucian Anda kemarin. Bagaimana Anda memeriksa bahwa semuanya lengkap?" (b) "Ceritakan saat Anda perlu pakaian Anda kembali dengan cepat. Apa yang Anda lakukan?" (c) Pecah dan buka pertanyaannya: "Ceritakan bagaimana penjemputan kemarin berlangsung", lalu "Bagaimana waktunya cocok dengan hari Anda?".

3. Untuk $n = 150$: $E = 1.96 \times \sqrt{0.25/150} = 1.96 \times 0.0408 = 0.080$, sekitar ±8,0 poin. Untuk $E = 0.05$: $n = 1.96^2 \times 0.25/0.05^2 = 3.8416 \times 0.25/0.0025 = 384.16$, jadi 385 jawaban. Programnya sepakat: `margin_of_error(150)` menghasilkan 8,0 dan `margin_of_error(385)` menghasilkan 4,99.

4. Dengan $z = 1.959964$: $z^3 = 7.529$, $z^5 = 28.925$. Suku pertama $(7.529 + 1.960)/(4 \times 9) = 9.489/36 = 0.2636$. Suku kedua $(144.62 + 120.46 + 5.88)/(96 \times 81) = 270.97/7776 = 0.0348$. Jadi $t \approx 1.960 + 0.264 + 0.035 = 2.259$, selisih 0,003 dari 2,262; itulah sebabnya pengujian memberi toleransi 0,005 pada 9 derajat bebas. Untuk sepuluh jawaban itu: jumlah 38, rata-rata 3,8; kuadrat simpangan 0,04, 1,44, 0,64, 0,04, 0,04, 3,24, 1,44, 0,04, 0,64, 0,04, totalnya 7,60; $s = \sqrt{7.60/9} = 0.919$; setengah lebar $2.259 \times 0.919/\sqrt{10} = 2.259 \times 0.2906 = 0.656$. Selangnya [3.14,4.46].

5. Dalam Python, tambahkan setelah rincian menurut usia:

```
    print("\nBy channel (mean, n):")
    for key, text in items:
        parts = []
        for ch in ("wa", "walkin"):
            vals = [int(r[key]) for r in rows if r["channel"] == ch]
            n, mean, sd, half, top2 = item_stats(vals)
            parts.append(f"{ch} {mean:.2f} +/- {half:.2f} (n={n})")
        print(f"  {key}: " + "; ".join(parts))
```

Dalam Java, gelung yang sama menyaring kolom 2 sebelum memanggil `itemStats`. Studi ini memiliki 49 responden WhatsApp dan hanya 11 responden yang datang langsung, sehingga setengah lebar selang kelompok datang langsung lebar, dari ±0,56 sampai ±0,78. Selisih terbesar ada pada Q2: 2,37 ± 0,29 untuk pelanggan WhatsApp dibandingkan 2,91 ± 0,56 untuk pelanggan yang datang langsung. Kedua selang, [2,08; 2,66] dan [2,35; 3,47], bertumpang tindih, sehingga survei ini tidak memberi bukti kuat adanya selisih. Arah selisihnya masuk akal (pelanggan yang datang langsung melihat tokonya dan dapat bertanya langsung), sehingga layak menjadi pertanyaan pada putaran wawancara berikutnya. Kesimpulan yang jujur adalah "belum ada bukti selisih", bukan "tidak ada selisih".

6. Dasarnya tidak berubah: sepuluh kode dalam wawancara 1–4. Wawancara 5 menambah `bulk-order-weekly`; wawancara 6 kini menambah `lost-sock-story`. Deret (5, 6): $2/10 = 20\%$. Deret (6, 7): wawancara 6 menambah 1, wawancara 7 menambah 0, jadi $1/10 = 10\%$. Deret (7, 8): wawancara 8 tidak lagi menambah apa pun (kaus kaki yang hilang sudah muncul), jadi $0/10 = 0\% \leq 5\%$ dan saturasi tercapai pada wawancara 8. `saturation.py` pada berkas yang telah disunting mencetak "Saturation reached at interview 8".

7. Teks yang sesuai: (1) siapa dan untuk apa: "puskesmas dan tim mahasiswa ingin memperpendek waktu tunggu"; (2) apa yang terjadi: "selama empat minggu, kirim pesan suara singkat ke nomor studi setiap kali Anda datang, sebutkan jam kedatangan dan jam Anda dipanggil"; (3) apa yang disimpan dan siapa yang mendengarnya: "hanya tim studi; catatan memakai kode; pesan suara dihapus setelah dituliskan; kami tidak akan bertanya tentang

penyakit Anda"; (4) sukarela, dapat berhenti kapan saja, tanpa pengaruh terhadap perawatan Anda; (5) ucapan terima kasih: pulsa setiap minggu; (6) pertanyaan terpisah: "Apakah Anda setuju ikut serta?", "Apakah Anda setuju pesan suara Anda disimpan sampai dituliskan?" dan, jika dipakai layanan transkripsi AI, "Apakah Anda setuju pesan suara Anda ditranskripsikan oleh layanan tersebut?". Jika seorang pasien menarik persetujuannya, pemrosesan datanya harus dihentikan dalam 72 jam setelah permintaan yang terverifikasi dan pasien diberi tahu bahwa pemrosesan telah berhenti, sebagaimana disyaratkan PP 33/2026; pesan suaranya yang sudah ada dihapus, dan perawatannya tidak terpengaruh.

8. Wawasan: "Pelanggan yang lebih tua perlu membaca harga dan waktu selesai tanpa bantuan, karena mereka ingin mengurus cuciannya sendiri, tetapi saat ini pesannya berupa gambar kecil dan teks panjang yang mereka minta dibacakan cucunya." Pertanyaan: "Bagaimana kita bisa membuat pesan harga terbaca sekilas di ponsel apa pun?" dan "Bagaimana kita bisa memungkinkan pelanggan yang lebih tua mengonfirmasi pesanan dengan satu pesan suara?".

Daftar Pustaka

[1] ANTARA News, "Survei APJII: Penetrasi internet di Indonesia 2026 capai 81,7 persen," May 21, 2026, reporting Asosiasi Penyelenggara Jasa Internet Indonesia, *Survei Penetrasi Internet Indonesia 2026*. [Online]. Available: https://www.antaranews.com/berita/5576225/survei-apjii-penetrasi-internet-di-indonesia-2026-capai-817-persen

[2] StatCounter Global Stats, "Desktop vs Mobile vs Tablet Market Share Indonesia," data for August 2026, accessed Sep. 2026. [Online]. Available: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/indonesia

[3] International Organization for Standardization, *ISO 9241-210:2019 Ergonomics of Human-System Interaction — Part 210: Human-Centred Design for Interactive Systems*. Geneva, Switzerland: ISO, 2019.

[4] C. Rohrer, "When to Use Which User-Experience Research Methods," Nielsen Norman Group, updated 2022. [Online]. Available: https://www.nngroup.com/articles/which-ux-research-methods/

[5] Republik Indonesia, *Undang-Undang Nomor 8 Tahun 2016 tentang Penyandang Disabilitas*. Jakarta, Indonesia, 2016.

[6] G. Guest, E. Namey, and M. Chen, "A simple method to assess and report thematic saturation in qualitative research," *PLOS ONE*, vol. 15, no. 5, e0232076, 2020, doi: 10.1371/journal.pone.0232076.

[7] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. Jakarta, Indonesia, 2022.

[8] Bisnis.com, "Aturan Turunan UU PDP: Persetujuan Data Konsumen Lebih Ketat," Aug. 29, 2026, on Peraturan Pemerintah Nomor 33 Tahun 2026. [Online]. Available: https://teknologi.bisnis.com/read/20260829/84/1999753/aturan-turunan-uu-pdp-persetujuan-data-konsumen-lebih-ketat

[9] Hukumonline Pro, "Pemerintah Akhirnya Terbitkan Kerangka Implementasi Pelindungan Data Pribadi: Berlaku Mulai 16 Januari 2027," 2026. [Online]. Available: https://pro.hukumonline.com/a/lt6a97c3fe4527c/pemerintah-akhirnya-terbitkan-kerangka-implementasi-pelindungan-data-pribadi–berlaku-mulai-16-januari-2027

[10] S. Portigal, *Interviewing Users: How to Uncover Compelling Insights*, 2nd ed. Brooklyn, NY, USA: Rosenfeld Media, 2023.

[11] K. Holtzblatt and H. Beyer, *Contextual Design: Design for Life*, 2nd ed. Cambridge, MA, USA: Morgan Kaufmann, 2017.

[12] J. Sauro and J. R. Lewis, *Quantifying the User Experience: Practical Statistics for User Research*, 2nd ed. Cambridge, MA, USA: Morgan Kaufmann, 2016.

[13] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[14] T. T. Ngo, D. Nguyen Van, A.-M. Nguyen, P.-A. Do, and A. Nguyen-Quoc, "Qualitative Coding Analysis through Open-Source Large Language Models: A User Study and Design Recommendations," in *Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems (CHI EA '26)*, Barcelona, Spain, 2026, doi: 10.1145/3772363.3798320.

[15] M. Rosala and K. Moran, "Synthetic Users: If, When, and How to Use AI-Generated 'Research'," Nielsen Norman Group, Jun. 21, 2024. [Online]. Available: https://www.nngroup.com/articles/synthetic-users/

Bab 6

Prototipe: dari Kertas hingga Interaktif (dan Berbantuan AI)

Prototipe adalah pertanyaan yang dapat Anda pegang. Sebelum puskesmas membelanjakan anggarannya untuk kios layar sentuh, sebelum sebuah laundry mengubah pesan yang dikirimnya seribu kali sebulan, seseorang sebaiknya menaruh versi kasarnya di depan orang sungguhan dan mengamati apa yang terjadi. Bab ini mengajarkan cara melakukannya pada setiap tingkat kehalusan. Di akhir bab Anda akan mampu menyatakan pertanyaan yang harus dijawab sebuah prototipe dan memilih fidelitasnya sesuai pertanyaan itu; menjalankan sesi prototipe kertas dengan seorang manusia yang berperan sebagai "komputer"; menuliskan alur layar dalam berkas teks kecil dan memeriksanya dengan program Java dan Python yang menemukan layar tak terjangkau, jalan buntu, dan jumlah ketukan paling sedikit menuju tujuan; membangun prototipe berkode yang dapat diklik untuk kios antrean, dilayani oleh server HTTP bawaan JDK dan oleh Flask, dengan tombol besar yang aksesibel dan log setiap tampilan layar; mengambil tangkapan layarnya secara otomatis dengan Playwright; serta memakai alat prototipe berbantuan AI tahun 2026, seperti Figma Make dan Google Stitch, tanpa membiarkan kecepatannya menipu Anda sehingga melewatkan pertanyaan-pertanyaan yang penting.

6.1 Antrean Sebelum Fajar

Setiap hari kerja pukul enam lewat seperempat, sebelum gerbang sebuah puskesmas di pinggir kota di Jawa Timur dibuka, antrean sudah terbentuk di luar. Pasien lanjut usia datang paling awal, karena mereka tahu nomor untuk poli umum habis menjelang siang. Ibu-ibu yang membawa bayi ke poli KIA menyusul, lalu para pekerja yang berharap dilayani sebelum jam kerjanya. Pukul setengah delapan gerbang dibuka, petugas di meja depan menuliskan nomor di sobekan kertas, dan penantian yang sesungguhnya dimulai.

Kepala puskesmas, dr. Wulan, punya rencana dan pos anggaran: kios layar sentuh di pintu masuk yang mencetak nomor antrean, seperti di bank dan rumah sakit besar. Seorang vendor sudah mengirim brosur dengan contoh layar: masuk dengan NIK pasien, pertanyaan tentang BPJS Kesehatan, daftar sebelas layanan berhuruf kecil, dan halaman konfirmasi. Kelihatannya profesional. Vendor membutuhkan dua bulan untuk memasangnya, dan puskesmas harus mengeluarkan sebagian besar anggaran peralatannya untuk membelinya.

Sebuah tim mahasiswa yang sedang menjalankan program pengabdian masyarakat di puskesmas itu meminta waktu satu minggu sebelum pesanan ditandatangani. Mereka tidak memegang layar vendor; mereka memegang kertas, gunting, spidol, dan kardus yang dipotong seukuran kios. Dalam seminggu itu mereka mempelajari tiga hal yang tidak mungkin diungkapkan brosur. Pasien lanjut usia membaca "BPJS atau Umum?" sebagai pertanyaan tentang poli mana yang dituju, bukan tentang cara membayar. Layar bantuan dalam alur vendor tidak punya jalan kembali. Dan jam buka poli, yang ingin diketahui separuh pasien yang datang pagi, berada di layar yang tidak dapat dicapai siapa pun. Tidak satu pun dari masalah ini sulit setelah terlihat. Semuanya akan memakan waktu berminggu-minggu keluhan dan permintaan perubahan seandainya kios sudah terpasang.

Itulah janji prototipe: menemukan masalah selagi masih murah untuk diperbaiki. Bab ini mengikuti tim mahasiswa itu dari versi kertas sampai prototipe berkode yang dapat diklik, dan memperlihatkan bagaimana cara berpikir yang sama berlaku untuk pesan-pesan laundry di Bab 5 dan untuk alat AI yang, pada 2026, dapat menghasilkan layar yang rapi dari sebuah kalimat.

6.2 Gagasannya dalam Bahasa Sederhana

6.2.1 Prototipe adalah pertanyaan

> **DEFINISI.** *Prototipe* adalah representasi apa pun dari sebuah rancangan, dibuat sebelum produk jadi, yang memungkinkan orang merasakan atau menilai suatu aspeknya. *Fidelitas*-nya adalah seberapa dekat ia menyerupai produk jadi dalam tampilan, interaksi, dan isi. Prototipe disebut *berfidelitas rendah* bila kasar dan cepat dibuat (sketsa, layar kertas) dan *berfidelitas tinggi* bila tampak dan berperilaku seperti produk sungguhan (*mockup* yang dapat diklik, prototipe berkode).

Houde dan Hill mengusulkan cara paling berguna untuk memikirkan prototipe: tanyakan *apa yang diprototipekan oleh prototipe itu* [1]. Setiap prototipe menjelajahi satu atau lebih dari tiga pertanyaan. Pertanyaan **peran** (*role*) menanyakan apa yang dilakukan artefak itu dalam kehidupan seseorang: apakah kios yang mencetak nomor benar-benar memperpendek penantian sebelum fajar, atau masalahnya ada di tempat lain? Pertanyaan **tampilan dan rasa** (*look and feel*) menanyakan bagaimana rasanya dilihat dan disentuh: apakah tombolnya cukup besar untuk jari yang gemetar, apakah teksnya terbaca dari jarak selengan? Pertanyaan **implementasi** menanyakan apakah ia dapat dibangun: dapatkah printer termal kecil dikendalikan dari peramban, apakah kios tetap bekerja ketika koneksi internet putus? Kardus dengan layar kertas menjawab pertanyaan peran dan banyak bagian tampilan dan rasa; ia tidak berkata apa pun tentang implementasi. Printer sungguhan yang tersambung ke laptop menjawab implementasi dan bukan yang lain. Tim yang baik tahu untuk pertanyaan mana setiap prototipenya dibuat.

6.2.2 Sketsa dan prototipe; hal yang tepat dan hal yang benar

Buxton menarik garis tegas antara *membuat sketsa* dan *membuat prototipe* [2]. Sketsa itu cepat, murah, banyak, dan sekali pakai; ia menyarankan dan menjelajah, dan tujuannya bertanya *apakah kita merancang hal yang tepat?* Prototipe lebih halus, lebih sedikit, dan lebih mahal; ia menguji dan menegaskan, dan tujuannya bertanya *apakah kita merancang hal itu dengan benar?* Tim yang langsung melompat ke prototipe yang rapi telah menjawab pertanyaan kedua sebelum pertanyaan pertama, dan biasanya untuk rancangan yang keliru.

Gambar 6.1 menyusun bentuk-bentuk prototipe yang umum pada sumbu fidelitas, dari sketsa yang dibuat dalam hitungan menit sampai prototipe berkode yang memakan waktu berhari-hari. Biaya membuat prototipe, dan biaya mengubahnya, naik dari kiri ke kanan. Begitu pula risiko sebuah jebakan halus: orang yang melihat layar rapi mengomentari warna dan huruf, sedangkan orang yang melihat sketsa pensil mengomentari apakah gagasannya masuk akal.



Gambar 6.1 Spektrum fidelitas: dari sketsa yang bertanya apakah ini hal yang tepat sampai prototipe berkode yang bertanya apakah hal itu sudah benar; alat berbantuan AI mencapai fidelitas tinggi dengan cepat, sehingga pertanyaan di sisi kiri mudah terlewat.

> **WAWASAN KUNCI:** Sesuaikan fidelitas dengan pertanyaannya, bukan dengan alat yang Anda punya. Pertanyaan awal (apakah ini layanan yang tepat? dalam urutan apa?) layak dijawab dengan prototipe kasar, karena prototipe kasar mengundang kritik yang jujur dan tidak mahal untuk dibuang. Pertanyaan akhir (apakah tombol ini cukup besar? apakah kata-katanya jelas?) layak dijawab dengan prototipe yang halus.

6.2.3 Prototipe horizontal dan vertikal

Pembedaan kedua menyangkut cakupan. Prototipe *horizontal* memperlihatkan banyak fitur, masing-masing hanya sebagai permukaan: setiap layar kios ada, tetapi menekan "Cetak" hanya menampilkan gambar tiket, bukan mencetaknya. Prototipe *vertikal* mengimplementasikan satu jalur secara mendalam: jalur "ambil nomor untuk poli gigi" benar-benar menerbitkan nomor, menghitung orang yang mengantre di depan, dan memperkirakan lama tunggu, sementara jalur lain belum ada. Kebanyakan prototipe merupakan paduan, sering disebut *bentuk T*: lebar dan dangkal di seluruh produk, dalam di sepanjang satu jalur yang paling penting. Prototipe kios berkode di Subbab 6.6 persis seperti itu.

6.3 Prototipe Kertas

6.3.1 Bahan dan peran

Prototipe kertas adalah cara tercepat untuk menguji alur antarmuka dengan orang sungguhan. Panduan praktis Snyder menggambarkannya sebagai teknik ketika "komputer" adalah seorang manusia yang memindahkan layar-layar kertas sebagai tanggapan atas sentuhan partisipan, sementara fasilitator memberikan tugas dan pengamat mencatat [3]. Bahannya murah: kartu indeks atau kertas A5 untuk layar, kertas tempel untuk jendela sembul dan nilai yang berubah, spidol, gunting, selotip yang mudah dilepas, dan sesuatu yang berperan sebagai bingkai perangkat. Tim mahasiswa memotong bagian depan kardus seukuran layar kios 21 inci dan menyelipkan layar-layar kertas dari belakang.

Ada tiga peran dalam sebuah sesi:

- **Fasilitator** menyambut partisipan, menjelaskan bahwa yang diuji adalah rancangannya, bukan orangnya, membacakan tugas, dan meminta partisipan berpikir sambil bersuara (*think-aloud*; Bab 7 membahasnya secara mendalam).

- **Komputer** duduk di samping atau di belakang prototipe, tidak pernah berbicara, dan menukar layar persis seperti yang ditentukan rancangan, termasuk untuk sentuhan yang "salah". Jika partisipan menyentuh sesuatu yang belum punya layar, komputer menunjukkan kartu kosong bertanda tanya; saat itu sendiri adalah sebuah temuan.
- **Pengamat** mencatat apa yang dilakukan, dikatakan, dan dibimbangkan partisipan, satu pengamatan per baris, seperti di Subbab 5.4.4.

6.3.2 Sebuah sesi di puskesmas

Tim menyiapkan alur vendor sebagai delapan layar kertas dan tiga tugas: "Anda ingin berobat ke dokter umum hari ini; ambil nomor", "Anda ingin tahu sampai jam berapa poli gigi buka hari ini", dan "Anda tidak sengaja menekan Bantuan; kembalilah ke tempat semula". Dengan izin dr. Wulan dan teks persetujuan seperti yang dijelaskan Subbab 5.3.3, mereka menguji delapan orang selama dua pagi: lima pasien lanjut usia, dua ibu, dan seorang pekerja.

Versi kertas mengungkap ketiga masalah di Subbab 6.1 dalam empat sesi pertama. Versi itu juga menghasilkan temuan yang lebih kecil: tiga dari lima partisipan lanjut usia menyentuh teks judul karena mengira itu tombol; dua partisipan mencoba menggulir daftar yang tidak bisa digulir; dan beberapa orang, sebelum hal lain, bertanya berapa lama harus menunggu, yang tidak disebutkan pada tiket vendor. Tidak satu pun dari temuan ini membutuhkan satu baris kode pun untuk ditemukan.

6.3.3 Wizard of Oz

Kadang yang perlu diprototipekan bukan sekumpulan layar, melainkan perilaku yang sulit dibangun, seperti memahami ucapan. Teknik *Wizard of Oz* memalsukannya: seorang manusia yang tersembunyi, sang penyihir, menghasilkan tanggapan sistem secara manual, sementara partisipan mengira, atau setuju untuk berpura-pura, bahwa sistemlah yang menghasilkannya. Kelley memakainya pada awal 1980-an untuk merancang aplikasi perkantoran berbahasa alami sebelum kemampuan memahami bahasanya ada, dengan mengulang-ulang rancangan bersama manusia di dalam lingkarannya sampai kosakata dan dialognya tepat [4].

Tim mahasiswa memakainya untuk satu gagasan: pilihan suara bagi pasien yang tidak dapat membaca layar. Seorang partisipan berkata "saya mau ke poli gigi" kepada kios; seorang anggota tim yang bersembunyi di balik kardus mendengarkan lalu menyelipkan layar konfirmasi. Sesi itu menunjukkan bahwa pilihan suara dipakai dan disambut baik oleh dua partisipan lanjut usia, dan bahwa konfirmasi harus mengulang nama layanan dengan suara, karena partisipan yang tidak dapat membaca layar juga tidak dapat membaca konfirmasinya. Wizard of Oz juga merupakan cara alami untuk membuat prototipe *AI assistant* sebelum membangunnya, penggunaan yang dibahas kembali di Bab 13.

6.4 Memeriksa Alur Sebelum Membangunnya

6.4.1 Alur sebagai berkas teks

Prototipe kertas adalah sebuah graf: layar adalah simpul, tombol adalah panah dari satu layar ke layar lain. Sebagian kesalahan dalam alur dapat ditemukan tanpa partisipan sama sekali, cukup dengan melihat grafnya: layar tanpa jalan masuk bersifat *tak terjangkau*, layar tanpa jalan keluar adalah *jalan buntu* (kecuali bila ia tujuan tempat tugas berakhir), dan jumlah ketukan menuju tujuan adalah panjang lintasan terpendek ke sana. Tidak ada yang menggambar graf dengan

sempurna di atas kertas, sehingga alur itu layak dituliskan dalam bentuk yang dapat diperiksa program.

Formatnya satu baris per tombol, `screen -> next : label`, ditambah satu baris `start` dan satu atau lebih baris `goal`; `#` memulai komentar. **Kode 6.1**, berkas `data/flow_v1.txt`, adalah alur vendor sebagaimana diuji tim di atas kertas:

Kode 6.1 `flow_v1.txt`: alur kios pada uji kertas, satu baris per tombol.

```
# Paper prototype of the puskesmas queue kiosk, version 1
# screen -> next screen : label on the button
start welcome
goal ticket
welcome -> service : Ambil antrean / Take a number
welcome -> help : Bantuan / Help
service -> bpjs : Umum / General
service -> bpjs : Gigi / Dental
bpjs -> nik : Pakai BPJS / Use BPJS
bpjs -> confirm : Bayar sendiri / Pay myself
nik -> confirm : Lanjut / Next
confirm -> ticket : Cetak / Print
schedule -> service : Kembali / Back
```

6.4.2 Pemeriksanya

Pemeriksa ini membaca berkas ke dalam peta dari setiap layar ke tombol-tombol keluarnya, lalu menjalankan *penelusuran melebar* (*breadth-first search*) dari layar awal. Penelusuran melebar mengunjungi layar menurut jaraknya: pertama layar awal, lalu setiap layar yang berjarak satu ketukan, lalu setiap layar yang berjarak dua ketukan, dan seterusnya, dengan memakai antrean. Saat pertama kali mencapai sebuah layar, ia telah menemukan lintasan terpendek ke layar itu, karena setiap lintasan yang lebih pendek pasti sudah dijelajahi lebih dahulu. Setelah penelusuran, layar yang tidak pernah dicapai adalah layar tak terjangkau; layar tanpa tombol keluar yang bukan tujuan adalah jalan buntu. **Kode 6.2** adalah versi Python-nya.

Kode 6.2 `flow_check.py`: menemukan layar tak terjangkau, jalan buntu, dan ketukan paling sedikit menuju setiap tujuan.

```
"""Check a screen-flow file of a prototype: unreachable screens, dead ends,
and the fewest taps from the start to each goal.
Usage: python flow_check.py [flow file]"""
import sys
from collections import deque


def load(path):
    """Return (start, goals, links) where links maps screen -> [(next, label)]."""
    start, goals, links = None, set(), {}
    with open(path, encoding="utf-8") as f:
        for raw in f:
            line = raw.split("#", 1)[0].strip()
            if not line:
                continue
            if line.startswith("start "):
                start = line.split()[1]
            elif line.startswith("goal "):
                goals.add(line.split()[1])
            else:
                left, label = line.split(":", 1)
                src, dst = [s.strip() for s in left.split("->")]
                links.setdefault(src, []).append((dst, label.strip()))
                links.setdefault(dst, [])
    return start, goals, links


def taps_from(start, links):
    """Breadth-first search: fewest taps and the path to every screen."""
    best = {start: [start]}
    queue = deque([start])
    while queue:
        screen = queue.popleft()
        for nxt, _ in links.get(screen, []):
            if nxt not in best:
                best[nxt] = best[screen] + [nxt]
                queue.append(nxt)
    return best
```

Kode 6.2 (Lanjutan) *flow_check.py: menemukan layar tak terjangkau, jalan buntu, dan ketukan paling sedikit menuju setiap tujuan.*

```
def check(start, goals, links):
    paths = taps_from(start, links)
    screens = sorted(links)
    unreachable = [s for s in screens if s not in paths]
    dead_ends = [s for s in screens if not links[s] and s not in goals]
    return paths, unreachable, dead_ends


def main(args):
    path = args[0] if args else "../data/flow_v1.txt"
    start, goals, links = load(path)
    paths, unreachable, dead_ends = check(start, goals, links)
    print(f"{path}: {len(links)} screens, "
          f"{sum(len(v) for v in links.values())} buttons")
    for goal in sorted(goals):
        if goal in paths:
            steps = paths[goal]
            print(f"  goal '{goal}': {len(steps) - 1} taps  "
                  + " > ".join(steps))
        else:
            print(f"  goal '{goal}': NOT REACHABLE from '{start}'")
    print("  unreachable screens: " + (", ".join(unreachable) or "none"))
    print("  dead ends (no way on or back): " + (", ".join(dead_ends) or "none"))


if __name__ == "__main__":
    main(sys.argv[1:])
```

Kode 6.3 adalah versi Java-nya. Program ini menyimpan layar dalam `TreeMap` agar laporannya keluar dalam urutan alfabet, seperti yang dilakukan panggilan `sorted` di Python, dan memakai `ArrayDeque` sebagai antrean penelusuran.

Kode 6.3 `FlowCheck.java`: pemeriksa yang sama dalam Java.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;
import java.util.TreeSet;

/** Check a screen-flow file of a prototype: unreachable screens, dead ends,
 *  and the fewest taps from the start to each goal.
 *  Usage: java FlowCheck [flow file] */
public class FlowCheck {
    String start;
    TreeSet<String> goals = new TreeSet<>();
    TreeMap<String, List<String>> links = new TreeMap<>();
    int buttons = 0;

    static FlowCheck load(String path) throws IOException {
        FlowCheck f = new FlowCheck();
        for (String raw : Files.readAllLines(Path.of(path))) {
            String line = raw.split("#", 2)[0].strip();
            if (line.isEmpty()) continue;
            if (line.startsWith("start ")) f.start = line.split("\\s+")[1];
            else if (line.startsWith("goal ")) f.goals.add(line.split("\\s+")[1]);
            else {
                String left = line.split(":", 2)[0];
                String[] ends = left.split("->");
                String src = ends[0].strip(), dst = ends[1].strip();
                f.links.computeIfAbsent(src, k -> new ArrayList<>()).add(dst);
                f.links.computeIfAbsent(dst, k -> new ArrayList<>());
                f.buttons++;
            }
        }
        return f;
    }
```

Kode 6.3 (Lanjutan) *FlowCheck.java: pemeriksa yang sama dalam Java.*

```
    /** Breadth-first search: fewest taps and the path to every screen. */
    Map<String, List<String>> tapsFrom() {
        Map<String, List<String>> best = new LinkedHashMap<>();
        best.put(start, List.of(start));
        ArrayDeque<String> queue = new ArrayDeque<>(List.of(start));
        while (!queue.isEmpty()) {
            String screen = queue.poll();
            for (String next : links.getOrDefault(screen, List.of()))
                if (!best.containsKey(next)) {
                    List<String> p = new ArrayList<>(best.get(screen));
                    p.add(next);
                    best.put(next, p);
                    queue.add(next);
                }
        }
        return best;
    }

    List<String> unreachable(Map<String, List<String>> paths) {
        List<String> out = new ArrayList<>();
        for (String s : links.keySet()) if (!paths.containsKey(s)) out.add(s);
        return out;
    }

    List<String> deadEnds() {
        List<String> out = new ArrayList<>();
        for (Map.Entry<String, List<String>> e : links.entrySet())
            if (e.getValue().isEmpty() && !goals.contains(e.getKey()))
                out.add(e.getKey());
        return out;
    }

    public static void main(String[] args) throws IOException {
        String path = args.length > 0 ? args[0] : "../data/flow_v1.txt";
        FlowCheck f = load(path);
        Map<String, List<String>> paths = f.tapsFrom();
        System.out.println(path + ": " + f.links.size() + " screens, "
            + f.buttons + " buttons");
        for (String goal : f.goals) {
            if (paths.containsKey(goal)) {
                List<String> steps = paths.get(goal);
                System.out.println("  goal '" + goal + "': " + (steps.size() - 1)
                    + " taps  " + String.join(" > ", steps));
            } else {
                System.out.println("  goal '" + goal + "': NOT REACHABLE from '"
                    + f.start + "'");
```

Kode 6.3 (Lanjutan) *FlowCheck.java: pemeriksa yang sama dalam Java.*

```
            }
        }
        List<String> un = f.unreachable(paths), dead = f.deadEnds();
        System.out.println("  unreachable screens: "
            + (un.isEmpty() ? "none" : String.join(", ", un)));
        System.out.println("  dead ends (no way on or back): "
            + (dead.isEmpty() ? "none" : String.join(", ", dead)));
    }
}
```

6.4.3 Menjalankannya pada kedua versi

Dijalankan pada versi uji kertas, `python3 flow_check.py ../data/flow_v1.txt` (atau `java -cp out FlowCheck ../data/flow_v1.txt`) mencetak:

```
../data/flow_v1.txt: 8 screens, 9 buttons
  goal 'ticket': 4 taps  welcome > service > bpjs > confirm > ticket
  unreachable screens: schedule
  dead ends (no way on or back): help
```

Program ini menemukan dua dari tiga masalah yang ditemukan partisipan: jam poli tak terjangkau dan layar bantuan adalah jalan buntu. Program ini tidak dapat menemukan masalah ketiga, bahwa "BPJS atau Umum?" disalahpahami, karena itu masalah makna, bukan struktur. Program memeriksa struktur; manusia memeriksa makna. Keduanya dibutuhkan.

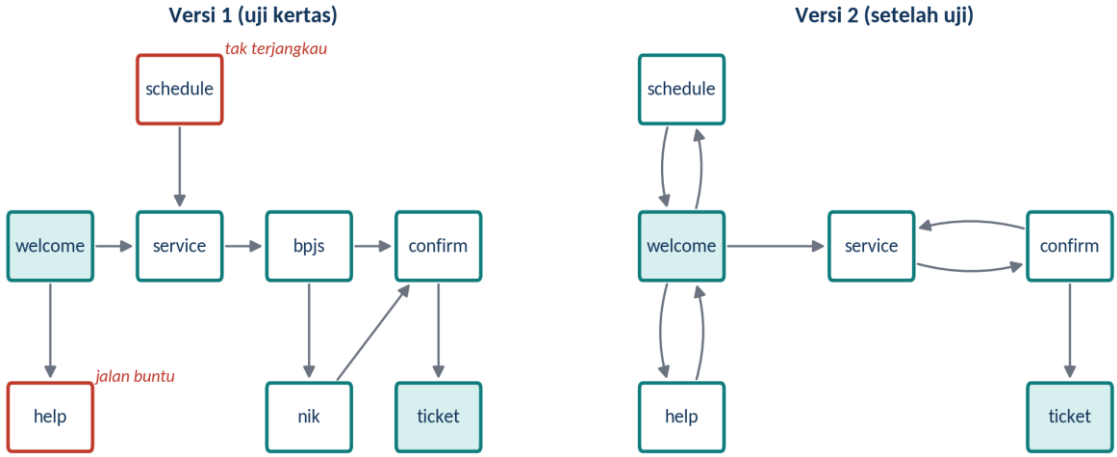
Setelah sesi-sesi kertas, tim membuat tiga perubahan. Pertanyaan BPJS dipindahkan dari kios ke meja depan, tempat seorang petugas dapat menjelaskannya, sekaligus menghilangkan layar NIK. Bantuan dan jam poli sama-sama mendapat tombol "Kembali" dan tempat di layar sambutan. Dan dua layanan yang tidak ada di daftar vendor tetapi justru dibutuhkan pasien pagi, KIA dan poli lansia, ditambahkan. Alur yang baru, `data/flow_v2.txt`, adalah **Kode 6.4**:

Kode 6.4 `flow_v2.txt`: alur setelah uji kertas.

```
# Version 2 after the paper test: help returns, schedule reachable,
# the BPJS question moved to the counter
start welcome
goal ticket
welcome -> service : Ambil antrean / Take a number
welcome -> help : Bantuan / Help
welcome -> schedule : Jadwal poli / Clinic hours
help -> welcome : Kembali / Back
schedule -> welcome : Kembali / Back
service -> confirm : Umum / General
service -> confirm : Gigi / Dental
service -> confirm : KIA / Mother and child
service -> confirm : Lansia / Elderly
confirm -> service : Ganti / Change
confirm -> ticket : Cetak / Print
```

```
../data/flow_v2.txt: 6 screens, 11 buttons
  goal 'ticket': 3 taps  welcome > service > confirm > ticket
  unreachable screens: none
  dead ends (no way on or back): none
```

Gambar 6.2 menggambar kedua graf itu. Tiket kini berjarak tiga ketukan dari layar sambutan, bukan empat (atau lima bagi pasien BPJS), setiap layar dapat dicapai, dan setiap layar mempunyai jalan maju atau kembali.



Gambar 6.2 Alur kios sebelum dan sesudah uji kertas: versi 1 memiliki layar tak terjangkau dan jalan buntu serta membutuhkan empat atau lima ketukan menuju tiket; versi 2 membutuhkan tiga, dan setiap layar mempunyai jalan kembali.

6.5 Wireframe dan Mockup yang Dapat Diklik

6.5.1 Wireframe

Wireframe adalah gambar layar yang memperlihatkan struktur tanpa gaya: kotak untuk isi, label untuk tombol, garis untuk teks, semuanya abu-abu. Wireframe menetapkan apa yang ada di setiap layar dan di mana, dan sengaja meninggalkan warna, gambar, dan kata-kata akhir, agar percakapan tetap berfokus pada tata letak dan hierarki. Di wireframe-lah layar-layar kertas menjadi cukup tepat untuk dibangun pengembang, dan di sanalah aturan visual Bab 8 (grid, hierarki, kontras) pertama kali diterapkan.

Untuk kios ini, wireframe menetapkan empat keputusan yang didorong uji kertas: satu pertanyaan per layar; tindakan utama berupa tombol selebar layar setinggi sekurang-kurangnya 72 piksel, dengan tindakan sekunder di bawahnya dalam gaya yang lebih tenang; huruf dan nama layanan pada setiap tombol layanan, agar pasien yang mendengar “B, gigi” di meja depan menemukan kata-kata yang sama di layar; dan pita kuning di bagian atas bertuliskan “PROTOTIPE”, agar tidak ada yang mengira sesi uji sebagai antrean sungguhan.

6.5.2 Mockup yang dapat diklik dan alat-alat tahun 2026

Mockup yang dapat diklik menautkan layar-layar yang sudah dirancang sehingga partisipan dapat mengetuknya satu per satu di perangkat sungguhan. Pada 2026 kebanyakan tim membuatnya dengan alat desain kolaboratif. Figma paling banyak dipakai; pada konferensi Config 2025 Figma memperkenalkan Figma Make, yang mengubah deskripsi tertulis atau rancangan yang sudah ada menjadi prototipe atau aplikasi yang bekerja, bersama Figma Sites untuk menerbitkan situs web, Figma Draw untuk ilustrasi, Figma Buzz untuk aset pemasaran, dan tata letak Grid yang mirip CSS [5]. Pada Config 2026 Figma menambahkan, antara lain, Figma Motion untuk animasi siap produksi, *code layers* yang membawa kode yang bekerja ke kanvas desain sebagai lapisan yang dapat disunting, serta pembaruan agen Figma dengan *skill* khusus, pencarian web, dan lampiran berkas [6]. Penpot adalah alternatif sumber terbuka yang utama: berjalan di peramban, dapat dipasang di server sendiri, dan menyimpan rancangan dalam format terbuka, hal yang penting bagi lembaga publik seperti puskesmas yang mungkin tidak ingin rancangan yang berhadapan dengan pasien tersimpan di server vendor asing [7].

Apa pun alatnya, mockup yang dapat diklik tetaplah prototipe tampilan dan rasa serta alur, bukan implementasi. Mockup menunjukkan bahwa “Cetak nomor” menuju layar tiket; ia tidak menunjukkan bahwa nomor benar-benar diterbitkan, bahwa dua pasien yang menekan bersamaan mendapat nomor berbeda, atau bahwa perkiraan waktu tunggunya masuk akal. Untuk pertanyaan-pertanyaan itu tim membangun prototipe berkode.

6.6 Membangun Prototipe Berkode

6.6.1 Apa yang harus dilakukannya

Prototipe berkode adalah program kecil yang berperilaku seperti produk di sepanjang jalur terpentingnya. Prototipe kios ini harus:

- menampilkan enam layar `flow_v2.txt` dalam bahasa Indonesia atau Inggris (`?lang=id` atau `?lang=en`);

- menerbitkan nomor antrean sungguhan yang terus naik per layanan, dimulai setelah orang-orang yang sudah menunggu, dan memperkirakan waktu tunggu dari banyaknya orang di depan dan rata-rata waktu per pasien;
- memakai tombol besar (sekurang-kurangnya setinggi 72 piksel, jauh di atas batas minimum 24 kali 24 piksel CSS pada kriteria keberhasilan 2.5.8 WCAG 2.2 [8]), teks gelap di atas putih, dan garis fokus yang terlihat;
- memasang pita "PROTOTIPE" di setiap layar;
- mencatat setiap tampilan layar beserta cap waktunya, agar sesi uji dapat diputar ulang dan diukur waktunya kemudian (Bab 7 menganalisis log semacam ini).

Prototipe ini *tidak* mencetak, tidak terhubung ke rekam medis puskesmas, dan tidak menyimpan apa pun tentang orang yang menekan tombolnya. Jumlah orang yang menunggu adalah nilai awal rekaan: ini prototipe, dan berpura-pura sebaliknya adalah bentuk penipuan terhadap partisipan.

6.6.2 Layar-layarnya

Layar-layar dihasilkan oleh satu fungsi, `render(screen, lang, code, number)`, yang mengembalikan halaman HTML lengkap. Menyimpan semua layar dalam satu modul, terpisah dari server web, memberi dua keuntungan: pengujian dapat memeriksa halaman tanpa menyalakan server, dan versi Java serta Python dapat dibandingkan byte demi byte. **Kode 6.5** adalah modul Python-nya.

Kode 6.5 `kiosk_pages.py`: teks, layanan, gaya, dan enam layar prototipe kios.

```
"""Screens of the puskesmas queue-kiosk prototype (shared by the server
and the tests). Every screen is plain HTML with large buttons."""
from html import escape

TEXT = {
    "id": {"title": "Antrean Puskesmas", "proto": "PROTOTIPE - bukan antrean asli",
           "welcome": "Selamat datang. Mau berobat ke mana hari ini?",
           "take": "Ambil nomor antrean", "help": "Bantuan",
           "hours": "Jadwal poli", "back": "Kembali", "choose": "Pilih layanan",
           "sure": "Benar layanan ini?", "print": "Cetak nomor",
           "change": "Ganti layanan", "your": "Nomor Anda",
           "ahead": "orang di depan Anda", "wait": "Perkiraan tunggu",
           "min": "menit", "done": "Selesai",
           "helptext": "Tekan tombol hijau besar. "
                       "Petugas di meja depan siap membantu.",
           "hourstext": "Senin-Kamis 07.30-14.00, Jumat 07.30-11.00, "
                        "Sabtu 07.30-12.00"},
    "en": {"title": "Health Center Queue", "proto": "PROTOTYPE - not a real queue",
           "welcome": "Welcome. Which service do you need today?",
           "take": "Take a queue number", "help": "Help",
           "hours": "Clinic hours", "back": "Back", "choose": "Choose a service",
           "sure": "Is this the right service?", "print": "Print my number",
           "change": "Change service", "your": "Your number",
           "ahead": "people ahead of you", "wait": "Estimated wait",
           "min": "min", "done": "Done",
           "helptext": "Press the big green button. "
                       "Staff at the front desk can help.",
           "hourstext": "Mon-Thu 07:30-14:00, Fri 07:30-11:00, Sat 07:30-12:00"},
}
# code: (Indonesian name, English name, people ahead, minutes per patient)
SERVICES = {"A": ("Umum", "General", 14, 6), "B": ("Gigi", "Dental", 3, 15),
            "C": ("KIA", "Mother and child", 6, 8),
            "D": ("Lansia", "Elderly", 4, 7)}
STYLE = """body{font-family:Arial,sans-serif;margin:0;background:#fff;color:#12355B}
main{max-width:720px;margin:auto;padding:16px}
.proto{background:#E0A100;color:#000;text-align:center;padding:6px;font-size:18px}
h1{font-size:34px}p{font-size:24px}
.btn{display:block;min-height:72px;margin:14px 0;padding:18px;font-size:28px;
border-radius:14px;text-decoration:none;text-align:center;box-sizing:border-box}
.go{background:#0F7C7C;color:#fff}.alt{background:#F3F4F6;color:#12355B;
border:2px solid #12355B}.big{font-size:96px;font-weight:bold;margin:8px 0}
a:focus{outline:4px solid #E0A100;outline-offset:3px}"""
```

Kode 6.5 (Lanjutan) *kiosk_pages.py: teks, layanan, gaya, dan enam layar prototipe kios.*

```
def name(code, lang):
    s = SERVICES[code]
    return s[0] if lang == "id" else s[1]


def button(href, label, kind="go"):
    return f'<a class="btn {kind}" href="{href}">{escape(label)}</a>'


def render(screen, lang="id", code="A", number=0):
    t = TEXT[lang if lang in TEXT else "id"]
    q = f"?lang={lang}"
    if screen == "welcome":
        body = (f"<h1>{t['welcome']}</h1>" + button("/service" + q, t["take"])
                + button("/help" + q, t["help"], "alt")
                + button("/schedule" + q, t["hours"], "alt"))
    elif screen == "service":
        body = f"<h1>{t['choose']}</h1>" + "".join(
            button(f"/confirm{q}&s={c}", f"{c} - {name(c, lang)}")
            for c in SERVICES) + button("/" + q, t["back"], "alt")
    elif screen == "confirm":
        body = (f"<h1>{t['sure']}</h1><p class='big'>{name(code, lang)}</p>"
                + button(f"/ticket{q}&s={code}", t["print"])
                + button("/service" + q, t["change"], "alt"))
    elif screen == "ticket":
        ahead, per = SERVICES[code][2], SERVICES[code][3]
        body = (f"<h1>{t['your']}</h1><p class='big'>{code}-{number:03d}</p>"
                f"<p>{name(code, lang)}: {ahead} {t['ahead']}</p>"
                f"<p>{t['wait']}: {ahead * per} {t['min']}</p>"
                + button("/" + q, t["done"]))
    elif screen == "help":
        body = f"<h1>{t['help']}</h1><p>{t['helptext']}</p>" + button(
            "/" + q, t["back"], "alt")
    else:
        body = f"<h1>{t['hours']}</h1><p>{t['hourstext']}</p>" + button(
            "/" + q, t["back"], "alt")
    return (f'<!doctype html><html lang="{lang}"><head><meta charset="utf-8">'
            f'<meta name="viewport" content="width=device-width,initial-scale=1">'
            f"<title>{t['title']}</title><style>{STYLE}</style></head><body>"
            f'<div class="proto">{t["proto"]}</div><main>{body}</main>'
            "</body></html>")
```

Tiga hal perlu ditunjukkan. Teks kedua bahasa berada dalam satu kamus, sehingga terjemahan adalah perubahan data, bukan kode. Label tombol melewati `escape`, sehingga label yang memuat `<` atau `&` tidak dapat merusak halaman. Dan halaman menyatakan bahasanya dalam `<html lang="...">`, yang memungkinkan pembaca layar melafalkan bahasa Indonesia sebagai bahasa Indonesia, hal yang dikembangkan di Bab 11.

6.6.3 Servernya dalam Python

Kode 6.6 melayani layar-layar itu dengan Flask [9]. Setiap rute memanggil `show`, yang membaca bahasa dan kode layanan dari *query string*, menerbitkan nomor untuk layar tiket, mencatat tampilan, dan mengembalikan halaman.

Kode 6.6 `kiosk_proto.py`: prototipe kios yang dilayani Flask pada porta 4060.

```
"""Clickable prototype of the puskesmas queue kiosk, served with Flask.
Every screen view is logged to ../data/events.csv for the test session.
Usage: python kiosk_proto.py   then open http://localhost:4060/?lang=en"""
import time
from flask import Flask, request
from kiosk_pages import SERVICES, render

app = Flask(__name__)
next_number = {code: s[2] + 1 for code, s in SERVICES.items()}
LOG = "../data/events.csv"


def log(screen, lang):
    with open(LOG, "a", encoding="utf-8") as f:
        f.write(f"{time.time():.3f},{lang},{screen}\n")


def show(screen):
    lang = request.args.get("lang", "id")
    code = request.args.get("s", "A")
    if code not in SERVICES:
        code = "A"
    number = 0
    if screen == "ticket":
        number = next_number[code]
        next_number[code] += 1
    log(screen, lang)
    return render(screen, lang, code, number)


app.add_url_rule("/", "welcome", lambda: show("welcome"))
for s in ("service", "confirm", "ticket", "help", "schedule"):
    app.add_url_rule("/" + s, s, lambda s=s: show(s))

if __name__ == "__main__":
    app.run(port=4060)
```

Gelung di bagian bawah mendaftarkan lima rute dengan satu baris masing-masing; argumen bawaan `s=s` pada `lambda` menangkap nama layar saat itu, idiom Python yang baku, yang tanpanya setiap rute akan menampilkan layar terakhir dari gelung.

6.6.4 Prototipe yang sama dalam Java

Kode 6.7 adalah versi Java dari layar-layar tersebut, yang menghasilkan HTML persis sama dengan modul Python.

Kode 6.7 `KioskPages.java`: layar-layar yang sama dalam Java.

```
import java.util.LinkedHashMap;
import java.util.Map;

/** Screens of the puskesmas queue-kiosk prototype (shared by the server
 *  and the tests). Every screen is plain HTML with large buttons. */
public class KioskPages {
    static final Map<String, Map<String, String>> TEXT = Map.of(
        "id", Map.ofEntries(Map.entry("title", "Antrean Puskesmas"),
            Map.entry("proto", "PROTOTIPE - bukan antrean asli"),
            Map.entry("welcome",
                "Selamat datang. Mau berobat ke mana hari ini?"),
            Map.entry("take", "Ambil nomor antrean"), Map.entry("help", "Bantuan"),
            Map.entry("hours", "Jadwal poli"), Map.entry("back", "Kembali"),
            Map.entry("choose", "Pilih layanan"),
            Map.entry("sure", "Benar layanan ini?"),
            Map.entry("print", "Cetak nomor"), Map.entry("change", "Ganti layanan"),
            Map.entry("your", "Nomor Anda"),
            Map.entry("ahead", "orang di depan Anda"),
            Map.entry("wait", "Perkiraan tunggu"), Map.entry("min", "menit"),
            Map.entry("done", "Selesai"),
            Map.entry("helptext", "Tekan tombol hijau besar. "
                + "Petugas di meja depan siap membantu."),
            Map.entry("hourstext", "Senin-Kamis 07.30-14.00, Jumat 07.30-11.00, "
                + "Sabtu 07.30-12.00")),
        "en", Map.ofEntries(Map.entry("title", "Health Center Queue"),
            Map.entry("proto", "PROTOTYPE - not a real queue"),
            Map.entry("welcome", "Welcome. Which service do you need today?"),
            Map.entry("take", "Take a queue number"), Map.entry("help", "Help"),
            Map.entry("hours", "Clinic hours"), Map.entry("back", "Back"),
            Map.entry("choose", "Choose a service"),
            Map.entry("sure", "Is this the right service?"),
            Map.entry("print", "Print my number"),
            Map.entry("change", "Change service"), Map.entry("your", "Your number"),
            Map.entry("ahead", "people ahead of you"),
            Map.entry("wait", "Estimated wait"), Map.entry("min", "min"),
            Map.entry("done", "Done"),
            Map.entry("helptext", "Press the big green button. "
                + "Staff at the front desk can help."),
            Map.entry("hourstext", "Mon-Thu 07:30-14:00, Fri 07:30-11:00, "
                + "Sat 07:30-12:00")));
```

Kode 6.7 (Lanjutan) *KioskPages.java: layar-layar yang sama dalam Java.*

```
    /** code -> {Indonesian name, English name, people ahead, minutes each}. */
    static final Map<String, Object[]> SERVICES = new LinkedHashMap<>();
    static {
        SERVICES.put("A", new Object[] {"Umum", "General", 14, 6});
        SERVICES.put("B", new Object[] {"Gigi", "Dental", 3, 15});
        SERVICES.put("C", new Object[] {"KIA", "Mother and child", 6, 8});
        SERVICES.put("D", new Object[] {"Lansia", "Elderly", 4, 7});
    }
    static final String STYLE = String.join("\n",
        "body{font-family:Arial,sans-serif;margin:0;background:#fff;color:#12355B}",
        "main{max-width:720px;margin:auto;padding:16px}",
        ".proto{background:#E0A100;color:#000;text-align:center;padding:6px;"
            + "font-size:18px}",
        "h1{font-size:34px}p{font-size:24px}",
        ".btn{display:block;min-height:72px;margin:14px 0;padding:18px;"
            + "font-size:28px;",
        "border-radius:14px;text-decoration:none;text-align:center;"
            + "box-sizing:border-box}",
        ".go{background:#0F7C7C;color:#fff}.alt{background:#F3F4F6;color:#12355B;",
        "border:2px solid #12355B}.big{font-size:96px;font-weight:bold;"
            + "margin:8px 0}",
        "a:focus{outline:4px solid #E0A100;outline-offset:3px}");

    static String name(String code, String lang) {
        return (String) SERVICES.get(code)[lang.equals("id") ? 0 : 1];
    }

    static String escape(String s) {
        return s.replace("&", "&amp;").replace("<", "&lt;").replace(">", "&gt;")
            .replace("\"", "&quot;").replace("'", "&#x27;");
    }

    static String button(String href, String label, String kind) {
        return "<a class=\"btn " + kind + "\" href=\"" + href + "\">"
            + escape(label) + "</a>";
    }
```

Kode 6.7 (Lanjutan) *KioskPages.java: layar-layar yang sama dalam Java.*

```
    static String render(String screen, String lang, String code, int number) {
        if (!TEXT.containsKey(lang)) lang = "id";
        Map<String, String> t = TEXT.get(lang);
        String q = "?lang=" + lang, body;
        switch (screen) {
            case "welcome" -> body = "<h1>" + t.get("welcome") + "</h1>"
                + button("/service" + q, t.get("take"), "go")
                + button("/help" + q, t.get("help"), "alt")
                + button("/schedule" + q, t.get("hours"), "alt");
            case "service" -> {
                StringBuilder b =
                    new StringBuilder("<h1>" + t.get("choose") + "</h1>");
                for (String c : SERVICES.keySet())
                    b.append(button("/confirm" + q + "&s=" + c,
                        c + " - " + name(c, lang), "go"));
                body = b + button("/" + q, t.get("back"), "alt");
            }
            case "confirm" -> body = "<h1>" + t.get("sure")
                + "</h1><p class='big'>" + name(code, lang) + "</p>"
                + button("/ticket" + q + "&s=" + code, t.get("print"), "go")
                + button("/service" + q, t.get("change"), "alt");
            case "ticket" -> {
                int ahead = (int) SERVICES.get(code)[2];
                int per = (int) SERVICES.get(code)[3];
                body = "<h1>" + t.get("your") + "</h1><p class='big'>"
                    + String.format("%s-%03d", code, number) + "</p>"
                    + "<p>" + name(code, lang) + ": " + ahead + " "
                    + t.get("ahead") + "</p><p>" + t.get("wait") + ": "
                    + ahead * per + " " + t.get("min") + "</p>"
                    + button("/" + q, t.get("done"), "go");
            }
            case "help" -> body = "<h1>" + t.get("help") + "</h1><p>"
                + t.get("helptext") + "</p>"
                + button("/" + q, t.get("back"), "alt");
            default -> body = "<h1>" + t.get("hours") + "</h1><p>"
                + t.get("hourstext") + "</p>"
                + button("/" + q, t.get("back"), "alt");
        }
        return "<!doctype html><html lang=\"" + lang + "\"><head>"
            + "<meta charset=\"utf-8\"><meta name=\"viewport\" "
            + "content=\"width=device-width,initial-scale=1\">"
            + "<title>" + t.get("title") + "</title><style>" + STYLE
            + "</style></head><body><div class=\"proto\">" + t.get("proto")
            + "</div><main>" + body + "</main></body></html>";
    }
}
```

Kode 6.8 melayaninya dengan `com.sun.net.httpserver.HttpServer`, server HTTP kecil yang menjadi bagian JDK sejak Java 6 dan tidak membutuhkan pustaka apa pun [10]. Satu penangan melayani setiap jalur: ia memetakan jalur ke nama layar, menolak jalur yang tidak dikenal dengan status 404, mengurai *query string*, dan menuliskan halaman. Penghitung nomor yang sudah diterbitkan dan berkas log dijaga dengan `synchronized`, karena server JDK dapat menangani dua permintaan pada saat yang sama, dan dua pasien tidak boleh menerima nomor yang sama.

Kode 6.8 `KioskProto.java`: prototipe kios yang dilayani server HTTP bawaan JDK pada porta 4061.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;

/** Clickable prototype of the puskesmas queue kiosk, served with the JDK's
 *  built-in HTTP server. Every screen view is logged to ../data/events.csv.
 *  Usage: java -cp out KioskProto   then open http://localhost:4061/?lang=en */
public class KioskProto {
    static final Map<String, Integer> nextNumber = new HashMap<>();

    static Map<String, String> query(HttpExchange ex) {
        Map<String, String> m = new HashMap<>();
        String raw = ex.getRequestURI().getRawQuery();
        if (raw != null)
            for (String pair : raw.split("&")) {
                String[] kv = pair.split("=", 2);
                if (kv.length == 2) m.put(kv[0], kv[1]);
            }
        return m;
    }

    static synchronized void log(String screen, String lang) throws IOException {
        try (FileWriter w = new FileWriter("../data/events.csv", true)) {
            w.write(String.format(Locale.US, "%.3f,%s,%s%n",
                System.currentTimeMillis() / 1000.0, lang, screen));
        }
    }

    static void handle(HttpExchange ex) throws IOException {
        String path = ex.getRequestURI().getPath();
        String screen = path.equals("/") ? "welcome" : path.substring(1);
        byte[] out;
        int status = 200;
        if (!screen.matches("welcome|service|confirm|ticket|help|schedule")) {
            status = 404;
            out = "Not found".getBytes(StandardCharsets.UTF_8);
        } else {
            Map<String, String> q = query(ex);
            String lang = q.getOrDefault("lang", "id");
```

Kode 6.8 (Lanjutan) *KioskProto.java: prototipe kios yang dilayani server HTTP bawaan JDK pada porta 4061.*

```
            String code = q.getOrDefault("s", "A");
            if (!KioskPages.SERVICES.containsKey(code)) code = "A";
            int number = 0;
            if (screen.equals("ticket")) {
                synchronized (nextNumber) {
                    number = nextNumber.merge(code, 1, Integer::sum);
                }
            }
            log(screen, lang);
            out = KioskPages.render(screen, lang, code, number)
                .getBytes(StandardCharsets.UTF_8);
        }
        ex.getResponseHeaders().set("Content-Type", "text/html; charset=utf-8");
        ex.sendResponseHeaders(status, out.length);
        try (OutputStream os = ex.getResponseBody()) {
            os.write(out);
        }
    }

    public static void main(String[] args) throws IOException {
        for (Map.Entry<String, Object[]> e : KioskPages.SERVICES.entrySet())
            nextNumber.put(e.getKey(), (int) e.getValue()[2]);
        HttpServer server = HttpServer.create(new InetSocketAddress(4061), 0);
        server.createContext("/", KioskProto::handle);
        server.start();
        System.out.println("Kiosk prototype on http://localhost:4061/?lang=en");
    }
}
```

6.6.5 Menjalankannya dan mengambil tangkapan layar

Nyalakan server Python dari folder `python` dengan `python3 kiosk_proto.py` (pasang Flask lebih dahulu dengan `pip install -r ../requirements.txt`), atau server Java dari folder `java` dengan `java -cp out KioskProto`. Flask melaporkan:

```
 * Serving Flask app 'kiosk_proto'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
production WSGI server instead.
 * Running on http://127.0.0.1:4060
Press CTRL+C to quit
```

dan server Java mencetak `Kiosk prototype on http://localhost:4061/?lang=en`. Buka alamat itu di peramban dan klik satu per satu.

Untuk laporan, dan untuk membandingkan versi, ada gunanya mengambil tangkapan layar secara otomatis, menelusuri prototipe persis seperti yang dilakukan pasien. **Kode 6.9** melakukannya dengan Playwright, pustaka otomasi peramban yang mengendalikan peramban Chromium sungguhan [11]. Program ini membuka layar sambutan dalam ukuran seperti kios, mengambil nomor untuk poli gigi, dan menyimpan tangkapan layar pada setiap langkah.

Kode 6.9 `shoot.py`: menelusuri prototipe yang sedang berjalan dengan Playwright dan menyimpan tangkapan layar.

```
"""Walk through the running prototype like a patient and save screenshots.
Start the server first (python kiosk_proto.py), then:
    python shoot.py [base URL] [lang]
e.g. python shoot.py http://localhost:4060 en
Needs: pip install playwright   and   python -m playwright install chromium"""
import sys
from playwright.sync_api import sync_playwright

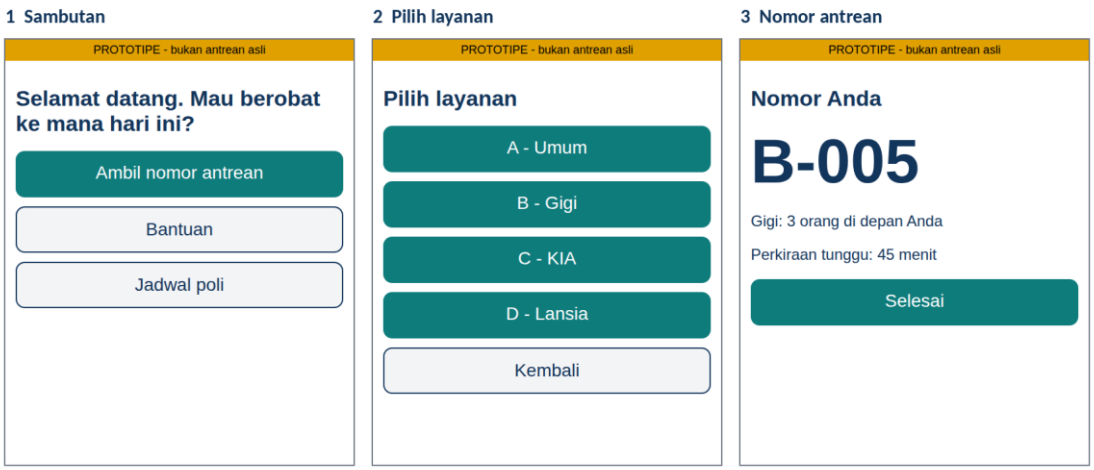

def main(args):
    base = args[0] if args else "http://localhost:4060"
    lang = args[1] if len(args) > 1 else "en"
    with sync_playwright() as p:
        browser = p.chromium.launch()
        page = browser.new_page(viewport={"width": 540, "height": 860})
        page.goto(f"{base}/?lang={lang}")
        page.screenshot(path=f"../screens/welcome_{lang}.png")
        page.click("a.btn.go")                          # take a number
        page.screenshot(path=f"../screens/service_{lang}.png")
        page.click("text=B - ")                         # dental
        page.click("a.btn.go")                          # print my number
        page.screenshot(path=f"../screens/ticket_{lang}.png")
        print("title:", page.title())
        print("ticket:", page.inner_text("p.big"))
        print("saved 3 screenshots in ../screens/")
        browser.close()


if __name__ == "__main__":
    main(sys.argv[1:])
```

Dengan server Flask menyala, `python3 shoot.py http://localhost:4060 en` mencetak:

```
title: Health Center Queue
ticket: B-004
saved 3 screenshots in ../screens/
```

Menjalankannya terhadap server Java pada porta 4061 mencetak tiga baris yang sama, karena kedua server menghasilkan halaman yang sama dan memulai penghitung yang sama. **Gambar 6.3** memperlihatkan ketiga tangkapan layar dalam versi bahasa Indonesia (`python3 shoot.py http://localhost:4060 id`). Poli gigi memiliki tiga orang yang menunggu, sehingga tiket baru pertama adalah B-004 dan perkiraan waktu tunggunya $3 \times 15 = 45$ menit.



Gambar 6.3 Tiga layar prototipe kios yang sedang berjalan, ditangkap oleh `shoot.py`: layar sambutan, pilihan layanan, dan tiket dengan banyaknya orang di depan serta perkiraan waktu tunggu.

Setiap kali dijalankan, prototipe juga menambahkan baris ke `data/events.csv`, satu baris per tampilan layar:

```
1790349318.935,en,welcome
1790349319.117,en,service
1790349319.241,en,confirm
```

Kolom pertama adalah waktu dalam detik sejak 1970, sehingga selisih dua baris adalah waktu yang dihabiskan di sebuah layar. Sesi uji sungguhan menghasilkan log ini dari ketukan partisipan, bukan dari Playwright, dan Bab 7 mengubah log semacam itu menjadi waktu penyelesaian tugas.

6.7 Menguji Kode Prototipe

Prototipe memang sekali pakai, tetapi tetap tidak boleh menyesatkan: prototipe yang memberi dua pasien nomor yang sama akan menghasilkan sesi uji tentang masalah yang keliru. **Kode 6.10** memeriksa kedua alur, atribut bahasa dan banyaknya tombol pada sebuah halaman, serta, melalui *test client* Flask, bahwa tiket-tiket berurutan terus naik dan setiap tampilan tercatat. Pengujian ini mengalihkan log ke berkas sementara agar menjalankan pengujian tidak mengotori data studi.

Kode 6.10 `test_proto.py`: pengujian pemeriksa alur dan prototipe.

```
"""Tests for the kiosk prototype and the flow checker.
Run: python -m unittest test_proto.py"""
import os
import tempfile
import unittest
import flow_check
import kiosk_proto
from kiosk_pages import render


class ProtoTest(unittest.TestCase):
    def test_flow_v1_problems(self):
        paths, unreachable, dead = flow_check.check(
            *flow_check.load("../data/flow_v1.txt"))
        self.assertEqual(unreachable, ["schedule"])
        self.assertEqual(dead, ["help"])
        self.assertEqual(len(paths["ticket"]) - 1, 4)

    def test_flow_v2_clean_and_shorter(self):
        paths, unreachable, dead = flow_check.check(
            *flow_check.load("../data/flow_v2.txt"))
        self.assertEqual((unreachable, dead), ([], []))
        self.assertEqual(len(paths["ticket"]) - 1, 3)

    def test_pages_are_labelled_and_big(self):
        html = render("service", "en")
        self.assertIn('<html lang="en">', html)
        self.assertEqual(html.count('class="btn'), 5)
        self.assertIn("min-height:72px", html)

    def test_ticket_through_the_server(self):
        kiosk_proto.LOG = os.path.join(tempfile.mkdtemp(), "events.csv")
        client = kiosk_proto.app.test_client()
        first = client.get("/ticket?lang=en&s=B").get_data(as_text=True)
        second = client.get("/ticket?lang=en&s=B").get_data(as_text=True)
        self.assertIn("B-004", first)
        self.assertIn("B-005", second)
        self.assertIn("Estimated wait: 45 min", first)
        with open(kiosk_proto.LOG, encoding="utf-8") as f:
            self.assertEqual(len(f.readlines()), 2)


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

```
Ran 4 tests in 0.012s

OK
```

Kode 6.11 melakukan pemeriksaan yang sama dalam Java, kecuali pengujian server, karena penghitung server Java berada di dalam `KioskProto`.

Kode 6.11 `ProtoTest.java`: pemeriksaan yang sama dalam Java.

```
import java.io.IOException;
import java.util.List;
import java.util.Map;

/** Tests for the kiosk pages and the flow checker, without a test library.
 *  Run: java -cp out ProtoTest */
public class ProtoTest {
    static int passed = 0;

    static void check(boolean ok, String name) {
        if (!ok) throw new AssertionError("FAILED: " + name);
        passed++;
    }

    static int count(String s, String part) {
        return s.split(java.util.regex.Pattern.quote(part), -1).length - 1;
    }

    public static void main(String[] args) throws IOException {
        FlowCheck v1 = FlowCheck.load("../data/flow_v1.txt");
        Map<String, List<String>> p1 = v1.tapsFrom();
        check(v1.unreachable(p1).equals(List.of("schedule")), "v1 unreachable");
        check(v1.deadEnds().equals(List.of("help")), "v1 dead end");
        check(p1.get("ticket").size() - 1 == 4, "v1 four taps");
        FlowCheck v2 = FlowCheck.load("../data/flow_v2.txt");
        Map<String, List<String>> p2 = v2.tapsFrom();
        check(v2.unreachable(p2).isEmpty() && v2.deadEnds().isEmpty(), "v2 clean");
        check(p2.get("ticket").size() - 1 == 3, "v2 three taps");
        String html = KioskPages.render("service", "en", "A", 0);
        check(html.contains("<html lang=\"en\">"), "lang attribute");
        check(count(html, "class=\"btn") == 5, "five buttons");
        String ticket = KioskPages.render("ticket", "en", "B", 4);
        check(ticket.contains("B-004") && ticket.contains("Estimated wait: 45 min"),
            "ticket text");
        System.out.println(passed + " checks passed");
    }
}
```

```
8 checks passed
```

6.8 Prototipe Berbantuan AI pada 2026

6.8.1 Apa yang dilakukan alat-alat itu

Pada 2026, satu kalimat dapat menghasilkan layar berfidelitas tinggi. Figma Make mengubah *prompt* atau rancangan yang sudah ada menjadi prototipe yang bekerja [5]. Stitch dari Google, yang diperkenalkan Google Labs pada Mei 2025, mengubah *prompt* teks dan gambar, termasuk sketsa tangan dan wireframe, menjadi rancangan UI dan kode *front-end* yang dapat diekspor ke Figma, dengan memanfaatkan kemampuan multimodal model Gemini [12]. Alat bantu pemrograman serbaguna menghasilkan aplikasi Flask atau React dari sebuah deskripsi. Tim mahasiswa mencoba sebuah *prompt*: “kios antrean layar sentuh untuk puskesmas di Indonesia, tombol besar, untuk pengguna lanjut usia”. Dalam waktu kurang dari semenit sebuah alat menghasilkan empat layar yang menarik.

Layar-layar itu memberi pelajaran ke dua arah. Tampilannya bersih, dengan jarak dan warna yang wajar. Namun layar-layar itu juga memuat halaman masuk dengan NIK, pertanyaan BPJS, *carousel* tips kesehatan, dan tautan “Bantuan” kecil di pojok: alat itu mereproduksi kios yang lazim, termasuk setiap masalah yang baru saja dihilangkan oleh uji kertas. Alat itu tidak mungkin

tahu bahwa pasien lanjut usia salah membaca pertanyaan BPJS; pengetahuan itu hanya ada di catatan para pengamat.

6.8.2 Memakai AI tanpa melewatkan pertanyaan

Pelajarannya bukan bahwa prototipe berbantuan AI itu buruk; pelajarannya adalah bahwa alat itu cepat di ujung kanan **Gambar 6.1**, dan kecepatan di ujung itu membuat ujung kiri menggoda untuk dilewati. Beberapa aturan menjaganya tetap berguna:

1. **Buat sketsa dan uji lebih dahulu.** Pakai kertas atau papan tulis untuk pertanyaan peran dan alur; libatkan AI setelah alurnya mantap dan Anda membutuhkan layar untuk menguji tampilan dan rasa.
2. **Tuliskan hasil riset di dalam *prompt*.** Masukkan temuan ke *prompt*: "tanpa login; satu pertanyaan per layar; tombol utama selebar layar setinggi minimal 72 px; empat layanan A–D dengan nama-nama ini; tampilkan orang di depan dan perkiraan waktu tunggu pada tiket". Alat AI hanya dapat mengikuti pengetahuan yang diberikan kepadanya.
3. **Periksa setiap layar hasil bangkitan terhadap berkas alur dan daftar periksa aksesibilitas Anda.** Rancangan bangkitan sering memuat teks kecil, kontras rendah, ikon tanpa label, dan komponen yang tampak dapat diklik padahal tidak. Pemeriksa kontras Bab 8 dan daftar periksa Bab 11 berlaku bagi layar semacam itu persis seperti bagi layar buatan tangan.
4. **Perlakukan kode bangkitan sebagai prototipe.** Kode yang dihasilkan dari *prompt* pada dasarnya belum ditinjau, belum diuji, dan belum aman; di bab ini prototipe berkode punya pengujian, dan prototipe bangkitan pun membutuhkannya.
5. **Perhatikan apa yang Anda unggah.** *Prompt* yang berisi data pasien sungguhan, dokumen internal, atau rancangan yang belum dirilis mengirimkannya ke pihak ketiga. Untuk puskesmas, utamakan alat yang dapat berjalan secara lokal atau alat desain yang dipasang sendiri, dan pakai contoh data rekaan.
6. **Uji dengan manusia.** Prototipe bangkitan tetaplah prototipe: taruh di depan partisipan, dengan tugas dan pengamatan yang sama seperti versi kertas.

> **PERINGATAN:** Prototipe rapi yang dihasilkan dalam semenit tampak sudah jadi, dan orang menanggapi hal yang tampak jadi dengan persetujuan yang sopan, bukan kritik yang jujur. Ketika menguji prototipe bangkitan AI pada tahap awal, pertimbangkan untuk menampilkannya dalam skala abu-abu atau mencetaknya di kertas, agar partisipan membicarakan gagasannya, bukan warnanya.

6.9 Membuat Prototipe Pesan Laundry

Tidak setiap antarmuka berupa layar. Empat perubahan yang direkomendasikan kepada Bu Rina di Subbab 5.9.3 berupa *pesan*: pesan penjemputan dengan perkiraan rentang harga, waktu selesai, dan jumlah potong; pesan "sudah selesai" dengan dua pilihan jam pengantaran; foto beserta jumlah potong; dan daftar harga berhuruf besar yang dikirim sebagai teks. Pesan dibuat prototipenya dengan cara yang sama seperti layar, pada fidelitas terendah yang dapat menjawab pertanyaannya.

Tim menulis setiap pesan di sebuah kartu dan, bersama enam pelanggan dari wawancara, memerankan percakapannya: seseorang berperan sebagai laundry, mengirim kartu-kartu itu sebagai pesan WhatsApp dari ponsel cadangan, sementara pelanggan menjawab seperti biasanya. Ini prototipe Wizard of Oz untuk layanan pesan: tanpa otomasi, hanya kata-kata yang tepat pada saat yang tepat. Dua temuan langsung muncul. Rentang harga "Rp28.000–35.000" disambut baik,

tetapi pelanggan bertanya apa yang membuatnya lebih mahal, sehingga pesan itu mendapat satu baris tambahan ("tambahan untuk selimut dan bed cover"). Dan dua pilihan jam pengantaran harus diberi nomor ("balas 1 untuk 16.00–18.00 atau 2 untuk 19.00–21.00"), karena pelanggan menjawab "yang malam" dan Dimas tidak tahu malam yang mana.

Pesan-pesan itu kemudian dipakai sungguhan selama dua minggu, dengan Bu Rina mengirimnya secara manual dari templat di WhatsApp Business. Baru setelah itu, ketika kata-katanya sudah mantap, otomasi layak dipertimbangkan. Urutannya penting: buat prototipe kata-katanya, lalu perilakunya, baru otomasinya.

6.10 Menerapkannya dalam Kenyataan

Rencana prototipe muat di sebuah kartu:

1. **Pertanyaannya.** Tuliskan pertanyaan mana yang dijawab prototipe: peran, tampilan dan rasa, atau implementasi, dan apa yang akan Anda putuskan dari jawabannya.
2. **Fidelitasnya.** Pilih fidelitas terendah yang dapat menjawabnya. Kertas untuk alur, wireframe untuk tata letak, mockup yang dapat diklik untuk tampilan dan rasa di perangkat sungguhan, kode untuk perilaku dan implementasi.
3. **Berkas alurnya.** Tuliskan alur sebagai baris-baris `screen -> next : label` dan jalankan `flow_check` sebelum setiap putaran uji. Perbaiki layar tak terjangkau dan jalan buntu sebelum partisipan menemukannya.
4. **Tugas-tugasnya.** Tiga sampai lima tugas yang realistis, ditulis dengan kata-kata pengguna, dengan akhir yang jelas ("Anda sudah memegang nomor").
5. **Partisipannya.** Lima sampai delapan orang per putaran, termasuk orang-orang di pinggiran: pengguna lanjut usia, pengguna pertama kali, orang yang membaca dengan lambat.
6. **Kejujurannya.** Beri label prototipe sebagai prototipe, minta persetujuan, dan jangan pernah mengumpulkan data pribadi sungguhan dalam prototipe.
7. **Iterasinya.** Setelah setiap putaran, ubah rancangan dan uji lagi. Dua atau tiga putaran pendek lebih baik daripada satu putaran panjang.

Kartu yang sama bekerja jauh di luar klinik. Untuk **ATM** sebuah bank daerah, bagian depan dari kardus dengan layar kertas dan papan tombol sungguhan menguji urutan langkah penarikan tunai; pertanyaan implementasi (apakah mesin uangnya merespons tepat waktu?) menjadi urusan insinyur bank. Untuk **aplikasi ojek**, mockup yang dapat diklik di ponsel, diuji di pangkalan pengemudi, menguji apakah pengemudi melihat titik jemput sebelum menerima pesanan. Untuk **pemesanan warung lewat WhatsApp**, kartu-kartu pesan yang diperankan bersama pelanggan tetap, seperti di Subbab 6.9, sudah cukup.

6.11 IMK dalam Praktik: Profitina

Keputusan-keputusan produk Profitina memperlihatkan ketiga pertanyaan prototipe dalam praktik.

Membuat prototipe perilaku: pengiriman seketika yang optimistis. Dalam komunitas *real-time* Profitina, kolom tulis langsung kosong dan kiriman langsung muncul, selagi masih dikirim; jika pengiriman gagal, teksnya dikembalikan sehingga tidak ada yang hilang. Apakah itu terasa cepat dan dapat dipercaya adalah pertanyaan tampilan dan rasa yang tidak dapat dijawab mockup statis, karena menyangkut *pengaturan waktu*. Prototipe berkode dengan tunda buatan dan kegagalan buatan adalah fidelitas yang tepat: partisipan mengirim kiriman, prototipe sesekali

sengaja gagal, dan pengamat melihat apakah orang menyadari teks yang dikembalikan dan memahami apa yang terjadi.

Gerbang mutu untuk konten bangkitan. Content Engine Profitina menyusun draf konten dengan model lokal (Qwen2.5) dan hanya menyajikan draf AI bila skornya benar-benar lebih baik daripada templat bawaan; bila tidak, templatlah yang disajikan. Itu disiplin yang sama dengan yang dianjurkan bab ini untuk prototipe bangkitan AI: hasil bangkitan harus mengalahkan pembanding yang sudah dikenal sebelum dipakai, dan "dibangkitkan" bukan sinonim "lebih baik".

Prototipe implementasi untuk widget. Widget komunitas disematkan di seluruh situs pemasaran dan di dalam aplikasi, dan mengalirkan pembaruan melalui *Server-Sent Events*. Apakah widget semacam itu sanggup mengikuti umpan yang ramai pada koneksi ponsel yang lambat adalah pertanyaan implementasi, yang dijawab dengan prototipe berkode kecil jauh sebelum kehalusan visual. Profitina memberi saran dan penggunalah yang memutuskan; prototipe-prototipenya, seperti prototipe lainnya, ada untuk mencari tahu sebelum berkomitmen.

6.12 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Memulai dengan fidelitas tinggi.** Prototipe pertama yang rapi menarik komentar tentang warna dan menyembunyikan masalah alur. Mulailah dengan kertas untuk pertanyaan peran dan urutan.
- **Membuat prototipe tanpa pertanyaan.** "Ayo bikin prototipe" menghasilkan demo. Tuliskan pertanyaannya dan keputusan yang dibantunya.
- **Membela prototipe.** Fasilitator yang menjelaskan rancangan selama sesi menguji penjelasannya, bukan rancangannya. Diamlah; biarkan partisipan berjuang; catat.
- **Layar tak terjangkau dan jalan buntu.** Keduanya mudah tercipta di atas kertas dan di alat desain. Tuliskan alurnya dan periksa dengan `flow_check` sebelum setiap putaran.
- **Lupa memberi label prototipe.** Prototipe yang realistis di tempat umum dapat dikira layanan sungguhan. Pertahankan pita "PROTOTIPE" yang terlihat.
- **Data sungguhan dalam prototipe.** Prototipe yang meminta NIK atau menyimpan nama mengumpulkan data pribadi tanpa perlu. Pakai contoh rekaan.
- **Memercayai keluaran AI sebagai riset.** Layar bangkitan mereproduksi rancangan yang lazim, termasuk masalah-masalahnya yang lazim. Tuliskan temuan di *prompt*, lalu uji dengan manusia.
- **Memperlakukan kode prototipe sebagai kode produk.** Kode prototipe ditulis untuk menjawab pertanyaan dengan cepat; sebelum menjadi produk ia membutuhkan tinjauan, keamanan, pengujian, dan pemeriksaan aksesibilitas.
- **Mengotomasi sebelum kata-katanya berhasil.** Untuk layanan pesan, buat prototipe pesannya secara manual lebih dahulu; otomasikan hanya setelah pesannya mantap.

6.13 Ringkasan Bab

- Prototipe adalah representasi rancangan yang dibuat untuk menjawab pertanyaan. Tiga pertanyaan Houde dan Hill adalah peran, tampilan dan rasa, serta implementasi.
- Fidelitas harus sesuai dengan pertanyaannya (**Gambar 6.1**): sketsa dan prototipe kertas bertanya apakah ini hal yang tepat; mockup yang dapat diklik dan prototipe berkode bertanya apakah hal itu sudah benar.
- Prototipe kertas melibatkan fasilitator, "komputer" manusia yang diam, dan pengamat; Wizard of Oz memalsukan perilaku yang sulit, seperti suara, dengan manusia yang tersembunyi.

- Alur adalah graf. `flow_check` dalam Java dan Python (**Kode 6.2**, **Kode 6.3**) memakai penelusuran melebar untuk menemukan layar tak terjangkau, jalan buntu, dan ketukan paling sedikit menuju tujuan; uji kertas dan pemeriksa bersama-sama memperpendek jalur kios menuju tiket dari empat atau lima ketukan menjadi tiga (**Gambar 6.2**).
- Prototipe berkode mengimplementasikan jalur terpenting secara mendalam: kios menerbitkan nomor sungguhan, memperkirakan waktu tunggu, mencatat setiap tampilan, memakai tombol besar yang aksesibel, dan diberi label prototipe, dalam Flask dan dengan server HTTP JDK (**Gambar 6.3**).
- Alat berbantuan AI tahun 2026 (Figma Make, Stitch, alat bantu pemrograman) mencapai fidelitas tinggi dalam hitungan menit. Pakailah setelah alurnya mantap, tuliskan temuan riset di *prompt*-nya, periksa keluarannya, dan uji dengan manusia.

6.14 Soal Latihan

1. Untuk setiap prototipe berikut, sebutkan pertanyaan Houde dan Hill (peran, tampilan dan rasa, implementasi) mana yang terutama dijawabnya: (a) laptop yang tersambung ke printer termal dan mencetak “A-001” ketika sebuah tombol ditekan; (b) kios kardus dengan layar kertas; (c) mockup Figma yang ditampilkan di tablet di pintu masuk puskesmas; (d) satu pagi ketika seorang petugas berdiri di pintu masuk membawa papan klip dan membagikan nomor yang menyebutkan perkiraan waktu tunggu kepada pasien.
2. Tambahkan layar `language` ke `flow_v2.txt`, yang dapat dicapai dari `welcome` dengan tombol “English / Bahasa Indonesia” dan kembali ke `welcome`. Jalankan `flow_check` sebelum dan sesudah menambahkan jalan kembalinya, dan jelaskan perbedaan keluarannya.
3. Alur lengkap vendor membuat pasien BPJS melewati `welcome > service > bpjs > nik > confirm > ticket`. Berapa ketukan itu, dan mengapa `flow_check` melaporkan empat ketukan menuju tiket untuk `flow_v1.txt`? Apa artinya bagi pelaporan yang hanya menyebut lintasan terpendek?
4. Kembangkan `flow_check` (dalam Python atau Java) agar juga melaporkan, untuk setiap tujuan, lintasan sederhana *terpanjang* dari awal (lintasan tanpa layar berulang dengan ketukan terbanyak). Jalankan pada `flow_v1.txt`.
5. Ubah prototipe kios agar layar tiket juga menampilkan jam pasien sebaiknya kembali, dengan anggapan waktu saat ini pukul 07.30 dan perkiraan waktu tunggu pada tiket. Fungsi mana yang harus berubah, dan mengapa lebih baik waktu saat ini dilewatkan sebagai parameter daripada membaca jam di dalam `render`?
6. Tulis *prompt* untuk alat prototipe AI yang menghasilkan kios `flow_v2.txt`, dengan memakai sekurang-kurangnya lima temuan dari Subbab 6.3 dan 6.4. Lalu sebutkan empat hal yang akan Anda periksa pada layar bangkitannya sebelum menunjukkannya kepada partisipan.
7. Rencanakan uji Wizard of Oz untuk pilihan suara pada laundry di Bab 5, ketika pelanggan mengirim pesan suara untuk memesan. Siapa penyihirnya, apa yang dipercaya atau disetujui pelanggan, apa tugas-tugasnya, dan apa yang harus dinyatakan dalam teks persetujuan?

6.15 Penyelesaian Soal Latihan

1. (a) Implementasi: prototipe ini memberi tahu apakah nomor dapat dicetak, tidak berkata apa pun tentang apakah pasien memahami kiosnya. (b) Peran dan banyak bagian tampilan dan rasa (ukuran, urutan, kata-kata), tanpa implementasi. (c) Tampilan dan rasa di layar yang realistis di tempat yang sebenarnya, serta sebagian peran; tanpa implementasi. (d) Peran: “kios” manusia menguji apakah memberi tahu pasien perkiraan waktu tunggu mengubah jam kedatangan

mereka dan apakah mereka pergi lalu kembali, sebelum apa pun dibangun. Jenis terakhir ini kadang disebut *prototipe layanan*.

2. Menambahkan hanya `welcome -> language : English / Bahasa Indonesia` menjadikan `language` jalan buntu: pemeriksa melaporkan `dead ends (no way on or back): language`. Menambahkan `language -> welcome : Kembali / Back` menghilangkannya, dan laporan kembali menjadi `none`. Banyaknya layar naik menjadi 7 dan banyaknya tombol menjadi 12 lalu 13, sedangkan jalur menuju tiket tetap tiga ketukan.

3. Lima ketukan. `flow_check` melaporkan empat karena ia melaporkan lintasan *terpendek*, yang melewati `bpjs -> confirm` (tombol "bayar sendiri") dan melompati layar NIK. Pasien BPJS, yang di puskesmas itu merupakan mayoritas pasien pagi, akan selalu menempuh jalur yang lebih panjang. Lintasan terpendek adalah batas bawah; laporan sebaiknya juga mempertimbangkan jalur yang benar-benar ditempuh pengguna yang paling umum, dan Soal 4 mulai menanganinya.

4. Penelusuran mendalam yang mencoba setiap lintasan tanpa layar berulang dan menyimpan yang terpanjang ke setiap tujuan:

```
def longest(start, goal, links, path=None):
    path = (path or []) + [start]
    if start == goal:
        return path
    best = None
    for nxt, _ in links.get(start, []):
        if nxt not in path:
            p = longest(nxt, goal, links, path)
            if p and (best is None or len(p) > len(best)):
                best = p
    return best
```

Untuk `flow_v1.txt`, `longest("welcome", "ticket", links)` mengembalikan `welcome > service > bpjs > nik > confirm > ticket`, lima ketukan. Penelusuran ini eksponensial pada kasus terburuk, yang tidak masalah untuk prototipe sepuluh layar tetapi tidak untuk situs web dengan ribuan halaman. Dalam Java, rekursi yang sama memakai `List<String>` untuk lintasannya.

5. Hanya cabang tiket pada `render` yang berubah, ditambah pemanggilnya: `render` mendapat parameter `now` (menit sejak tengah malam, misalnya 450 untuk 07.30), dan tiket menampilkan `now + ahead * per` dalam format jam dan menit; untuk tiket poli gigi di Subbab 6.6.5, 450 + 45 = 495 menit, yaitu 08.15. Server melewatkan waktu sebenarnya; pengujian melewatkan waktu tetap. Membaca jam di dalam `render` akan membuat keluarannya berubah dari menit ke menit, sehingga pengujian tidak dapat memeriksanya dan halaman Java serta Python tidak lagi dapat dibandingkan byte demi byte.

6. Contoh *prompt*: "Rancang empat layar untuk kios antrean layar sentuh di puskesmas Indonesia. Tanpa login dan tanpa pertanyaan asuransi. Satu pertanyaan per layar. Layar sambutan: satu tombol utama selebar layar 'Ambil nomor antrean' setinggi minimal 72 px, dan dua tombol sekunder 'Bantuan' dan 'Jadwal poli'. Layar layanan: empat tombol selebar layar 'A - Umum', 'B - Gigi', 'C - KIA', 'D - Lansia' dan tombol 'Kembali'. Layar konfirmasi yang mengulang layanan terpilih dengan teks sangat besar. Layar tiket dengan nomor bertulisan sangat besar, banyaknya orang di depan, dan perkiraan waktu tunggu dalam menit. Teks biru tua di atas putih, ukuran teks minimal 24 px, garis fokus yang terlihat, pita kuning 'PROTOTIPE' di atas. Bahasa Indonesia." Pemeriksaan: setiap layar punya jalan maju atau kembali (tuliskan berkas alurnya dan jalankan `flow_check`); kontras teks minimal 4,5:1 (pemeriksa Bab 8); tidak ada kontrol kecil yang hanya

berupa ikon; tidak ada fitur tambahan yang sudah dihapus berdasarkan riset (login, BPJS, *carousel*).

7. Penyihirnya Bu Rina (atau Dimas), yang mendengarkan pesan suara dan membalas secara manual dengan pesan-pesan prototipe di Subbab 6.9; tidak ada yang diotomasi. Pelanggan setuju mencoba "cara baru memesan lewat pesan suara" dan diberi tahu bahwa ada orang yang mendengarkan dan membalas. Tugas: "pesan penjemputan untuk besok malam lewat pesan suara", "ubah penjemputan menjadi lusa", "tanyakan harga cuci selimut". Teks persetujuan menyatakan bahwa pesan suara hanya didengarkan staf laundry, dihapus setelah pesanan dicatat, dan bahwa pelanggan dapat berhenti kapan saja dan memesan seperti biasa; teks itu juga menanyakan secara terpisah apakah pesan suara boleh disimpan selama masa studi.

Daftar Pustaka

[1] S. Houde and C. Hill, "What do prototypes prototype?," in *Handbook of Human-Computer Interaction*, 2nd ed., M. Helander, T. K. Landauer, and P. Prabhu, Eds. Amsterdam, The Netherlands: Elsevier, 1997, pp. 367–381.

[2] B. Buxton, *Sketching User Experiences: Getting the Design Right and the Right Design*. San Francisco, CA, USA: Morgan Kaufmann, 2007.

[3] C. Snyder, *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. San Francisco, CA, USA: Morgan Kaufmann, 2003.

[4] J. F. Kelley, "An iterative design methodology for user-friendly natural language office information applications," *ACM Transactions on Office Information Systems*, vol. 2, no. 1, pp. 26–41, 1984.

[5] Figma, "Config 2025 Launches Deepen Figma's Design Capabilities As Its Platform Expands," press release, May 7, 2025. [Online]. Available: https://www.figma.com/blog/config-2025-press-release/

[6] Figma Learn, "What's new from Config 2026," Help Center, Jun. 2026. [Online]. Available: https://help.figma.com/hc/en-us/articles/39582753756695-What-s-new-from-Config-2026

[7] Penpot, "Release Notes," accessed Sep. 2026. [Online]. Available: https://penpot.app/release-notes

[8] W3C, *Web Content Accessibility Guidelines (WCAG) 2.2*, W3C Recommendation, Oct. 5, 2023, latest version accessed Sep. 2026. [Online]. Available: https://www.w3.org/TR/WCAG22/

[9] Pallets, "Flask Documentation (3.1.x)," accessed Sep. 2026. [Online]. Available: https://flask.palletsprojects.com/

[10] 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

[11] Microsoft, "Playwright for Python," documentation, accessed Sep. 2026. [Online]. Available: https://playwright.dev/python/

[12] Google for Developers, "From idea to app: Introducing Stitch, a new way to design UIs," May 20, 2025. [Online]. Available: https://developers.googleblog.com/stitch-a-new-way-to-design-uis/

Bab 7

Evaluasi dan Pengujian Usabilitas

Sebuah rancangan adalah klaim bahwa orang dapat memakainya. Evaluasi adalah cara memeriksa klaim itu. Bab ini mengajarkan metode-metode yang dipakai tim pada 2026 untuk mengetahui apakah sebuah antarmuka dapat dipakai dengan baik, seberapa baik, dan apa yang harus diperbaiki lebih dahulu. Di akhir bab Anda akan mampu menjalankan evaluasi heuristik dan menilai tingkat keparahan temuannya; merencanakan dan memandu uji usabilitas *think-aloud* dengan tugas-tugas yang realistis; mengukur keberhasilan tugas, waktu, dan kesalahan, serta melaporkan tingkat keberhasilan dengan selang kepercayaan *adjusted-Wald* dan waktu dengan rata-rata geometrik; menghitung skor *System Usability Scale* (SUS) dan UMUX-Lite yang hanya dua butir, lalu menafsirkannya terhadap tolok ukur yang sudah terbit; menjelaskan mengapa lima sampai delapan partisipan per putaran menemukan sebagian besar masalah, dan kapan tidak; serta menjalankan dan menganalisis uji A/B dengan uji-z dua proporsi, selang kepercayaan untuk selisihnya, dan perhitungan ukuran sampel. Setiap perhitungan dikerjakan dengan tangan lebih dahulu, lalu dengan program Java dan Python.

7.1 Pesan Dulu di Warung Bu Tini

Warung Bu Tini menjual nasi campur dari ruang depan yang sempit di dekat sebuah gedung perkantoran dan kampus di Surabaya. Antara pukul dua belas dan satu, antrean mencapai jalan, dan pekerja kantor yang hanya punya waktu makan siang empat puluh lima menit sering menyerah lalu berjalan ke restoran cepat saji. Arif, anak Bu Tini yang menjadi mahasiswa informatika tahun kedua, membuat halaman web kecil agar pelanggan dapat memesan pada pagi hari dan mengambil pada jam yang dipilih: memilih lauk, memilih jumlah, memilih jam ambil, membayar dengan QRIS atau di kasir, lalu menerima konfirmasi.

Arif sudah mengujinya sendiri berkali-kali, dan halaman itu berfungsi. Teman-temannya mencobanya dan bilang bagus. Bu Tini sudah siap mencetak papan berisi kode QR menuju halaman itu. Sebelum ia melakukannya, dosen Arif mengajukan pertanyaan sederhana: *dari mana kamu tahu pelanggan ibumu bisa memakainya?* Arif tidak tahu. Ia tahu bahwa *dirinya* bisa memakainya, dan bahwa teman-temannya tidak mengeluh. Keduanya bukan bukti tentang seorang pensiunan guru berusia enam puluh tahun yang setiap hari membeli makan siang untuk suaminya, atau tentang pekerja kantor yang memesan lewat ponsel kecil sambil berjalan di antara dua rapat.

Bab ini mengikuti Arif saat ia mencari tahu. Pertama, dua teman sekelasnya memeriksa halaman itu terhadap sekumpulan prinsip, yaitu *evaluasi heuristik*. Lalu delapan pelanggan sungguhan mencoba empat tugas sambil berpikir dengan bersuara, dan Arif mengukur apa yang terjadi. Jawabannya ternyata lebih menarik daripada “berfungsi” atau “tidak berfungsi”: sebagian besar pelanggan menyelesaikan sebagian besar tugas, skor kepuasan rata-ratanya hampir persis sama dengan produk rata-rata, dan di balik rata-rata itu tersembunyi dua pelanggan lanjut usia yang hampir gagal memakai halaman tersebut. Setelah perbaikan, uji A/B selama dua minggu terhadap label tombol pesan memperlihatkan cara mengukur dampak sebuah perubahan pada ratusan pelanggan yang tidak pernah bertemu perancangnya.

7.2 Gagasannya dalam Bahasa Sederhana

7.2.1 Apa arti "mudah dipakai"

> **DEFINISI.** *Usabilitas* (*usability*) adalah sejauh mana sebuah sistem dapat dipakai oleh pengguna tertentu untuk mencapai tujuan tertentu dengan *efektivitas* (ketepatan dan kelengkapan), *efisiensi* (sumber daya seperti waktu dan tenaga yang dikeluarkan), dan *kepuasan* (tanggapan pengguna, dari rasa nyaman sampai penerimaan) dalam konteks penggunaan tertentu [1]. *Evaluasi usabilitas* adalah setiap kegiatan sistematis yang menilai sebuah rancangan terhadap kriteria tersebut.

Definisi dari ISO 9241-11:2018 ini mengandung dua pelajaran. Pertama, usabilitas bukan satu angka: sebuah rancangan bisa efektif tetapi lambat, atau cepat tetapi menjengkelkan, dan masing-masing membutuhkan ukurannya sendiri. Kedua, usabilitas hanya ada untuk *pengguna, tujuan, dan konteks tertentu*: halaman pemesanan yang sama bisa sangat mudah dipakai oleh mahasiswa di depan meja dan hampir tidak dapat dipakai oleh pelanggan lanjut usia dengan ponsel kecil di bawah terik matahari. Karena itu setiap evaluasi harus menyatakan siapa yang diuji, melakukan apa, dan di mana.

7.2.2 Dua keluarga evaluasi

Evaluasi terbagi dalam dua keluarga. **Metode inspeksi** memakai pakar sebagai pengganti pengguna: para evaluator menelusuri rancangan untuk mencari pelanggaran terhadap prinsip-prinsip yang dikenal. Metode ini cepat dan murah serta menemukan banyak masalah, tetapi juga melaporkan sebagian masalah yang tidak pernah dialami pengguna sungguhan, dan melewatkan masalah yang berasal dari pengetahuan serta kebiasaan pengguna sendiri. **Metode empiris** memakai pengguna sungguhan yang mengerjakan tugas sungguhan: uji usabilitas, kuesioner, analitik, dan eksperimen. Metode ini lebih lambat dan lebih mahal per temuan, tetapi yang ditemukannya nyata.

Pembedaan kedua menyangkut tujuan. Evaluasi **formatif** dilakukan selama perancangan dan bertanya *apa yang salah dan bagaimana memperbaikinya?*; keluaran utamanya daftar masalah. Evaluasi **sumatif** dilakukan pada suatu tonggak dan bertanya *seberapa baik?*; keluaran utamanya angka, sering dibandingkan dengan target, pesaing, atau versi sebelumnya. Uji Arif bersifat formatif; uji A/B di akhir bab bersifat sumatif. Sebagian besar angka di bab ini melayani kedua tujuan.

7.3 Evaluasi Heuristik

7.3.1 Metodenya

Evaluasi heuristik diperkenalkan Nielsen dan Molich sebagai metode "hemat": beberapa evaluator memeriksa sebuah antarmuka dan menilai kesesuaiannya dengan sekumpulan kecil prinsip usabilitas yang diakui, yaitu *heuristik* [2]. Bab 3 menyajikan sepuluh heuristik Nielsen (visibilitas status sistem, kecocokan dengan dunia nyata, kendali dan kebebasan pengguna, konsistensi dan standar, pencegahan kesalahan, mengenali alih-alih mengingat, keluwesan dan efisiensi, desain yang estetis dan minimalis, membantu pengguna pulih dari kesalahan, serta bantuan dan dokumentasi). Di sini kita memakainya sebagai alat evaluasi.

Prosedurnya terdiri atas empat langkah:

1. **Beri pengarahan kepada evaluator** tentang pengguna, tujuan, dan konteksnya. Teman-teman Arif diberi tahu: pelanggannya pekerja kantor dan warga dari segala usia, kebanyakan memakai ponsel, memesan antara pukul 07.00 dan 11.00 untuk diambil antara pukul 11.30 dan 13.30.
2. **Periksa secara mandiri.** Setiap evaluator menelusuri antarmuka sekurang-kurangnya dua kali, sekali untuk merasakan alurnya dan sekali untuk memeriksa setiap layar, lalu mencatat setiap masalah beserta heuristik yang dilanggarnya dan tempat terjadinya. Bekerja sendiri-sendiri itu penting: evaluator yang terlalu cepat berdiskusi cenderung sepakat pada segelintir masalah yang sama.
3. **Gabungkan.** Para evaluator menggabungkan daftar mereka menjadi satu dan membuang yang ganda.
4. **Nilai keparahannya.** Setiap evaluator menilai setiap masalah secara mandiri, lalu nilainya dirata-rata.

Nielsen dan Molich menemukan bahwa satu evaluator hanya menemukan sebagian kecil masalah dan bahwa menggabungkan beberapa evaluator menemukan jauh lebih banyak; itulah sebabnya tiga sampai lima evaluator merupakan anjuran yang lazim [2].

7.3.2 Tingkat keparahan

Daftar masalah tidak berguna tanpa prioritas. Skala keparahan yang umum, juga dari Nielsen, berkisar dari 0 sampai 4 [3]:

Tabel 7.1 Skala keparahan masalah usabilitas, dengan contoh dari halaman pemesanan.

Nilai	Arti	Contoh dari laundry, warung, atau klinik
0	Bukan masalah usabilitas	Warna yang tidak disukai evaluator
1	Kosmetik: perbaiki hanya jika sempat	Foto menu lambat dimuat pada koneksi lambat
2	Kecil: prioritas rendah	Harga baru berubah setelah keluar dari kolom jumlah
3	Besar: penting untuk diperbaiki	Kode QRIS terlalu kecil untuk dipindai dari ponsel lain
4	Bencana: wajib diperbaiki sebelum dirilis	Tombol kembali mengosongkan keranjang

Keparahan memadukan tiga faktor: *frekuensi* (berapa banyak pengguna yang mengalaminya), *dampak* (seberapa sulit diatasi), dan *kegigihan* (apakah pengguna mengalaminya sekali lalu belajar, atau berulang-ulang). Masalah yang dialami semua orang sekali dan mudah diatasi bisa lebih ringan daripada masalah yang dialami sedikit pengguna setiap kali dan tidak dapat diatasi.

Dua teman Arif menemukan enam masalah dalam satu sore, termasuk ikon menu tanpa label yang menyembunyikan tautan “ubah pesanan” (melanggar *mengenali alih-alih mengingat*) dan pesan galat “Invalid input” (melanggar *membantu pengguna mengenali, mendiagnosis, dan pulih dari kesalahan*). Mereka melewatkan tiga masalah yang kemudian ditemukan uji pengguna, termasuk kenyataan bahwa konfirmasi tidak menyebut kapan makanan siap, yang hanya disadari pelanggan yang benar-benar mengatur makan siangnya berdasarkan informasi itu.

7.4 Uji Usabilitas dengan Pengguna Sungguhan

7.4.1 Merencanakan uji

Uji usabilitas menempatkan pengguna yang mewakili di depan rancangan dengan tugas-tugas yang realistis dan mengamati apa yang mereka lakukan. Buku pegangan Rubin dan Chisnell, yang masih menjadi panduan baku, menyusun rencana uji di sekitar tujuan, pertanyaan riset,

partisipan, metode, daftar tugas, lingkungan, data yang dikumpulkan, dan laporan [4]. Untuk warung ini rencananya singkat:

- **Pertanyaan riset.** Dapatkah pelanggan memesan, mengubah pesanan, dan membayar tanpa bantuan? Di mana mereka ragu atau gagal? Apakah halaman ini sama mudahnya dipakai oleh pelanggan lanjut usia?
- **Partisipan.** Delapan pelanggan tetap yang direkrut di kasir: empat pekerja kantor, dua mahasiswa, dan dua warga pensiunan berusia di atas enam puluh tahun (P3 dan P6), masing-masing diberi makan siang gratis sebagai ucapan terima kasih. Teks persetujuannya mengikuti Subbab 5.3.3, dengan perekaman layar dan suara ditanyakan terpisah.
- **Tempat.** Setiap partisipan memakai ponselnya sendiri, di warung setelah jam ramai, agar ukuran layar, koneksi, dan kebiasaannya nyata.
- **Tugas.** Empat tugas, masing-masing dengan akhir yang jelas:
  - T1: *Cari tahu menu hari ini dan harga satu porsi nasi campur ayam.*
  - T2: *Pesan dua porsi nasi campur ayam untuk diambil pukul 12.00.*
  - T3: *Anda baru tahu seorang rekan juga mau makan siang. Ubah pesanan Anda menjadi tiga porsi.*
  - T4: *Bayar pesanan Anda dengan QRIS.* (Dipakai kode QRIS uji; tidak ada uang yang berpindah.)
- **Ukuran.** Berhasil atau gagalnya setiap tugas, waktu pengerjaan, banyaknya kesalahan, kuesioner kepuasan di akhir (SUS dan UMUX-Lite), dan masalah-masalah yang teramati.

Tugas ditulis sebagai tujuan dengan kata-kata pelanggan, tidak pernah sebagai instruksi dengan kata-kata antarmuka. “Ubah pesanan Anda menjadi tiga porsi” adalah tujuan; “ketuk ikon menu lalu pilih Edit” adalah sekumpulan instruksi yang tidak menguji apa pun selain kemampuan membaca.

7.4.2 Berpikir dengan bersuara

Dalam uji *think-aloud*, partisipan mengucapkan apa yang ia pikirkan sambil bekerja: apa yang ia cari, apa yang ia harapkan, apa yang mengejutkannya. Metode ini bertumpu pada analisis Ericsson dan Simon tentang laporan verbal, yang menunjukkan bahwa orang dapat melaporkan isi memori kerjanya sambil mengerjakan tugas tanpa banyak mengubah cara ia mengerjakannya, asalkan ia diminta *mengucapkan*, bukan *menjelaskan* [5]. Itulah sebabnya instruksi fasilitator berbunyi “silakan ucapkan apa pun yang Anda pikirkan sambil jalan”, bukan “silakan jelaskan mengapa Anda melakukan sesuatu”.

Tugas fasilitator sebagian besar adalah diam. Ketika partisipan terdiam, dorongan netral (“apa yang sedang Anda pikirkan?”) sudah cukup. Ketika partisipan meminta bantuan (“di mana saya mengubahnya?”), fasilitator mengembalikan pertanyaan itu (“menurut Anda di mana seharusnya?”) dan baru memberi bantuan ketika partisipan benar-benar buntu dan sisa sesi akan terbuang, lalu mencatat tugas itu sebagai kegagalan. Menggiring, menjelaskan, atau membela rancangan, bahkan dengan sebuah desahan, merusak datanya.

7.4.3 Apa yang terjadi

Empat momen dari delapan sesi memperlihatkan apa yang diungkap *think-aloud*:

- P3, pensiunan guru, pada T3: “Ubah… nggak ada ‘ubah’. Mungkin saya harus pesan lagi? Nanti bayarnya dua kali.” Ia memesan untuk kedua kalinya, yang dihitung sebagai kegagalan T3 dan akan menghasilkan tagihan ganda.

- P6, pensiunan pegawai negeri, pada T2: “Daftar jamnya mulai jam tujuh pagi… ini kan makan siang, kok jam tujuh?” Ia menggulir selama dua puluh detik melewati jam-jam yang tidak akan dipilih siapa pun.
- P1, pekerja kantor, pada T3: “Saya tekan kembali untuk membetulkan jumlahnya, keranjang saya malah kosong. Bagus.” Tombol kembali pada peramban membuang isi keranjang.
- P5, mahasiswa, menyelesaikan setiap tugas dengan cepat dan di akhir berkata halaman itu “biasa saja, seperti yang lain”. Partisipan seperti P5 menunjukkan bahwa sebuah rancangan berfungsi bagi pengguna yang percaya diri; mereka tidak dapat menunjukkan bahwa rancangan itu berfungsi bagi semua orang.

7.5 Mengukur Kinerja

7.5.1 Keberhasilan tugas dan selang kepercayaannya

Ukuran paling dasar adalah apakah tugas diselesaikan. *Tingkat penyelesaian* sebuah tugas adalah banyaknya partisipan yang berhasil, x, dibagi banyaknya yang mencoba, n. Pada sampel kecil, selang kepercayaan biasa untuk proporsi berperilaku buruk: untuk 8 berhasil dari 8, selang itu menghasilkan “100% sampai 100%” yang tidak masuk akal. Sauro dan Lewis menganjurkan selang *adjusted-Wald* untuk tingkat penyelesaian dalam uji usabilitas, karena cakupannya baik bahkan untuk sampel yang sangat kecil dan untuk tingkat yang mendekati 0% atau 100% [6]. Selang ini menambahkan $z^2/2$ keberhasilan dan z^2 percobaan sebelum menghitung:

$$\hat{p}_{\text{adj}} = \frac{x + z^2/2}{n + z^2}, \qquad \hat{p}_{\text{adj}} \pm z\sqrt{\frac{\hat{p}_{\text{adj}}(1-\hat{p}_{\text{adj}})}{n + z^2}} \tag{7.1}$$

dengan $z = 1.96$ untuk kepercayaan 95%, dan selangnya dipotong ke rentang 0 sampai 1. Dengan $z^2 = 3.8415$, menambahkan $z^2/2 \approx 1.92$ keberhasilan dan $z^2 \approx 3.84$ percobaan hampir sama dengan menambahkan dua keberhasilan dan dua kegagalan, sehingga metode ini juga dikenal sebagai “tambahkan dua berhasil dan dua gagal”.

Dihitung dengan tangan untuk T1, ketika kedelapan partisipan berhasil: $\hat{p}_{\text{adj}} = (8 + 1.921)/(8 + 3.842) = 9.921/11.842 = 0.838$; setengah lebarnya $1.96 \times \sqrt{0.838 \times 0.162/11.842} = 1.96 \times 0.1071 = 0.210$; selangnya $0.838 \pm 0.210 = [0.628, 1.048]$, dipotong menjadi $[62.8\%, 100\%]$. Jadi “8 dari 8” hanya berarti bahwa, dengan kepercayaan 95%, sekurang-kurangnya sekitar 63% pelanggan seperti mereka akan menyelesaikan tugas itu. Untuk T3, ketika 4 dari 8 berhasil: $\hat{p}_{\text{adj}} = 5.921/11.842 = 0.500$, setengah lebar $1.96 \times \sqrt{0.25/11.842} = 0.285$, selang $[21.5\%, 78.5\%]$. Sampel kecil menghasilkan selang yang lebar; itu bukan cacat metodenya, melainkan pernyataan jujur tentang seberapa banyak yang dapat dikatakan delapan orang.

7.5.2 Waktu pengerjaan dan rata-rata geometrik

Waktu pengerjaan tugas itu menceng: sebagian besar partisipan membutuhkan waktu yang mirip, segelintir membutuhkan jauh lebih lama. Rata-rata aritmetika tertarik ke atas oleh segelintir yang lambat itu, dan pada sampel kecil satu partisipan yang sangat lambat bisa mendominasinya. Untuk sampel yang lebih kecil dari sekitar 25, Sauro dan Lewis menganjurkan *rata-rata geometrik*, yaitu akar pangkat n dari hasil kali waktu-waktu itu, yang dihitung melalui logaritma [6]:

$$\bar{t}_{\text{geo}} = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \ln t_i\right) \tag{7.2}$$

Untuk T4, tujuh waktu yang berhasil adalah 70, 41, 96, 55, 38, 47, dan 62 detik. Logaritma naturalnya 4,248, 3,714, 4,564, 4,007, 3,638, 3,850, dan 4,127, dengan jumlah 28,148 dan rata-rata 4,021; $e^{4.021} = 55.8$ detik. Rata-rata aritmetikanya $409/7 = 58.4$ detik; selisihnya kecil di sini karena tidak ada waktu yang ekstrem, tetapi untuk T2, dengan satu keberhasilan pada 140 detik, selisihnya membesar. Menurut kebiasaan, waktu percobaan yang gagal tidak dimasukkan ke dalam ukuran waktu, karena partisipan yang menyerah lebih awal tidak "membutuhkan" waktu itu untuk menyelesaikan tugas; kegagalannya sudah terhitung dalam tingkat penyelesaian.

7.5.3 Kesalahan

Kesalahan adalah setiap tindakan yang tidak mengarah ke tujuan dan harus dibatalkan atau disiasati partisipan: mengetuk tombol yang salah, memasukkan nilai yang tidak sah, kembali ke layar sebelumnya. Menghitung kesalahan per tugas memperlihatkan *di mana* sebuah rancangan menyesatkan, yang sering lebih berguna untuk memperbaikinya daripada waktu.

7.5.4 Hasilnya

`data/sessions.csv` memuat satu baris per partisipan dan tugas: berhasil (1 atau 0), detik, dan kesalahan. Bagian pertama program di Subbab 7.7 mencetak ringkasan tugas:

```
Task  success  95% CI (adj. Wald)  geo-mean time  errors
T1    8/8      62.8% - 100.0%      37.4 s     0.25
T2    7/8      50.8% -  99.9%      82.9 s     1.25
T3    4/8      21.5% -  78.5%      97.9 s     1.75
T4    7/8      50.8% -  99.9%      55.8 s     0.50
```

T3, mengubah pesanan, jelas merupakan masalahnya: hanya separuh partisipan yang berhasil, dan yang berhasil pun membutuhkan waktu paling lama dan membuat kesalahan paling banyak. Yang gagal adalah P1 (tombol kembali mengosongkan keranjang), P3 (memesan lagi), P6 (menyerah), dan P8 (sama sekali tidak menemukan tautan ubah).

7.6 Mengukur Kepuasan: SUS dan UMUX-Lite

7.6.1 System Usability Scale

System Usability Scale, yang diterbitkan Brooke pada 1996 sebagai skala usabilitas yang "cepat dan kasar", adalah kuesioner usabilitas baku yang paling luas dipakai [7]. Kuesioner ini memuat sepuluh pernyataan, dijawab dari 1 (sangat tidak setuju) sampai 5 (sangat setuju), yang bergantian antara rumusan positif dan negatif:

1. Saya rasa saya ingin sering memakai sistem ini.
2. Saya merasa sistem ini rumit tanpa perlu.
3. Saya rasa sistem ini mudah dipakai.
4. Saya rasa saya membutuhkan bantuan orang teknis untuk dapat memakai sistem ini.
5. Saya merasa berbagai fungsi dalam sistem ini terpadu dengan baik.
6. Saya rasa terlalu banyak ketidakkonsistenan dalam sistem ini.
7. Saya bayangkan kebanyakan orang akan belajar memakai sistem ini dengan sangat cepat.

8. Saya merasa sistem ini sangat merepotkan untuk dipakai.
9. Saya merasa sangat yakin ketika memakai sistem ini.
10. Saya perlu mempelajari banyak hal sebelum dapat mulai memakai sistem ini.

Untuk warung ini, "sistem" diganti dengan "halaman pemesanan", dan terjemahan bahasa Indonesianya diperiksa dengan menerjemahkannya kembali ke bahasa Inggris oleh orang kedua. Skornya dihitung dengan mengubah setiap jawaban menjadi sumbangan 0 sampai 4 lalu menskalakan jumlahnya menjadi 0–100. Jika a_i adalah jawaban butir i:

$$\text{SUS} = 2.5\left(\sum_{i\in\{1,3,5,7,9\}}(a_i - 1) + \sum_{i\in\{2,4,6,8,10\}}(5 - a_i)\right) \tag{7.3}$$

Butir ganjil bersifat positif, sehingga setuju menyumbang lebih banyak; butir genap bersifat negatif, sehingga tidak setuju menyumbang lebih banyak. Dihitung untuk P1, yang jawabannya 4, 2, 4, 1, 3, 2, 4, 2, 4, 2: sumbangan butir ganjil $3 + 3 + 2 + 3 + 3 = 14$; sumbangan butir genap $3 + 4 + 3 + 3 + 3 = 16$; jumlahnya 30; $\text{SUS} = 2.5 \times 30 = 75$.

Skor SUS *bukan* persentase, dan 70 tidak berarti "70% mudah dipakai". Maknanya datang dari perbandingan dengan produk lain. Dalam kumpulan besar studi SUS, skor rata-ratanya 68 [8], dan Sauro serta Lewis mengubah sebaran skor-skor itu menjadi *skala nilai berkurva* (*curved grading scale*) yang, misalnya, memberi nilai C untuk rata-rata 65,0 sampai 71,0, nilai A untuk 80,8 sampai 84,0, dan A+ untuk 84,1 ke atas [6].

7.6.2 UMUX-Lite

Ketika tidak ada waktu untuk sepuluh butir, misalnya dalam jendela sembul setelah transaksi, UMUX-Lite dua butir dari Lewis, Utesch, dan Maher adalah alternatif yang tervalidasi dengan baik [9]. Butir-butirnya, dijawab pada skala 7 titik, adalah "Kemampuan [sistem ini] memenuhi kebutuhan saya" dan "[Sistem ini] mudah dipakai". Dengan jawaban c dan e dari 1 sampai 7, skor mentahnya diskalakan ke 0–100, dan sebuah persamaan regresi menempatkannya pada skala SUS agar keduanya dapat dibandingkan [10]:

$$\text{UMUX}_{\text{raw}} = \frac{(c-1) + (e-1)}{12} \times 100, \qquad \widehat{\text{SUS}} = 0.65\,\text{UMUX}_{\text{raw}} + 22.9 \tag{7.4}$$

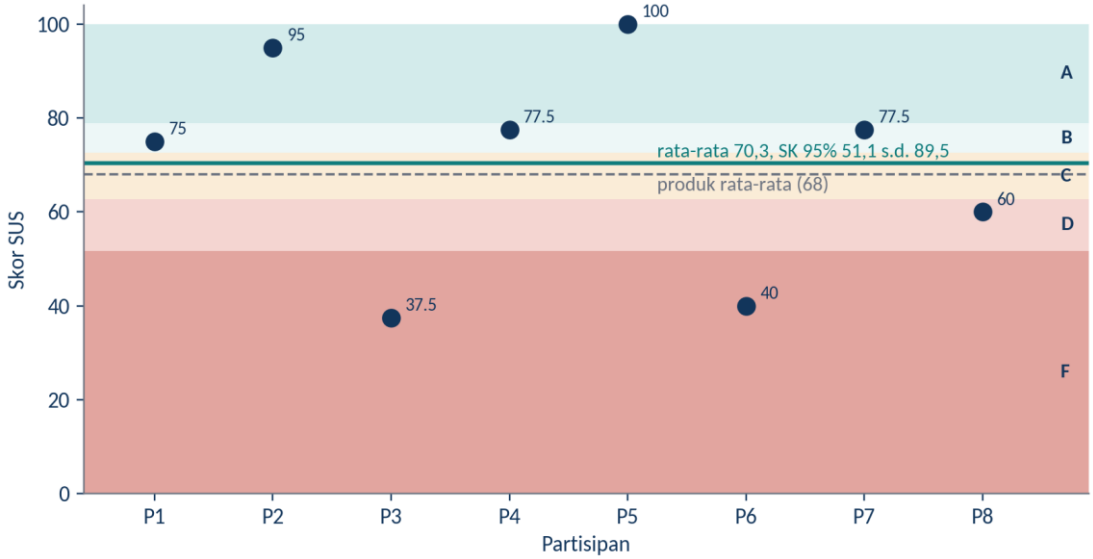
Untuk P1, $c = 6$ dan $e = 5$: skor mentah $= (5 + 4)/12 \times 100 = 75.0$, dan pada skala SUS $0.65 \times 75 + 22.9 = 71.65$.

7.6.3 Hasilnya

Program di Subbab 7.7 melanjutkan:

```
SUS per participant:
  P1:  75.0
  P2:  95.0
  P3:  37.5
  P4:  77.5
  P5: 100.0
  P6:  40.0
  P7:  77.5
  P8:  60.0
SUS mean 70.3 (95% CI 51.1 to 89.5), grade C; the average product scores 68
UMUX-Lite raw mean 74.0, on the SUS scale 71.0
```

Selang kepercayaannya adalah selang *t* dari Subbab 5.5.3, dihitung dari delapan skor. **Gambar 7.1** memetakan delapan skor itu terhadap pita-pita nilai. Rata-ratanya, 70,3, mendapat nilai C, sedikit di atas produk rata-rata; UMUX-Lite, yang dipetakan ke skala SUS, sepakat pada 71,0. Seandainya Arif hanya melaporkan rata-ratanya, ia akan berkata "kira-kira rata-rata". Gambarnya mengatakan sesuatu yang lebih penting: enam partisipan memberi skor 60 sampai 100, dan dua partisipan lanjut usia, P3 dan P6, memberi 37,5 dan 40, jauh di dalam pita F. Merekalah yang gagal pada T3 dan kesulitan pada T2. Rata-rata dapat menyembunyikan sekelompok orang yang gagal dilayani rancangan, dan biasanya yang tersembunyi justru orang-orang di pinggiran yang dibahas di Bab 5.



Gambar 7.1 Skor SUS kedelapan partisipan terhadap pita nilai berkurva; rata-ratanya (70,3) sedikit di atas produk rata-rata (68), tetapi kedua partisipan lanjut usia berada di pita F.

> **WAWASAN KUNCI:** Laporkan sebaran, bukan hanya rata-rata. Dengan delapan partisipan, grafik yang memuat kedelapan skor tidak memakan tempat lebih banyak daripada sebuah rata-rata dan langsung memperlihatkan apakah semua orang mendapat pengalaman serupa atau ada kelompok yang tertinggal.

7.7 Membangun Program Metrik

Kode 7.1 menghitung semua yang ada di Subbab 7.5 dan 7.6 dari tiga berkas data, ditambah taksiran penemuan masalah di Subbab 7.8. Setiap rumus menjadi fungsi kecil yang dinamai sesuai persamaannya: `adjusted_wald` adalah Persamaan (7.1), `geometric_mean` adalah Persamaan (7.2), `sus_score` adalah Persamaan (7.3), `umux_lite` adalah Persamaan (7.4), dan `discovery` adalah Persamaan (7.5) di bawah.

Kode 7.1 `usability_metrics.py`: keberhasilan tugas dengan selang *adjusted-Wald*, waktu rata-rata geometrik, SUS beserta nilainya, UMUX-Lite, dan penemuan masalah.

```python
"""Usability-test metrics: task success (adjusted-Wald CI), geometric mean
time, SUS with grade, UMUX-Lite, and the problem-discovery estimate.
Usage: python usability_metrics.py [data folder]"""
import csv
import math
import sys

Z = 1.959964
# Curved grading scale for SUS (lower bound of each grade)
GRADES = [(84.1, "A+"), (80.8, "A"), (78.9, "A-"), (77.2, "B+"), (74.1, "B"),
          (72.6, "B-"), (71.1, "C+"), (65.0, "C"), (62.7, "C-"), (51.7, "D"),
          (0.0, "F")]


def rows(path):
    with open(path, newline="", encoding="utf-8") as f:
        return list(csv.DictReader(f))


def adjusted_wald(successes, n):
    """95% adjusted-Wald interval for a completion rate."""
    p = (successes + Z * Z / 2) / (n + Z * Z)
    half = Z * math.sqrt(p * (1 - p) / (n + Z * Z))
    return max(0.0, p - half), min(1.0, p + half)


def geometric_mean(values):
    return math.exp(sum(math.log(v) for v in values) / len(values))


def sus_score(answers):
    """answers: the ten 1-5 responses in questionnaire order."""
    total = 0
    for i, a in enumerate(answers):
        total += (a - 1) if i % 2 == 0 else (5 - a)   # odd items positive
    return total * 2.5


def grade(score):
    for low, letter in GRADES:
        if score >= low:
            return letter
    return "F"
```

Kode 7.1 (Lanjutan) *usability_metrics.py: keberhasilan tugas dengan selang adjusted-Wald, waktu rata-rata geometrik, SUS beserta nilainya, UMUX-Lite, dan penemuan masalah.*

```
def umux_lite(capabilities, easy):
    """Raw UMUX-Lite (0-100) and its regression onto the SUS scale."""
    raw = (capabilities + easy - 2) * 100 / 12
    return raw, 0.65 * raw + 22.9


def t_critical(df):
    z = Z
    return (z + (z**3 + z) / (4 * df)
            + (5 * z**5 + 16 * z**3 + 3 * z) / (96 * df * df))


def mean_ci(values):
    n = len(values)
    mean = sum(values) / n
    sd = math.sqrt(sum((v - mean) ** 2 for v in values) / (n - 1))
    return mean, t_critical(n - 1) * sd / math.sqrt(n)


def discovery(p, n):
    """Expected share of problems found by n participants."""
    return 1 - (1 - p) ** n


def main(args):
    folder = args[0] if args else "../data"
    sessions = rows(folder + "/sessions.csv")
    print("Task  success  95% CI (adj. Wald)  geo-mean time  errors")
    for task in sorted({r["task"] for r in sessions}):
        rs = [r for r in sessions if r["task"] == task]
        ok = [r for r in rs if r["success"] == "1"]
        low, high = adjusted_wald(len(ok), len(rs))
        gm = geometric_mean([float(r["seconds"]) for r in ok])
        err = sum(int(r["errors"]) for r in rs) / len(rs)
        print(f"{task:<6}{len(ok)}/{len(rs)}    {low * 100:5.1f}% - "
              f"{high * 100:5.1f}%   {gm:8.1f} s   {err:6.2f}")
```

Kode 7.1 (Lanjutan) *usability_metrics.py: keberhasilan tugas dengan selang adjusted-Wald, waktu rata-rata geometrik, SUS beserta nilainya, UMUX-Lite, dan penemuan masalah.*

```
    scores = []
    print("\nSUS per participant:")
    for r in rows(folder + "/sus.csv"):
        s = sus_score([int(r[f"q{i}"]) for i in range(1, 11)])
        scores.append(s)
        print(f"  {r['participant']}: {s:5.1f}")
    mean, half = mean_ci(scores)
    print(f"SUS mean {mean:.1f} (95% CI {mean - half:.1f} to {mean + half:.1f}),"
          f" grade {grade(mean)}; the average product scores 68")

    raws, preds = [], []
    for r in rows(folder + "/umux.csv"):
        raw, pred = umux_lite(int(r["capabilities"]), int(r["easy"]))
        raws.append(raw)
        preds.append(pred)
    print(f"UMUX-Lite raw mean {sum(raws) / len(raws):.1f}, "
          f"on the SUS scale {sum(preds) / len(preds):.1f}")

    problems = rows(folder + "/problems.csv")
    n = len(scores)
    hits = sum(len(r["participants"].split()) for r in problems)
    p = hits / (len(problems) * n)
    print(f"\n{len(problems)} problems, {hits} sightings by {n} participants:"
          f" p = {p:.3f}")
    print("participants  found (p=0.31)  found (this study)")
    for k in (1, 2, 3, 5, 8, 10):
        print(f"{k:>12}  {discovery(0.31, k) * 100:13.0f}%"
              f"  {discovery(p, k) * 100:17.0f}%")


if __name__ == "__main__":
    main(sys.argv[1:])
```

Skala nilai berkurva berupa daftar batas bawah dari nilai tertinggi ke bawah; `grade` mengembalikan nilai pertama yang batas bawahnya dicapai skor. `sus_score` memakai posisi `i` dalam daftar: posisi genap (0, 2, ...) adalah butir bernomor ganjil yang positif.

Kode 7.2 adalah versi Java-nya. Program ini membaca setiap baris CSV dengan `split(",")`, yang cukup karena tidak ada kolom yang memuat koma.

Kode 7.2 `UsabilityMetrics.java`: metrik yang sama dalam Java.

```java
import java.io.IOException;
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;

/** Usability-test metrics: task success (adjusted-Wald CI), geometric mean
 *  time, SUS with grade, UMUX-Lite, and the problem-discovery estimate.
 *  Usage: java UsabilityMetrics [data folder] */
public class UsabilityMetrics {
    static final double Z = 1.959964;
    /** Curved grading scale for SUS (lower bound of each grade). */
    static final double[] LOW = {84.1, 80.8, 78.9, 77.2, 74.1, 72.6, 71.1, 65.0,
        62.7, 51.7, 0.0};
    static final String[] LETTER = {"A+", "A", "A-", "B+", "B", "B-", "C+", "C",
        "C-", "D", "F"};

    static List<String[]> rows(String path) throws IOException {
        List<String> lines = Files.readAllLines(Path.of(path));
        List<String[]> out = new ArrayList<>();
        for (String line : lines.subList(1, lines.size()))
            if (!line.isBlank()) out.add(line.split(","));
        return out;
    }

    /** 95% adjusted-Wald interval for a completion rate. */
    static double[] adjustedWald(int successes, int n) {
        double p = (successes + Z * Z / 2) / (n + Z * Z);
        double half = Z * Math.sqrt(p * (1 - p) / (n + Z * Z));
        return new double[] {Math.max(0, p - half), Math.min(1, p + half)};
    }

    static double geometricMean(List<Double> values) {
        double sum = 0;
        for (double v : values) sum += Math.log(v);
        return Math.exp(sum / values.size());
    }
```

Kode 7.2 (Lanjutan) *UsabilityMetrics.java: metrik yang sama dalam Java.*

```
    /** answers: the ten 1-5 responses in questionnaire order. */
    static double susScore(int[] answers) {
        int total = 0;
        for (int i = 0; i < answers.length; i++)
            total += (i % 2 == 0) ? answers[i] - 1 : 5 - answers[i];
        return total * 2.5;
    }

    static String grade(double score) {
        for (int i = 0; i < LOW.length; i++) if (score >= LOW[i]) return LETTER[i];
        return "F";
    }

    /** Raw UMUX-Lite (0-100) and its regression onto the SUS scale. */
    static double[] umuxLite(int capabilities, int easy) {
        double raw = (capabilities + easy - 2) * 100.0 / 12;
        return new double[] {raw, 0.65 * raw + 22.9};
    }

    static double tCritical(int df) {
        return Z + (Math.pow(Z, 3) + Z) / (4.0 * df)
            + (5 * Math.pow(Z, 5) + 16 * Math.pow(Z, 3) + 3 * Z) / (96.0 * df * df);
    }

    /** Expected share of problems found by n participants. */
    static double discovery(double p, int n) {
        return 1 - Math.pow(1 - p, n);
    }

    public static void main(String[] args) throws IOException {
        Locale.setDefault(Locale.US);
        String folder = args.length > 0 ? args[0] : "../data";
        List<String[]> sessions = rows(folder + "/sessions.csv");
        System.out.println(
            "Task  success  95% CI (adj. Wald)  geo-mean time  errors");
        TreeSet<String> tasks = new TreeSet<>();
        for (String[] r : sessions) tasks.add(r[1]);
        for (String task : tasks) {
            int n = 0, ok = 0, errors = 0;
            List<Double> times = new ArrayList<>();
            for (String[] r : sessions) {
                if (!r[1].equals(task)) continue;
                n++;
                errors += Integer.parseInt(r[4]);
                if (r[2].equals("1")) {
                    ok++;
```

Kode 7.2 (Lanjutan) *UsabilityMetrics.java: metrik yang sama dalam Java.*

```
                    times.add(Double.parseDouble(r[3]));
                }
            }
            double[] ci = adjustedWald(ok, n);
            System.out.printf("%-6s%d/%d    %5.1f%% - %5.1f%%   %8.1f s   %6.2f%n",
                task, ok, n, ci[0] * 100, ci[1] * 100, geometricMean(times),
                (double) errors / n);
        }
        System.out.println("\nSUS per participant:");
        List<Double> scores = new ArrayList<>();
        for (String[] r : rows(folder + "/sus.csv")) {
            int[] a = new int[10];
            for (int i = 0; i < 10; i++) a[i] = Integer.parseInt(r[i + 1]);
            double s = susScore(a);
            scores.add(s);
            System.out.printf("  %s: %5.1f%n", r[0], s);
        }
        int n = scores.size();
        double mean = 0, ss = 0;
        for (double s : scores) mean += s / n;
        for (double s : scores) ss += (s - mean) * (s - mean);
        double half = tCritical(n - 1) * Math.sqrt(ss / (n - 1)) / Math.sqrt(n);
        System.out.printf("SUS mean %.1f (95%% CI %.1f to %.1f), grade %s; "
            + "the average product scores 68%n", mean, mean - half, mean + half,
            grade(mean));
        double rawSum = 0, predSum = 0;
        List<String[]> umux = rows(folder + "/umux.csv");
        for (String[] r : umux) {
            double[] u = umuxLite(Integer.parseInt(r[1]), Integer.parseInt(r[2]));
            rawSum += u[0];
            predSum += u[1];
        }
        System.out.printf("UMUX-Lite raw mean %.1f, on the SUS scale %.1f%n",
            rawSum / umux.size(), predSum / umux.size());
        List<String[]> problems = rows(folder + "/problems.csv");
        int hits = 0;
        for (String[] r : problems) hits += r[2].strip().split("\\s+").length;
        double p = (double) hits / (problems.size() * n);
        System.out.printf("%n%d problems, %d sightings by %d participants: "
            + "p = %.3f%n", problems.size(), hits, n, p);
        System.out.println("participants  found (p=0.31)  found (this study)");
        for (int k : new int[] {1, 2, 3, 5, 8, 10})
            System.out.printf("%12d  %13.0f%%  %17.0f%%%n", k,
                discovery(0.31, k) * 100, discovery(p, k) * 100);
    }
}
```

Menjalankan `python3 usability_metrics.py` atau `java -cp out UsabilityMetrics` mencetak laporan lengkapnya, yang bagian-bagiannya sudah diperlihatkan di Subbab 7.5.4 dan 7.6.3:

```
Task  success  95% CI (adj. Wald)  geo-mean time  errors
T1    8/8     62.8% - 100.0%       37.4 s     0.25
T2    7/8     50.8% -  99.9%       82.9 s     1.25
T3    4/8     21.5% -  78.5%       97.9 s     1.75
T4    7/8     50.8% -  99.9%       55.8 s     0.50

SUS per participant:
  P1:  75.0
  P2:  95.0
  P3:  37.5
  P4:  77.5
  P5: 100.0
  P6:  40.0
  P7:  77.5
  P8:  60.0
SUS mean 70.3 (95% CI 51.1 to 89.5), grade C; the average product scores 68
UMUX-Lite raw mean 74.0, on the SUS scale 71.0

9 problems, 25 sightings by 8 participants: p = 0.347
participants  found (p=0.31)  found (this study)
           1             31%                 35%
           2             52%                 57%
           3             67%                 72%
           5             84%                 88%
           8             95%                 97%
          10             98%                 99%
```

7.8 Berapa Banyak Partisipan?

7.8.1 Model penemuan masalah

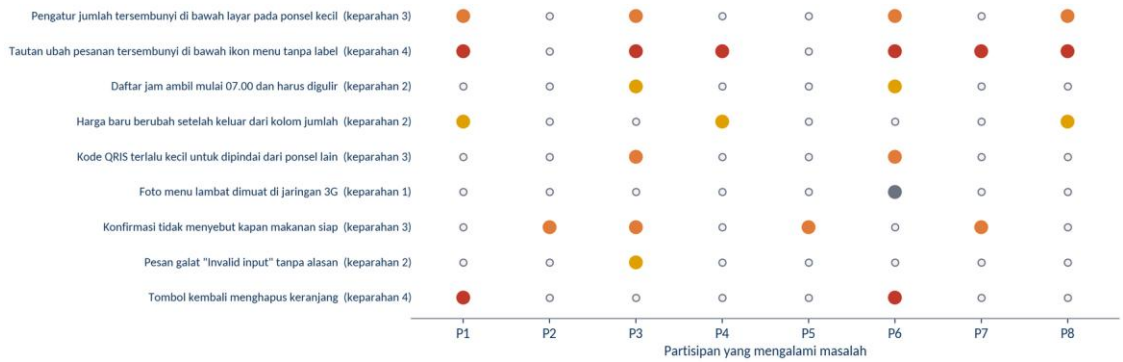
Mengapa delapan partisipan, bukan lima puluh? Nielsen dan Landauer memodelkan penemuan masalah usabilitas sebagai berikut [11]. Andaikan setiap partisipan, secara bebas, mengalami suatu masalah dengan peluang p. Maka peluang sebuah masalah dialami sekurang-kurangnya satu dari n partisipan adalah satu dikurangi peluang semuanya melewatkannya:

$$P(\text{found}) = 1 - (1 - p)^n \tag{7.5}$$

dan, dirata-rata atas semua masalah, inilah bagian masalah yang diharapkan terungkap oleh n partisipan. Pada proyek-proyek yang mereka kaji, rata-rata p sekitar 0,31, dan dengan $p = 0.31$ lima partisipan mengungkap $1 - 0.69^5 = 1 - 0.156 = 0.844$, sekitar 85% masalah. Perhitungan inilah sumber anjuran Nielsen yang terkenal bahwa hasil terbaik diperoleh dengan menguji sekitar lima pengguna, memperbaiki temuannya, lalu menguji lagi, bukan menguji sekali dengan banyak orang [12].

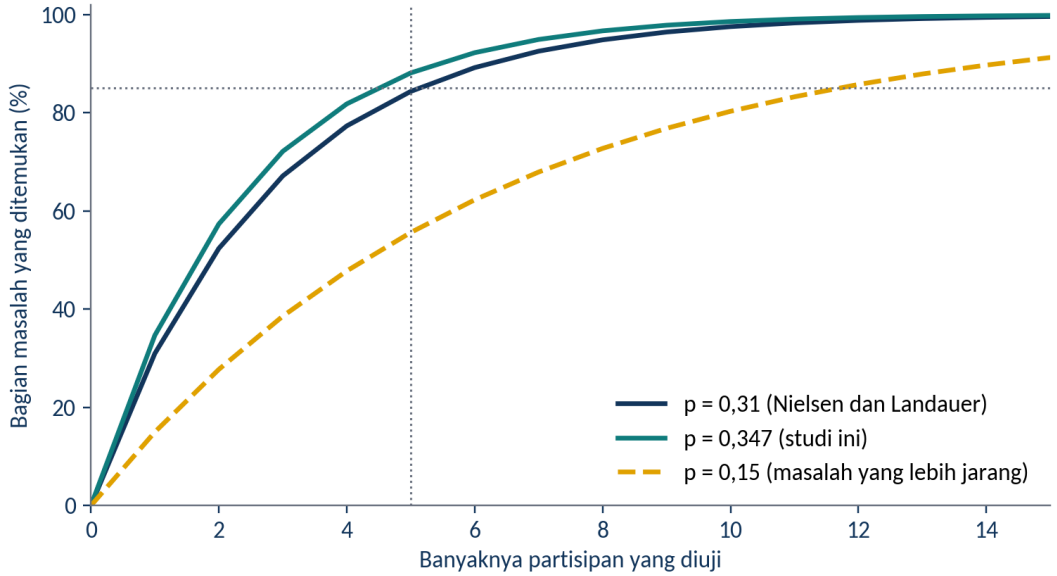
7.8.2 Menaksir p dari studi

Data uji Arif memungkinkan penaksiran p untuk rancangan ini. `data/problems.csv` mendaftar sembilan masalah yang ditemukan, dengan tingkat keparahan dan partisipan yang mengalami setiap masalah. **Gambar 7.2** menggambarnya sebagai matriks masalah-partisipan: setiap baris adalah masalah, setiap titik adalah partisipan yang mengalaminya.



Gambar 7.2 Sembilan masalah dalam uji warung dan partisipan yang mengalami masing-masing; titik berwarna yang lebih besar menandai kejadian, diwarnai menurut keparahan, dan P3 serta P6 mengalami masalah terbanyak.

Ada $9 \times 8 = 72$ sel dan 25 kejadian, sehingga $\hat{p} = 25/72 = 0.347$. **Gambar 7.3** menggambar Persamaan (7.5) untuk nilai ini, untuk 0,31 dari Nielsen dan Landauer, dan untuk rancangan yang masalahnya lebih jarang, $p = 0.15$.



Gambar 7.3 Bagian masalah yang diharapkan ditemukan seiring bertambahnya partisipan, dari $1 - (1 - p)^n$: dengan p sekitar 0,3, lima partisipan menemukan sekitar 85%; dengan masalah yang lebih jarang ($p = 0.15$), lima partisipan hanya menemukan sekitar 56%.

Model ini datang bersama tiga peringatan. Pertama, $\hat{p}$ yang ditaksir dari studi kecil bias ke atas, karena masalah yang tidak dialami siapa pun sama sekali tidak ada di dalam matriks; Lewis mengusulkan penyesuaian untuk hal ini [13]. Kedua, model ini menganggap semua partisipan serupa. Lihat matriksnya: P3 dan P6 mengalami jauh lebih banyak masalah daripada P5. Seandainya Arif menguji lima pekerja kantor yang percaya diri, p untuk *mereka* mungkin tampak tinggi dan ujinya tampak "lengkap", sementara masalah pelanggan lanjut usia tetap tersembunyi. Lima pengguna menemukan sebagian besar masalah *dari jenis pengguna yang diuji*. Ketika sebuah produk melayani kelompok-kelompok yang berbeda, ujilah setiap kelompok. Ketiga, masalah yang lebih jarang membutuhkan lebih banyak partisipan: dengan $p = 0.15$, lima partisipan hanya menemukan $1 - 0.85^5 = 0.556$ masalah. Itulah sebabnya anjurannya "lima, perbaiki, lalu uji lagi", bukan "lima lalu selesai".

7.9 Menguji Program

Kode 7.3 memeriksa rumus-rumus pada kasus yang jawabannya sudah diketahui: jawaban SUS terbaik dan terburuk menghasilkan 100 dan 0, semua tiga menghasilkan 50; batas-batas nilai; skor UMUX-Lite tertinggi; selang *adjusted-Wald* untuk 8 dari 8 yang dihitung dengan tangan di Subbab 7.5.1; rata-rata geometrik yang harus 100; 84% dari Persamaan (7.5) untuk lima partisipan; distribusi normal pada 1,96; dan uji A/B di Subbab 7.10.

Kode 7.3 `test_eval.py`: pengujian metrik usabilitas dan uji A/B.

```
"""Tests for usability_metrics.py and ab_test.py.
Run: python -m unittest test_eval.py"""
import unittest
from ab_test import phi, sample_size, two_proportion_test
from usability_metrics import (adjusted_wald, discovery, geometric_mean,
                               grade, sus_score, umux_lite)


class EvalTest(unittest.TestCase):
    def test_sus_extremes_and_neutral(self):
        self.assertEqual(sus_score([5, 1] * 5), 100.0)
        self.assertEqual(sus_score([1, 5] * 5), 0.0)
        self.assertEqual(sus_score([3] * 10), 50.0)

    def test_grade_boundaries(self):
        self.assertEqual(grade(68.0), "C")
        self.assertEqual(grade(84.1), "A+")
        self.assertEqual(grade(51.6), "F")

    def test_umux_lite_top_score(self):
        self.assertEqual(umux_lite(7, 7), (100.0, 87.9))

    def test_adjusted_wald_all_successes(self):
        low, high = adjusted_wald(8, 8)
        self.assertAlmostEqual(low, 0.628, places=3)
        self.assertEqual(high, 1.0)

    def test_geometric_mean(self):
        self.assertAlmostEqual(geometric_mean([10, 1000]), 100.0)

    def test_discovery(self):
        self.assertAlmostEqual(discovery(0.31, 5), 0.8436, places=4)

    def test_normal_cdf(self):
        self.assertAlmostEqual(phi(1.959964), 0.975, places=6)

    def test_ab_on_study_data(self):
        p1, p2, z, p, ci = two_proportion_test(157, 1210, 198, 1187)
        self.assertAlmostEqual(z, 2.554, places=3)
        self.assertLess(p, 0.05)
        self.assertEqual(sample_size(0.13, 0.16), 2162)


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

```
Ran 8 tests in 0.001s

OK
```

Kode 7.4 menjalankan pemeriksaan yang sama dalam Java.

Kode 7.4 `EvalTest.java`: pemeriksaan yang sama dalam Java.

```java
/** Tests for UsabilityMetrics and AbTest, without a test library.
 *  Run: java -cp out EvalTest */
public class EvalTest {
    static int passed = 0;

    static void check(boolean ok, String name) {
        if (!ok) throw new AssertionError("FAILED: " + name);
        passed++;
    }

    static boolean near(double a, double b, double tol) {
        return Math.abs(a - b) < tol;
    }

    public static void main(String[] args) {
        check(UsabilityMetrics.susScore(new int[] {5, 1, 5, 1, 5, 1, 5, 1, 5, 1})
            == 100.0, "SUS best");
        check(UsabilityMetrics.susScore(new int[] {3, 3, 3, 3, 3, 3, 3, 3, 3, 3})
            == 50.0, "SUS neutral");
        check(UsabilityMetrics.grade(68.0).equals("C"), "grade C");
        check(UsabilityMetrics.grade(51.6).equals("F"), "grade F");
        double[] u = UsabilityMetrics.umuxLite(7, 7);
        check(u[0] == 100.0 && near(u[1], 87.9, 1e-9), "UMUX-Lite top");
        double[] ci = UsabilityMetrics.adjustedWald(8, 8);
        check(near(ci[0], 0.628, 0.0005) && ci[1] == 1.0, "adjusted Wald");
        check(near(UsabilityMetrics.geometricMean(java.util.List.of(10.0, 1000.0)),
            100.0, 1e-9), "geometric mean");
        check(near(UsabilityMetrics.discovery(0.31, 5), 0.8436, 0.00005),
            "discovery");
        check(near(AbTest.phi(1.959964), 0.975, 1e-6), "normal CDF");
        double[] r = AbTest.twoProportionTest(157, 1210, 198, 1187);
        check(near(r[2], 2.554, 0.0005) && r[3] < 0.05, "A/B z");
        check(AbTest.sampleSize(0.13, 0.16) == 2162, "sample size");
        System.out.println(passed + " checks passed");
    }
}
```

```
11 checks passed
```

7.10 Uji A/B

7.10.1 Dari delapan orang ke ratusan

Setelah uji, Arif memperbaiki masalah-masalah yang berkategori bencana dan besar: tombol kembali mempertahankan keranjang, tombol “Ubah pesanan” berlabel muncul di konfirmasi, daftar jam ambil dimulai pukul 11.30, kode QRIS ditampilkan selebar layar dengan pilihan “simpan gambar”, dan konfirmasi menyebut “Siap pukul 12.00”. Putaran kedua dengan lima partisipan baru, termasuk dua pelanggan lanjut usia, tidak menemukan kegagalan pada T3.

Tersisa satu pertanyaan yang tidak dapat dijawab delapan atau lima partisipan: rumusan tombol pesan mana yang membuat lebih banyak pengunjung benar-benar memesan? Label lamanya *Kirim*. Sesi-sesi *think-aloud* mengisyaratkan bahwa pelanggan ragu apa yang mereka setujui dengan “Kirim”, sehingga Arif menulis versi kedua: *Pesan sekarang – ambil jam 12.00*, dengan jam yang dipilih terisi otomatis. Selisih antara dua versi semacam ini kecil, sehingga hanya dapat diukur pada banyak orang, dan orang-orang yang memakai halaman sungguhan itulah yang tepat.

> **DEFINISI.** *Uji A/B* (eksperimen terkendali daring) menugaskan setiap pengunjung secara acak ke salah satu dari dua versi rancangan, A (kontrol) atau B (perlakuan), lalu membandingkan sebuah metrik yang dipilih sebelumnya di antara kedua kelompok. Karena penugasannya

> acak, setiap selisih yang lebih besar daripada kebetulan dapat dikaitkan dengan perubahan itu [14].

Kohavi, Tang, dan Xu, berdasarkan pengalaman di perusahaan web besar, menekankan tiga aturan yang penting bahkan untuk sebuah warung: pilih metriknya *sebelum* uji dimulai (di sini, bagian pengunjung halaman menu yang memesan); tetapkan ukuran sampel atau durasinya di muka, jangan berhenti ketika hasilnya tampak bagus; dan periksa bahwa kedua kelompok berukuran sama seperti yang direncanakan, karena ketidakseimbangan yang tak terduga biasanya berarti ada *bug* dalam penugasannya [14].

7.10.2 Uji-z dua proporsi

Setelah dua minggu, `data/ab.csv` memuat hitungannya: versi A ditampilkan kepada 1.210 pengunjung dan menerima 157 pesanan; versi B kepada 1.187 pengunjung dan menerima 198. Tingkat penyelesaiannya $\hat{p}_A = 157/1210 = 0.1298$ dan $\hat{p}_B = 198/1187 = 0.1668$. Apakah selisih 3,7 poin itu nyata, atau dapat muncul karena kebetulan seandainya label tidak berpengaruh apa pun?

Uji-z dua proporsi menjawabnya. Di bawah hipotesis bahwa label tidak berpengaruh, kedua kelompok berbagi satu tingkat yang sama, yang ditaksir dengan menggabungkan: $\bar{p} = (x_A + x_B)/(n_A + n_B)$. Statistik ujinya membagi selisih yang teramati dengan galat bakunya di bawah hipotesis itu:

$$z = \frac{\hat{p}_B - \hat{p}_A}{\sqrt{\bar{p}(1-\bar{p})\left(\frac{1}{n_A} + \frac{1}{n_B}\right)}} \tag{7.6}$$

dan *nilai-p* dua sisinya adalah peluang sebuah peubah normal baku berjarak sekurang-kurangnya $|z|$ dari nol, $2(1 - \Phi(|z|))$, dengan Φ fungsi distribusi kumulatif normal baku.

Dengan tangan: $\bar{p} = 355/2397 = 0.1481$; $\bar{p}(1-\bar{p}) = 0.1481 \times 0.8519 = 0.1262$; $1/1210 + 1/1187 = 0.000826 + 0.000842 = 0.001669$; galat bakunya $\sqrt{0.1262 \times 0.001669} = \sqrt{0.0002106} = 0.01451$; $z = 0.0370/0.01451 = 2.55$. Nilai z sebesar 2,55 menyisakan sekitar 0,5% di setiap ekor, sehingga $p \approx 0.011$: seandainya label tidak berpengaruh, selisih sebesar ini hanya akan muncul sekitar sekali dalam seratus uji semacam itu. Selang kepercayaan 95% untuk selisihnya memakai galat baku tanpa penggabungan, $\sqrt{\hat{p}_A(1-\hat{p}_A)/n_A + \hat{p}_B(1-\hat{p}_B)/n_B}$, dan menghasilkan sekitar +0,9 sampai +6,6 poin.

7.10.3 Berapa pengunjung yang dibutuhkan?

Sebelum uji dimulai, kita harus tahu berapa pengunjung yang dibutuhkan untuk mendeteksi selisih terkecil yang layak ditindaklanjuti. Untuk uji pada taraf 5% dengan *kuasa* (*power*) 80% (peluang 80% untuk mendeteksi selisih sebenarnya sebesar yang dipilih), banyaknya pengunjung per versi untuk mendeteksi perubahan dari p_1 ke p_2 adalah

$$n = \frac{\left(z_{\alpha/2}\sqrt{2\bar{p}(1-\bar{p})} + z_{\beta}\sqrt{p_1(1-p_1) + p_2(1-p_2)}\right)^2}{(p_2 - p_1)^2} \tag{7.7}$$

dengan $\bar{p} = (p_1 + p_2)/2$, $z_{\alpha/2} = 1.96$, dan $z_{\beta} = 0.8416$. Penyebutnya adalah kuadrat selisih, sehingga memperkecil selisih yang ingin dideteksi menjadi separuhnya melipatempatkan sampelnya.

7.10.4 Programnya

Kode 7.5 mengimplementasikan Persamaan (7.6) dan Persamaan (7.7). Fungsi distribusi normal baku Φ dihitung dengan hampiran rasional 7.1.26 dari Abramowitz dan Stegun, yang galatnya di bawah 1.5×10^{-7} [15]; Python memiliki `math.erf`, tetapi Java tidak mempunyai fungsi galat bawaan, dan memakai hampiran yang sama di keduanya membuat keluarannya identik.

Kode 7.5 `ab_test.py`: uji-z dua proporsi, selang kepercayaan selisih, dan ukuran sampel.

```
"""Compare the completion rates of two designs (A/B test): two-proportion
z-test, confidence interval of the difference, and the sample size needed.
Usage: python ab_test.py [ab.csv]"""
import csv
import math
import sys

Z95, Z80 = 1.959964, 0.841621   # two-sided 5% level; 80% power


def phi(x):
    """Standard normal CDF (Abramowitz-Stegun 7.1.26, error < 1.5e-7)."""
    t = 1 / (1 + 0.3275911 * abs(x) / math.sqrt(2))
    poly = t * (0.254829592 + t * (-0.284496736 + t * (1.421413741
           + t * (-1.453152027 + t * 1.061405429))))
    erf = 1 - poly * math.exp(-x * x / 2)
    return 0.5 * (1 + erf) if x >= 0 else 0.5 * (1 - erf)


def two_proportion_test(x1, n1, x2, n2):
    """Return p1, p2, z, two-sided p-value, 95% CI of p2 - p1."""
    p1, p2 = x1 / n1, x2 / n2
    pooled = (x1 + x2) / (n1 + n2)
    se0 = math.sqrt(pooled * (1 - pooled) * (1 / n1 + 1 / n2))
    z = (p2 - p1) / se0
    p_value = 2 * (1 - phi(abs(z)))
    se = math.sqrt(p1 * (1 - p1) / n1 + p2 * (1 - p2) / n2)
    diff = p2 - p1
    return p1, p2, z, p_value, (diff - Z95 * se, diff + Z95 * se)


def sample_size(p1, p2):
    """Visitors per variant to detect p1 -> p2 (5% level, 80% power)."""
    pbar = (p1 + p2) / 2
    top = (Z95 * math.sqrt(2 * pbar * (1 - pbar))
           + Z80 * math.sqrt(p1 * (1 - p1) + p2 * (1 - p2)))
    return math.ceil(top * top / ((p2 - p1) ** 2))
```

Kode 7.5 (Lanjutan) *ab_test.py: uji-z dua proporsi, selang kepercayaan selisih, dan ukuran sampel.*

```
def main(args):
    path = args[0] if args else "../data/ab.csv"
    with open(path, newline="", encoding="utf-8") as f:
        a, b = list(csv.DictReader(f))
    x1, n1 = int(a["orders"]), int(a["visitors"])
    x2, n2 = int(b["orders"]), int(b["visitors"])
    p1, p2, z, p_value, (low, high) = two_proportion_test(x1, n1, x2, n2)
    print(f"A '{a['label']}': {x1}/{n1} = {p1 * 100:.2f}%")
    print(f"B '{b['label']}': {x2}/{n2} = {p2 * 100:.2f}%")
    print(f"difference B - A: {(p2 - p1) * 100:+.2f} points, "
          f"95% CI {low * 100:+.2f} to {high * 100:+.2f}")
    print(f"z = {z:.3f}, two-sided p = {p_value:.4f}")
    verdict = "significant" if p_value < 0.05 else "not significant"
    print(f"at the 5% level the difference is {verdict}")
    for target in (0.16, 0.14):
        print(f"visitors per variant to detect 13% -> {target * 100:.0f}%: "
              f"{sample_size(0.13, target)}")


if __name__ == "__main__":
    main(sys.argv[1:])
```

Kode 7.6 adalah versi Java-nya.

Kode 7.6 `AbTest.java`: analisis A/B yang sama dalam Java.

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.List;
import java.util.Locale;

/** Compare the completion rates of two designs (A/B test): two-proportion
 *  z-test, confidence interval of the difference, and the sample size needed.
 *  Usage: java AbTest [ab.csv] */
public class AbTest {
    static final double Z95 = 1.959964, Z80 = 0.841621;

    /** Standard normal CDF (Abramowitz-Stegun 7.1.26, error < 1.5e-7). */
    static double phi(double x) {
        double t = 1 / (1 + 0.3275911 * Math.abs(x) / Math.sqrt(2));
        double poly = t * (0.254829592 + t * (-0.284496736 + t * (1.421413741
            + t * (-1.453152027 + t * 1.061405429))));
        double erf = 1 - poly * Math.exp(-x * x / 2);
        return x >= 0 ? 0.5 * (1 + erf) : 0.5 * (1 - erf);
    }

    /** Returns {p1, p2, z, two-sided p-value, CI low, CI high}. */
    static double[] twoProportionTest(int x1, int n1, int x2, int n2) {
        double p1 = (double) x1 / n1, p2 = (double) x2 / n2;
        double pooled = (double) (x1 + x2) / (n1 + n2);
        double se0 = Math.sqrt(pooled * (1 - pooled) * (1.0 / n1 + 1.0 / n2));
        double z = (p2 - p1) / se0;
        double pValue = 2 * (1 - phi(Math.abs(z)));
        double se = Math.sqrt(p1 * (1 - p1) / n1 + p2 * (1 - p2) / n2);
        double diff = p2 - p1;
        return new double[] {p1, p2, z, pValue, diff - Z95 * se, diff + Z95 * se};
    }

    /** Visitors per variant to detect p1 -> p2 (5% level, 80% power). */
    static int sampleSize(double p1, double p2) {
        double pbar = (p1 + p2) / 2;
        double top = Z95 * Math.sqrt(2 * pbar * (1 - pbar))
            + Z80 * Math.sqrt(p1 * (1 - p1) + p2 * (1 - p2));
        return (int) Math.ceil(top * top / ((p2 - p1) * (p2 - p1)));
    }
```

Kode 7.6 (Lanjutan) *AbTest.java: analisis A/B yang sama dalam Java.*

```java
    public static void main(String[] args) throws IOException {
        Locale.setDefault(Locale.US);
        String path = args.length > 0 ? args[0] : "../data/ab.csv";
        List<String> lines = Files.readAllLines(Path.of(path));
        String[] a = lines.get(1).split(","), b = lines.get(2).split(",");
        int x1 = Integer.parseInt(a[3]), n1 = Integer.parseInt(a[2]);
        int x2 = Integer.parseInt(b[3]), n2 = Integer.parseInt(b[2]);
        double[] r = twoProportionTest(x1, n1, x2, n2);
        System.out.printf("A '%s': %d/%d = %.2f%%%n", a[1], x1, n1, r[0] * 100);
        System.out.printf("B '%s': %d/%d = %.2f%%%n", b[1], x2, n2, r[1] * 100);
        System.out.printf("difference B - A: %+.2f points, "
            + "95%% CI %+.2f to %+.2f%n", (r[1] - r[0]) * 100, r[4] * 100,
            r[5] * 100);
        System.out.printf("z = %.3f, two-sided p = %.4f%n", r[2], r[3]);
        System.out.println("at the 5% level the difference is "
            + (r[3] < 0.05 ? "significant" : "not significant"));
        for (double target : new double[] {0.16, 0.14})
            System.out.printf("visitors per variant to detect 13%% -> %.0f%%: %d%n",
                target * 100, sampleSize(0.13, target));
    }
}
```

`python3 ab_test.py` dan `java -cp out AbTest` mencetak:

```
A 'Kirim': 157/1210 = 12.98%
B 'Pesan sekarang - ambil jam 12.00': 198/1187 = 16.68%
difference B - A: +3.71 points, 95% CI +0.86 to +6.55
z = 2.554, two-sided p = 0.0107
at the 5% level the difference is significant
visitors per variant to detect 13% -> 16%: 2162
visitors per variant to detect 13% -> 14%: 18330
```

Label baru menang: +3,71 poin, signifikan pada taraf 5%. Namun perhatikan dua baris terakhir. Untuk mendeteksi kenaikan dari 13% ke 16% dengan kuasa 80%, uji ini membutuhkan sekitar 2.162 pengunjung per versi; dua minggu di warung menghasilkan sekitar 1.200. Hasilnya signifikan, tetapi kuasa uji ini kurang untuk efek 3 poin, dan selang kepercayaannya lebar: kenaikan sebenarnya bisa sekecil 0,9 poin atau sebesar 6,6 poin. Laporan jujur Arif berbunyi: "B sangat mungkin lebih baik; seberapa besar, kami belum tahu dengan tepat." Dan mendeteksi kenaikan 1 poin (13% ke 14%) membutuhkan 18.330 pengunjung per versi, berbulan-bulan lalu lintas warung itu, sehingga penyempurnaan kecil pada halaman ini lebih baik dinilai dengan uji usabilitas daripada dengan uji A/B.

> **PERINGATAN:** Jangan mengintip lalu berhenti. Jika Anda memeriksa nilai-p setiap hari dan berhenti saat pertama kali nilainya turun di bawah 0,05, Anda akan menyatakan "pemenang" jauh lebih sering daripada tingkat galat 5% yang dijanjikan, karena fluktuasi acak melintasi garis itu secara kebetulan di tengah jalan. Tetapkan durasi atau ukuran sampelnya di muka, lalu analisis sekali saja.

7.11 Evaluasi pada 2026: Jarak Jauh, Tanpa Pemandu, dan Berbantuan AI

Pada 2026 banyak uji usabilitas dilakukan *jarak jauh*: fasilitator dan partisipan bertemu lewat panggilan video, atau partisipan mengerjakan tugas sendirian dalam uji *tanpa pemandu* (*unmoderated*), ketika sebuah platform merekam layar dan suara lalu menyodorkan kuesioner di akhir. Uji jarak jauh menjangkau orang di luar kota dan menghemat perjalanan; uji tanpa pemandu menjangkau banyak partisipan dengan murah. Keduanya kehilangan sesuatu: fasilitator tidak dapat melihat tangan partisipan, dan tidak ada yang dapat bertanya "tadi Anda mengharapkan apa di situ?" pada saat yang tepat.

Platform riset kini menawarkan fitur AI di atasnya: transkrip otomatis, penanda momen ketika partisipan ragu, ringkasan sesi, bahkan moderator AI yang mengajukan pertanyaan lanjutan. Nasihat Subbab 5.10 berlaku: pakai AI untuk menemukan momen yang perlu ditonton, bukan untuk menggantikan menontonnya; periksa setiap ringkasan AI terhadap rekamannya; dan perlakukan rekaman wajah, suara, dan layar sebagai data pribadi menurut UU PDP, dengan persetujuan yang menyebut setiap layanan pihak ketiga yang akan memprosesnya. Ringkasan AI yang berbunyi "pengguna merasa halaman ini mudah" akan menyembunyikan P3 dan P6 persis seperti skor SUS rata-rata menyembunyikan mereka.

7.12 Menerapkannya dalam Kenyataan

Metode-metode bab ini dapat diperkecil untuk usaha terkecil dan diperbesar untuk layanan nasional. Rencana evaluasi satu halaman:

1. **Tujuan.** Formatif (menemukan masalah) atau sumatif (mengukur terhadap target)? Sebutkan keputusannya.
2. **Inspeksi lebih dahulu.** Dua atau tiga orang menjalankan evaluasi heuristik secara mandiri dan menilai keparahannya dengan **Tabel 7.1**. Perbaiki masalah kategori bencana sebelum pengguna melihatnya.
3. **Partisipan.** Lima sampai delapan per putaran dan per kelompok pengguna yang berbeda; libatkan kelompok pinggiran (pengguna lanjut usia, penglihatan lemah, pengguna pertama kali, koneksi lambat).
4. **Tugas.** Tiga sampai enam tujuan realistis dengan kata-kata pengguna, masing-masing dengan akhir yang jelas dan kriteria keberhasilan.
5. **Ukuran.** Penyelesaian dengan selang *adjusted-Wald*, waktu rata-rata geometrik, kesalahan, SUS (atau UMUX-Lite bila waktunya sempit), dan daftar masalah beserta keparahannya.
6. **Laporkan sebaran.** Gambarkan skor setiap partisipan; cari kelompok yang tertinggal.
7. **Iterasi.** Perbaiki, lalu uji lagi dengan partisipan baru.
8. **Eksperimen hanya untuk pertanyaan berlalu lintas besar.** Pakai uji A/B ketika banyak orang memakai produk sungguhan dan pertanyaannya "versi mana yang berkinerja lebih baik pada suatu metrik"; hitung ukuran sampelnya lebih dahulu dengan Persamaan (7.7).

Kios puskesmas (Bab 6) dievaluasi dengan tugas yang sama pada kios sungguhan, dengan pasien lanjut usia dilibatkan lebih banyak, dan log tampilan layar memberi waktu pengerjaan tanpa stopwatch. **Aplikasi ojek** diuji bersama pengemudi dengan ponsel mereka sendiri, di luar ruangan, karena sinar matahari dan pemakaian satu tangan adalah bagian dari konteksnya. **Desain ulang ATM** sebuah bank dievaluasi dengan tinjauan heuristik terhadap pedoman aksesibilitas, uji bersama nasabah tunanetra dan lanjut usia yang memakai colokan audio, serta tingkat penyelesaian per jenis transaksi dari log mesin.

7.13 IMK dalam Praktik: Profitina

Skor yang harus dipahami. Layar utama Profitina menampilkan skor visibilitas AI sebuah merek dan menjelaskan setiap sinyal di baliknya. Uji formatif untuk layar semacam itu memakai tugas seperti di Subbab 7.4: "Cari tahu mengapa skor Anda turun minggu ini", "Temukan satu perubahan yang paling membantu". Penyelesaian tugas dan *think-aloud* memperlihatkan apakah pemilik usaha yang nonteknis dapat berpindah dari angka ke tindakan; kuesioner kepuasan saja tidak akan memperlihatkannya.

Gerbang rilis yang dibangun di atas pengujian. Profitina hanya merilis ketika seluruh rangkaian pengujian otomatisnya, jauh lebih dari seribu pengujian termasuk penjaga UI JavaScript, berhasil semua, dan rilis bertahap baru melaporkan keberhasilan setelah versi yang terpasang menjawab pemeriksaan kesehatannya. Pengujian otomatis memeriksa bahwa produk melakukan apa yang dibangun untuk dilakukannya; pengujian itu tidak dapat memeriksa apakah orang memahaminya. Kedua jenis evaluasi saling melengkapi: pengujian otomatis menjaga agar perbaikan usabilitas kemarin tidak rusak, dan uji usabilitas menemukan masalah esok hari.

Mengukur jawaban AI bersama pengguna. Alur umpan balik Profitina memungkinkan pengguna menandai dan mengoreksi respons AI; setiap rilis model lokal diukur sebelum disajikan. Itu evaluasi sumatif yang tertanam dalam produk, dan ia mengikuti aturan bab ini: metrik tetap yang dipilih di muka, perbandingan terhadap versi yang berjalan, dan rilis hanya ketika versi baru benar-benar lebih baik. Profitina memberi saran dan penggunalah yang memutuskan; evaluasi adalah cara tim mengetahui apakah sarannya membantu.

7.14 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Menguji dengan teman dan rekan kerja.** Mereka tahu terlalu banyak dan terlalu baik hati. Rekrut pengguna sungguhan, termasuk kelompok pinggiran.
- **Tugas yang membocorkan jawabannya.** "Ketuk Edit untuk mengubah pesanan" hanya menguji kemampuan membaca. Tulis tujuan dengan kata-kata pengguna.
- **Membantu selama uji.** Setiap petunjuk mengubah kegagalan menjadi keberhasilan palsu. Diamlah; kembalikan pertanyaan; catat tugas yang dibantu sebagai kegagalan.
- **Melaporkan "8 dari 8" sebagai 100%.** Berikan selang *adjusted-Wald* (Persamaan (7.1)); dengan delapan partisipan selang itu dimulai sekitar 63%.
- **Merata-rata waktu yang menceng secara aritmetika.** Pakai rata-rata geometrik (Persamaan (7.2)) untuk sampel kecil.
- **Membaca SUS sebagai persentase.** SUS 70 kira-kira rata-rata, nilai C, bukan "70% mudah dipakai". Bandingkan dengan tolok ukur, dan gambarkan skor setiap orang.
- **Menghitung SUS tanpa membalik butir negatif.** Butir 2, 4, 6, 8, dan 10 dihitung $5 - a$, bukan $a - 1$ (Persamaan (7.3)).
- **"Lima pengguna selalu cukup."** Lima pengguna dari satu jenis menemukan sebagian besar masalah jenis itu. Uji setiap kelompok yang berbeda, dan uji lagi setelah perbaikan.
- **Mengintip uji A/B lalu berhenti lebih awal.** Tetapkan ukuran sampel di muka dengan Persamaan (7.7), lalu analisis sekali.
- **Menganggap hasil yang signifikan itu tepat.** Nilai-p di bawah 0,05 mengatakan selisihnya kemungkinan nyata, bukan seberapa besar. Laporkan selang kepercayaannya.
- **Membiarkan ringkasan AI menggantikan menonton.** Ringkasan meratakan justru pengalaman kelompok kecil yang penting.

7.15 Ringkasan Bab

- Usabilitas adalah efektivitas, efisiensi, dan kepuasan bagi pengguna, tujuan, dan konteks tertentu [1]. Evaluasi bersifat formatif (menemukan masalah) atau sumatif (mengukur), melalui inspeksi atau bersama pengguna.
- Evaluasi heuristik oleh beberapa evaluator mandiri menemukan banyak masalah dengan murah; nilai keparahan 0 sampai 4 (**Tabel 7.1**) menetapkan prioritas.
- Uji usabilitas *think-aloud* dengan tugas realistis memperlihatkan di mana dan mengapa pengguna kesulitan. Fasilitator sebagian besar diam.
- Laporkan penyelesaian dengan selang *adjusted-Wald* (Persamaan (7.1)), waktu dengan rata-rata geometrik (Persamaan (7.2)), dan kesalahan per tugas. Dalam uji warung, mengubah pesanan hanya berhasil bagi 4 dari 8 partisipan (selang 21,5% sampai 78,5%).
- SUS (Persamaan (7.3)) dan UMUX-Lite (Persamaan (7.4)) mengukur usabilitas yang dirasakan; 68 adalah produk rata-rata. Halaman warung mendapat 70,3 (C), tetapi dua partisipan lanjut usia berada di pita F (**Gambar 7.1**).
- Model penemuan masalah (Persamaan (7.5), **Gambar 7.3**) menjelaskan mengapa lima sampai delapan partisipan per putaran menemukan sebagian besar masalah kelompok yang diuji, dan mengapa kelompok yang berbeda serta masalah yang jarang membutuhkan lebih banyak.
- Uji A/B membandingkan dua versi pada banyak pengguna sungguhan; uji-z dua proporsi (Persamaan (7.6)) dan ukuran sampel yang ditetapkan di muka (Persamaan (7.7)) membuatnya dapat dipercaya. Label tombol baru menaikkan pesanan dari 12,98% menjadi 16,68% ($p = 0.011$), dengan selang lebar +0,9 sampai +6,6 poin.

7.16 Soal Latihan

1. Hitung dengan tangan skor SUS P3, yang jawabannya 3, 4, 2, 4, 3, 3, 3, 4, 2, 3, lalu periksa terhadap keluaran program.
2. Hitung dengan tangan selang *adjusted-Wald* 95% untuk 4 berhasil dari 5, dan bandingkan dengan selang sederhana $\hat{p} \pm 1.96\sqrt{\hat{p}(1-\hat{p})/n}$. Mana yang masuk akal, dan mengapa?
3. Untuk T2, waktu-waktu yang berhasil adalah 95, 62, 77, 58, 140, 81, dan 90 detik. Hitung rata-rata aritmetika dan rata-rata geometriknya, lalu jelaskan perbedaannya.
4. Dengan Persamaan (7.5), berapa partisipan yang dibutuhkan untuk menemukan, rata-rata, sekurang-kurangnya 90% masalah ketika $p = 0.31$? Dan ketika $p = 0.15$?
5. Dua teman sekelas menilai sembilan masalah di `problems.csv` secara mandiri. Jika penilaian mereka berbeda 2 atau lebih, apa yang harus mereka lakukan sebelum merata-ratanya, dan mengapa?
6. Uji A/B kedua membandingkan dua tata letak halaman menu: A 96 pesanan dari 800 pengunjung, B 108 dari 790. Tulis berkas `ab2.csv`, jalankan `ab_test` padanya, lalu tafsirkan hasilnya, termasuk baris-baris ukuran sampel.
7. Kembangkan `usability_metrics` (dalam Python atau Java) agar mencetak rata-rata SUS secara terpisah untuk partisipan berusia 60 tahun ke atas (P3 dan P6) dan untuk yang lain, lalu diskusikan apa yang dapat dan tidak dapat disimpulkan dari dua partisipan.
8. Warung ingin satu pertanyaan setelah setiap pesanan untuk memantau kepuasan dari bulan ke bulan. Apakah Anda akan memakai SUS, UMUX-Lite, atau yang lain, dan mengapa? Apa yang harus tetap dari bulan ke bulan agar angkanya dapat dibandingkan?

7.17 Penyelesaian Soal Latihan

1. Butir ganjil (1, 3, 5, 7, 9) dengan jawaban 3, 2, 3, 3, 2 menyumbang $2 + 1 + 2 + 2 + 1 = 8$. Butir genap (2, 4, 6, 8, 10) dengan jawaban 4, 4, 3, 4, 3 menyumbang $1 + 1 + 2 + 1 + 2 = 7$. Jumlahnya 15; SUS $= 2.5 \times 15 = 37.5$, sama dengan yang dicetak program.

2. *Adjusted Wald*: $\hat{p}_{\text{adj}} = (4 + 1.921)/(5 + 3.842) = 5.921/8.842 = 0.670$; setengah lebar $1.96 \times \sqrt{0.670 \times 0.330/8.842} = 1.96 \times 0.1581 = 0.310$; selang [0.360,0.980], yaitu 36,0% sampai 98,0%. Sederhana: $\hat{p} = 0.8$, setengah lebar $1.96 \times \sqrt{0.8 \times 0.2/5} = 1.96 \times 0.179 = 0.351$; selang [0.449,1.151], yang melewati 100% dan terlalu sempit di bagian bawah. Selang sederhana bergantung pada hampiran normal yang gagal untuk n kecil dan tingkat yang mendekati 0 atau 1; selang yang disesuaikan memperbaikinya, dan itulah sebabnya selang ini dianjurkan untuk uji usabilitas.

3. Rata-rata aritmetika $603/7 = 86.1$ detik. Logaritmanya: 4,554, 4,127, 4,344, 4,060, 4,942, 4,394, 4,500; jumlahnya 30,921; rata-rata 4,4173; $e^{4.4173} = 82.9$ detik, sama dengan cetakan program. Rata-rata aritmetika lebih tinggi karena satu waktu lambat 140 detik (P6, yang bingung dengan daftar jam ambil) menariknya ke atas; rata-rata geometrik kurang terpengaruh oleh satu nilai ekstrem dan merupakan taksiran waktu yang lazim yang lebih baik untuk sampel kecil yang menceng.

4. Selesaikan $1 - (1-p)^n \geq 0.90$, yaitu $n \geq \ln 0.10/\ln(1-p)$. Untuk $p = 0.31$: $-2.3026/-0.3711 = 6.2$, jadi 7 partisipan. Untuk $p = 0.15$: $-2.3026/-0.1625 = 14.2$, jadi 15 partisipan.

5. Mereka harus mendiskusikan setiap masalah semacam itu, kembali ke buktinya (siapa yang mengalaminya, seberapa sering, apakah dapat diatasi), lalu menilai ulang secara mandiri. Selisih

dua poin biasanya berarti para evaluator memahami masalahnya secara berbeda, misalnya yang satu mengira masalah itu menghalangi tugas dan yang lain tidak; merata-rata "4" dan "2" menjadi "3" akan menyembunyikan ketidaksepakatan tentang fakta alih-alih menyelesaikannya.

6. `ab2.csv`:

```
variant,label,visitors,orders
A,Menu list,800,96
B,Menu cards,790,108
```

Program mencetak tingkat 12,00% dan 13,67%, selisih +1,67 poin dengan selang 95% dari −1,62 sampai +4,96, $z = 0.996$, dan $p = 0.32$: tidak signifikan. Baris-baris ukuran sampelnya sama seperti sebelumnya (hanya bergantung pada 13% dan targetnya), dan menunjukkan bahwa sekitar 2.162 pengunjung per versi dibutuhkan bahkan untuk kenaikan 3 poin. Kesimpulan yang jujur adalah "belum ada bukti selisih dengan sampel ini", bukan "kedua tata letak berkinerja sama". Jika tata letak itu penting, jalankan uji lebih lama sampai sampel yang direncanakan tercapai, atau nilailah dengan uji usabilitas.

7. Dalam Python, setelah menghitung `scores`:

```python
older = [s for r, s in zip(rows(folder + "/sus.csv"), scores)
         if r["participant"] in ("P3", "P6")]
others = [s for r, s in zip(rows(folder + "/sus.csv"), scores)
          if r["participant"] not in ("P3", "P6")]
print(f"SUS aged 60+: {sum(older) / len(older):.1f} (n={len(older)}); "
      f"others: {sum(others) / len(others):.1f} (n={len(others)})")
```

Kode ini mencetak 38,8 untuk dua partisipan lanjut usia dan 80,8 untuk enam lainnya. Dengan dua partisipan, tidak ada selang kepercayaan yang bermakna dan tidak ada persentase pelanggan lanjut usia yang dapat ditaksir. Yang *dapat* disimpulkan bersifat kualitatif, tetapi tetap penting: rancangan ini gagal bagi kedua partisipan lanjut usia yang diuji, dengan sebab-sebab yang dapat dikenali (tautan ubah tersembunyi, daftar jam yang panjang, kode QRIS yang kecil). Langkah berikutnya yang tepat adalah satu putaran dengan lima partisipan lanjut usia atau lebih setelah perbaikan.

8. UMUX-Lite adalah pilihan yang wajar: dua butir, tervalidasi, dan dapat dibandingkan dengan SUS melalui Persamaan (7.4). SUS terlalu panjang untuk jendela sembul setelah memesan dan akan dilewati. Satu pertanyaan "seberapa mudah?" bisa dipakai, tetapi tidak dapat dibandingkan dengan tolok ukur yang sudah terbit. Agar dapat dibandingkan dari bulan ke bulan, pertahankan rumusan dan bahasa butir, skalanya (1 sampai 7), saat bertanya (tepat setelah konfirmasi pesanan), dan siapa yang ditanya (setiap pesanan ke-n, atau semua orang), serta laporkan banyaknya jawaban bersama rata-rata setiap bulan.

Daftar Pustaka

[1] International Organization for Standardization, *ISO 9241-11:2018 Ergonomics of Human-System Interaction — Part 11: Usability: Definitions and Concepts*. Geneva, Switzerland: ISO, 2018.

[2] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90)*, Seattle, WA, USA, 1990, pp. 249–256.

[3] J. Nielsen, "Severity Ratings for Usability Problems," Nielsen Norman Group, 1994, updated. [Online]. Available: https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/

[4] J. Rubin and D. Chisnell, *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, 2nd ed. Indianapolis, IN, USA: Wiley, 2008.

[5] K. A. Ericsson and H. A. Simon, *Protocol Analysis: Verbal Reports as Data*, rev. ed. Cambridge, MA, USA: MIT Press, 1993.

[6] J. Sauro and J. R. Lewis, *Quantifying the User Experience: Practical Statistics for User Research*, 2nd ed. Cambridge, MA, USA: Morgan Kaufmann, 2016.

[7] J. Brooke, "SUS: A 'quick and dirty' usability scale," in *Usability Evaluation in Industry*, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, Eds. London, U.K.: Taylor & Francis, 1996, pp. 189–194.

[8] J. Sauro, "Measuring Usability with the System Usability Scale (SUS)," MeasuringU, Feb. 2, 2011. [Online]. Available: https://measuringu.com/sus/

[9] J. R. Lewis, B. S. Utesch, and D. E. Maher, "UMUX-LITE: When there's no time for the SUS," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*, Paris, France, 2013, pp. 2099–2102.

[10] J. Sauro, "Measuring Usability: From the SUS to the UMUX-Lite," MeasuringU, Oct. 11, 2017. [Online]. Available: https://measuringu.com/umux-lite/

[11] J. Nielsen and T. K. Landauer, "A mathematical model of the finding of usability problems," in *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, 1993, pp. 206–213.

[12] J. Nielsen, "Why You Only Need to Test with 5 Users," Nielsen Norman Group, Mar. 18, 2000. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[13] J. R. Lewis, "Evaluation of procedures for adjusting problem-discovery rates estimated from small samples," *International Journal of Human-Computer Interaction*, vol. 13, no. 4, pp. 445–479, 2001.

[14] R. Kohavi, D. Tang, and Y. Xu, *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge, U.K.: Cambridge University Press, 2020.

[15] M. Abramowitz and I. A. Stegun, Eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Washington, DC, USA: National Bureau of Standards, 1964.

Bab 8

Desain Visual, Design System, dan Arsitektur Informasi

Orang tidak membaca layar; mereka memindainya, mencari kelompok, mencari hal yang paling penting, dan mencari kata yang cocok dengan keinginannya. Desain visual menentukan apa yang dilihat lebih dahulu dan apa yang saling berkaitan; arsitektur informasi menentukan di mana segala sesuatu berada dan apa namanya; *design system* mengambil keputusan-keputusan itu sekali lalu memakainya kembali di mana-mana. Di akhir bab ini Anda akan mampu memakai prinsip-prinsip Gestalt dan hierarki visual untuk membuat struktur terlihat; memilih ukuran huruf dari skala modular dan menata layar di atas grid jarak; menghitung rasio kontras WCAG 2.2 dari kode heksadesimal sebuah warna, baik dengan tangan maupun dengan program Java dan Python yang membaca berkas W3C Design Tokens dan menyarankan warna yang memenuhi syarat ketika sebuah pasangan gagal; menjelaskan apa itu *design system* dan apa arti bahasa desain tahun 2026 (Material 3 Expressive, Liquid Glass dari Apple, dan Fluent 2 dari Microsoft) bagi tim kecil; serta menjalankan dan menganalisis *card sorting* terbuka, dengan kemiripan kemunculan bersama dan pengelompokan *average linkage*, untuk mengusulkan menu yang sesuai dengan cara berpikir nasabah.

8.1 Aplikasi Sebuah Bank Perkreditan Rakyat

BPR Tani Makmur adalah bank perkreditan rakyat kecil di sebuah kabupaten di Jawa Timur. Sebagian besar nasabahnya petani, pedagang pasar kecamatan, guru, dan pensiunan pegawai negeri; banyak di antaranya berusia di atas lima puluh tahun. Lima tahun lalu bank ini meluncurkan aplikasi seluler buatan kontraktor. Aplikasi itu berfungsi dan dipakai: nasabah memeriksa saldo, mentransfer uang, dan membeli token listrik dengannya. Namun staf kantor cabang mendengar keluhan yang sama setiap minggu. “Hurufnya terlalu kecil.” “Saya tidak bisa membaca tulisan putih di atas hijau.” “Bayar angsuran di mana?” “Saya pencet sesuatu, malah masuk ke pengaturan.” Di loket, permintaan yang paling sering adalah agar petugas “tolong sekalian di HP saya”.

Layar beranda aplikasi itu menjelaskan banyak hal. Lima belas ikon tersusun dalam grid lima kali tiga, semuanya sama besar, masing-masing dengan warna cerah yang berbeda, sebagian besar tanpa label; saldo tertulis dengan huruf abu-abu kecil di bagian atas; fungsi “Transfer” dan “Bayar” bersebelahan dengan “Ganti PIN” dan “Lokasi cabang” tanpa petunjuk mana tugas sehari-hari dan mana yang jarang. Tidak ada yang salah di layar itu bila dilihat satu per satu. Bersama-sama, layar itu tidak menawarkan struktur: setiap butir berteriak sama kerasnya, sehingga tidak ada yang terdengar.

Bank itu meminta sebuah tim kecil mendesain ulang aplikasinya dalam tiga langkah yang diikuti bab ini. Pertama, membuat struktur terlihat dengan pengelompokan, hierarki, huruf, dan warna, serta memeriksa setiap pasangan warna terhadap standar aksesibilitas. Kedua, menuliskan keputusan-keputusan itu sebagai *design system* kecil berisi token dan komponen, agar kontraktor berikutnya tidak dapat membatalkannya. Ketiga, menyusun ulang menu berdasarkan cara nasabah sendiri mengelompokkan fungsi-fungsi bank, dengan *card sorting* bersama dua puluh

nasabah. Di sepanjang jalan, tim menemukan bahwa nasabah menganggap angsuran pinjaman sebagai *tagihan*, bukan bagian dari pinjaman mereka, jenis temuan yang tidak akan ditebak perancang mana pun.

8.2 Gagasannya dalam Bahasa Sederhana

Tiga gagasan mengalir di sepanjang bab ini.

> **DEFINISI.** *Desain visual* adalah penataan huruf, warna, ruang, dan gambar agar struktur dan prioritas sebuah antarmuka terlihat sekilas. *Arsitektur informasi* (*information architecture*, IA) adalah perancangan struktural lingkungan informasi bersama: bagaimana isi dan fungsi diorganisasi, diberi label, dinavigasi, dan dicari [1]. *Design system* adalah sekumpulan keputusan desain bersama yang terdokumentasi (*design token* untuk nilai seperti warna, ukuran, dan jarak; komponen seperti tombol dan kartu; serta aturan pemakaiannya) yang memungkinkan banyak orang membangun antarmuka yang konsisten.

Desain visual menjawab *apa yang saya lihat lebih dahulu, dan apa berpasangan dengan apa?* Arsitektur informasi menjawab *di mana letaknya, dan apa namanya? Design system* menjawab *bagaimana kita mengambil keputusan yang sama dengan cara yang sama setiap kali?* Layar yang indah dengan struktur yang salah akan gagal, begitu pula struktur yang baik yang tidak dapat dibaca siapa pun. Tim di bank itu membutuhkan ketiganya.

8.3 Melihat Kelompok: Gestalt dan Hierarki Visual

8.3.1 Prinsip-prinsip Gestalt

Pada awal abad kedua puluh, para psikolog Gestalt menggambarkan bagaimana orang melihat konfigurasi, bukan tanda-tanda yang terpisah: sistem penglihatan mengelompokkan unsur sebelum kita secara sadar melihatnya. Riset selama seabad sesudahnya menyempurnakan dan memperluas prinsip-prinsip ini serta menegaskan bahwa prinsip-prinsip itu mencerminkan cara persepsi benar-benar menata sebuah pemandangan [2]. Empat di antaranya melakukan sebagian besar pekerjaan dalam desain antarmuka, dan **Gambar 8.1** memperlihatkan masing-masing.



Gambar 8.1 Empat prinsip Gestalt yang dipakai dalam desain antarmuka: kedekatan, kesamaan, wilayah bersama, dan ketertutupan.

- **Kedekatan** (*proximity*). Unsur yang berdekatan terlihat sebagai satu kelompok. Jarak antara dua kelompok harus jelas lebih besar daripada jarak di dalam kelompok; jika tidak, mata melihat satu blok.
- **Kesamaan** (*similarity*). Unsur yang berbagi warna, bentuk, atau ukuran terlihat saling berkaitan. Itulah sebabnya semua hal yang dapat diketuk dalam aplikasi harus tampak serupa, dan mengapa warna acak untuk setiap ikon, seperti di layar beranda lama bank itu, merusak pengelompokan alih-alih menciptakannya.

- **Wilayah bersama** (*common region*). Unsur di dalam batas atau latar yang sama terlihat sebagai kelompok, bahkan ketika tidak berdekatan. Kartu berlatar terang mengelompokkan isinya lebih kuat daripada jarak saja [2].
- **Ketertutupan** (*closure*). Mata melengkapi bentuk yang tidak utuh. Ikon dan logo memanfaatkannya; begitu pula daftar yang butir terakhirnya terpotong di bagian bawah layar, yang memberi tahu pengguna "masih ada lagi di bawah".

Desain ulang layar beranda bank itu memakai kedekatan dan wilayah bersama sekaligus: saldo dan dua tindakan sehari-hari (transfer, bayar) berada dalam satu kartu di atas; fungsi yang lebih jarang berada di bawah dalam kartu kedua; pengaturan dan bantuan dipindahkan ke tab tersendiri. Kesamaan mengerjakan sisanya: setiap unsur yang dapat diketuk berwarna *teal* yang sama, dan tidak ada yang berwarna *teal* bila tidak dapat diketuk.

8.3.2 Hierarki visual

Hierarki visual adalah urutan mata menangkap sebuah layar. Hierarki diciptakan oleh kontras ukuran, ketebalan, warna, dan posisi: unsur yang paling besar, paling gelap, dan paling terpisah dilihat lebih dahulu. Aturannya, urutan visual harus sesuai dengan urutan kepentingan bagi tugas pengguna yang paling umum. Di layar beranda bank, tugas yang paling umum adalah memeriksa saldo, lalu mentransfer atau membayar; maka saldo menjadi teks terbesar di layar, dua tindakan utama menjadi satu-satunya tombol yang terisi warna, dan hal lain dibuat lebih tenang secara visual.

Uji cepat untuk hierarki adalah *uji menyipit*: lihat layar dengan mata setengah terpejam, atau buramkan tangkapan layarnya, lalu lihat apa yang tersisa. Di layar lama, yang tersisa lima belas gumpalan warna yang setara. Di layar baru, yang tersisa sebuah angka besar, dua tombol, dan daftar yang tenang, persis sesuai urutan kepentingannya.

8.4 Tipografi dan Tata Letak

8.4.1 Huruf yang dapat dibaca

Teks adalah sebagian besar isi antarmuka, sehingga keputusan tentang huruf lebih penting daripada pilihan visual lainnya. Empat aturan melayani nasabah bank ini:

- **Ukuran.** Teks isi sekurang-kurangnya 16 piksel CSS di ponsel, lebih besar untuk informasi utama. Mata yang lebih tua membutuhkan huruf yang lebih besar, dan pengaturan ukuran teks platform (Dynamic Type di iOS, skala huruf di Android) harus dihormati, agar nasabah yang sudah memperbesar teks di ponselnya juga mendapat teks yang besar di aplikasi.
- **Panjang baris.** Baris yang panjang sulit diikuti kembali ke awalnya; sekitar 45 sampai 75 karakter per baris nyaman untuk teks bacaan.
- **Ketebalan dan huruf kapital.** Gunakan ketebalan (biasa, tebal) alih-alih huruf kapital untuk penekanan; deretan panjang huruf kapital lebih lambat dibaca karena kata kehilangan bentuknya.
- **Angka.** Tampilkan uang dalam format yang dipakai nasabah sehari-hari: *Rp1.250.000*, dengan titik di antara ribuan, bukan *1250000* atau *IDR 1,250,000*. Bagi bank, angka adalah teks terpenting di layar.

8.4.2 Skala huruf modular

Memilih ukuran satu per satu menghasilkan banyak ukuran yang berbeda satu atau dua piksel, yang dilihat mata sebagai kekeliruan, bukan tingkatan. *Skala modular* memilih ukuran dasar b dan rasio r lalu hanya memakai ukuran-ukuran

$$s_k = b \cdot r^k, \quad k = \cdots, -1,0,1,2, \ldots \tag{8.1}$$

Dengan $b = 16$ px dan $r = 1.25$ (rasio yang dalam musik dikenal sebagai terts besar), ukurannya $16 \times 1.25^{-1} = 12.8$, 16, 20, 25, 31.25, dan 39.06 px. Setelah dibulatkan, bank memakai 13 untuk keterangan, 16 untuk teks isi, 20 untuk judul daftar, 25 untuk judul bagian, dan 39 untuk saldo. Setiap langkah 25% lebih besar daripada langkah di bawahnya, cukup untuk terlihat sebagai tingkatan yang berbeda.

8.4.3 Ruang dan grid

Ruang adalah cara kedekatan diungkapkan. Desain bank ini mengikuti praktik umum skala jarak yang dibangun di atas kelipatan 4 atau 8 piksel (4, 8, 12, 16, 24, 32, 48), sehingga setiap celah merupakan pilihan yang disengaja dari daftar pendek. Di dalam kelompok, unsur berjarak 8 px; antarkelompok, 24 px; di tepi layar, 16 px. *Grid* tata letak (di ponsel biasanya empat kolom dengan margin dan celah 16 px) menyejajarkan unsur ke dalam kolom, yang membuat layar tampak tertata dan memungkinkan mata turun menyusuri satu kolom alih-alih zig-zag.

Target sentuh juga membutuhkan ruang. WCAG 2.2 mensyaratkan target sekurang-kurangnya 24 kali 24 piksel CSS, atau jarak yang cukup di sekitar target yang lebih kecil (kriteria keberhasilan 2.5.8), dan pedoman platform menganjurkan target yang lebih besar (sekitar 44 sampai 48 poin atau piksel independen kepadatan) untuk kontrol utama [3]. Bab 10 kembali membahas sentuhan dan ibu jari.

8.5 Warna dan Kontras

8.5.1 Mengapa kontras diukur, bukan dinilai

Teks harus cukup menonjol dari latarnya agar dapat dibaca oleh orang dengan penglihatan lemah, di bawah sinar matahari terang, dan di layar murah. “Cukup” tidak dapat dinilai dengan mata di monitor bagus milik perancang, sehingga *Web Content Accessibility Guidelines* mendefinisikannya sebagai angka. WCAG 2.2, Rekomendasi W3C sejak Oktober 2023, mensyaratkan rasio kontras sekurang-kurangnya 4,5:1 untuk teks biasa dan 3:1 untuk teks besar (sekurang-kurangnya 18 poin, atau 14 poin tebal) pada tingkat AA (kriteria keberhasilan 1.4.3), serta sekurang-kurangnya 3:1 untuk batas visual kontrol, ikon, dan indikator fokus yang dibutuhkan untuk memahaminya (kriteria keberhasilan 1.4.11) [3]. Pada Oktober 2025 WCAG 2.2 juga disahkan sebagai standar internasional ISO/IEC 40500:2025, yang memperkuat kedudukannya dalam pengadaan dan hukum [4]. Penerusnya, WCAG 3.0, masih berupa *Working Draft*; W3C menerbitkan draf baru untuk ditinjau pada September 2026 [5], dan draf itu belum menjadi standar yang wajib dipenuhi siapa pun. Aturan yang berlaku saat ini, termasuk *European Accessibility Act* yang diterapkan sejak Juni 2025 dan aturan aksesibilitas Indonesia yang dibahas di Bab 11, merujuk ke WCAG 2.x.

Aturan kedua tidak bergantung pada kontras: *warna tidak boleh menjadi satu-satunya cara menyampaikan informasi* (kriteria keberhasilan 1.4.1) [3]. Aplikasi lama bank itu menampilkan transaksi gagal dengan merah dan transaksi berhasil dengan hijau, tanpa teks atau ikon;

defisiensi penglihatan warna merah-hijau cukup umum, beberapa persen laki-laki pada banyak populasi, sehingga sebagian nasabah tidak dapat membedakan keduanya. Desain baru menulis "Berhasil" atau "Gagal" di samping warnanya dan menambahkan ikon.

8.5.2 Luminans relatif dan rasio kontras

Rasio kontras membandingkan *luminans relatif* dua warna, yaitu kecerahannya sebagaimana dipersepsi, pada skala dari 0 (hitam) sampai 1 (putih). Warna layar diberikan sebagai tiga kanal sRGB, merah, hijau, dan biru, masing-masing dari 0 sampai 255 (dalam heksadesimal, `#RRGGBB`). Perhitungannya terdiri atas tiga langkah [3].

Pertama, setiap nilai kanal c_8 dibagi 255 lalu diubah dari pengodean sRGB, yang menyimpan kecerahan secara tidak linear, menjadi cahaya linear:

$$c = \frac{c_8}{255}, \qquad C = \begin{cases} c/12.92 & \text{jika } c \le 0.04045 \\ \left(\frac{c+0.055}{1.055}\right)^{2.4} & \text{jika tidak} \end{cases} \tag{8.2}$$

Kedua, ketiga kanal linear R, G, B diberi bobot sesuai kuatnya tanggapan mata terhadap masing-masing; hijau berbobot paling besar:

$$L = 0.2126\,R + 0.7152\,G + 0.0722\,B \tag{8.3}$$

Ketiga, jika L_1 adalah luminans warna yang lebih terang dan L_2 luminans warna yang lebih gelap, rasio kontrasnya adalah

$$\mathrm{CR} = \frac{L_1 + 0.05}{L_2 + 0.05} \tag{8.4}$$

Konstanta 0,05 memodelkan cahaya yang dipantulkan layar di ruangan biasa; konstanta ini menjaga rasio tetap berhingga untuk hitam murni dan menetapkan skalanya dari 1:1 (warna identik) sampai 21:1 (hitam di atas putih, $(1 + 0.05)/(0 + 0.05) = 21$).

Dihitung dengan tangan untuk teks *navy* bank `#12355B` di atas putih. Kanalnya $12_{16} = 18$, $35_{16} = 53$, dan $5\mathrm{B}_{16} = 91$. Dibagi 255: 0,0706, 0,2078, 0,3569, semuanya di atas 0,04045, sehingga masing-masing memakai rumus pangkat di Persamaan (8.2): $R = (0.1256/1.055)^{2.4} = 0.1190^{2.4} = 0.00605$, $G = 0.2493^{2.4} = 0.0356$, $B = 0.3904^{2.4} = 0.1046$. Menurut Persamaan (8.3), $L = 0.2126 \times 0.00605 + 0.7152 \times 0.0356 + 0.0722 \times 0.1046 = 0.0013 + 0.0255 + 0.0076 = 0.0343$. Putih memiliki $L = 1$. Menurut Persamaan (8.4), $\mathrm{CR} = 1.05/0.0843 = 12.46$, jauh di atas 4,5.

Untuk teks putih di atas *amber* bank `#E0A100`, luminans *amber* adalah 0,4134, dan $\mathrm{CR} = 1.05/0.4634 = 2.27$: bahkan di bawah 3:1 untuk teks besar. *Amber* adalah warna yang baik untuk sorotan atau ikon di atas *navy*; ia bukan latar untuk teks putih.

8.6 Membangun Pemeriksa Kontras

8.6.1 Warna sebagai *design token*

Tim menyimpan warna aplikasi dalam satu berkas dengan format W3C Design Tokens Community Group, yang *Design Tokens Format Module*-nya mencapai versi stabil pertama, 2025.10, pada 28 Oktober 2025 [6]. Berkas token berformat JSON; setiap token mempunyai

`$value` dan `$type`, dan token dapat dikelompokkan. Dalam format 2025.10, nilai warna berupa objek yang menyebut ruang warna dan komponennya, dengan `hex` opsional untuk warna sRGB. Awal berkas `data/tokens.json`:

```
{
  "color": {
    "$type": "color",
    "$description": "BPR Tani Makmur app palette",
    "navy": {
      "$value": {
        "colorSpace": "srgb",
        "components": [0.0706, 0.2078, 0.3569],
        "hex": "#12355B"
      }
    },
```

Berkas ini memuat palet baru ditambah dua warna aplikasi lama, `green-old` dan `sky-old`, agar yang lama dan yang baru dapat diperiksa berdampingan. Berkas kedua, `data/pairs.csv`, mendaftar pasangan yang benar-benar dipakai di layar beserta pemakaiannya: `text` untuk teks isi, `large` untuk teks besar, `ui` untuk ikon dan garis tepi.

8.6.2 Programnya dalam Python

Kode 8.1 membaca token, menghitung Persamaan (8.2), Persamaan (8.3), dan Persamaan (8.4) untuk setiap pasangan, membandingkan hasilnya dengan batas minimum untuk pemakaiannya, dan, untuk setiap pasangan yang gagal, menyarankan perbaikan: program menggelapkan warna yang lebih gelap dari keduanya dengan langkah 1% (mencampurnya dengan hitam) sampai pasangan itu mencapai batas minimum.

Kode 8.1 `contrast.py`: memeriksa pasangan warna token terhadap batas minimum kontras WCAG 2.2 dan menyarankan warna yang memenuhi syarat.

```python
"""Check color pairs of a design-token file against WCAG 2.2 contrast
minimums and suggest a darker foreground where a pair fails.
Usage: python contrast.py [tokens.json] [pairs.csv]"""
import csv
import json
import sys

# minimum ratio for AA by use: body text, large text, UI parts and icons
MINIMUM = {"text": 4.5, "large": 3.0, "ui": 3.0}


def load_tokens(path):
    """Return {token name: '#RRGGBB'} from a Design Tokens (2025.10) file."""
    with open(path, encoding="utf-8") as f:
        group = json.load(f)["color"]
    return {name: t["$value"]["hex"].upper() for name, t in group.items()
            if not name.startswith("$")}


def channel(c8):
    """sRGB channel 0-255 to linear light."""
    c = c8 / 255
    return c / 12.92 if c <= 0.04045 else ((c + 0.055) / 1.055) ** 2.4


def luminance(hex_color):
    r, g, b = (int(hex_color[i:i + 2], 16) for i in (1, 3, 5))
    return 0.2126 * channel(r) + 0.7152 * channel(g) + 0.0722 * channel(b)


def ratio(fg, bg):
    l1, l2 = sorted((luminance(fg), luminance(bg)), reverse=True)
    return (l1 + 0.05) / (l2 + 0.05)


def darken(hex_color, share):
    """Mix a color with black: share 0 keeps it, 1 gives black."""
    r, g, b = (int(hex_color[i:i + 2], 16) for i in (1, 3, 5))
    return "#{:02X}{:02X}{:02X}".format(
        *(round(c * (1 - share)) for c in (r, g, b)))
```

Kode 8.1 (Lanjutan) *contrast.py: memeriksa pasangan warna token terhadap batas minimum kontras WCAG 2.2 dan menyarankan warna yang memenuhi syarat.*

```
def fix(fg, bg, target):
    """Darken the darker color of the pair in 1% steps until the pair
    reaches the target; return (which one, new color)."""
    which = "foreground" if luminance(fg) < luminance(bg) else "background"
    dark, light = (fg, bg) if which == "foreground" else (bg, fg)
    for step in range(101):
        candidate = darken(dark, step / 100)
        if ratio(candidate, light) >= target:
            return which, candidate
    return which, None


def main(args):
    tokens = load_tokens(args[0] if args else "../data/tokens.json")
    pairs_path = args[1] if len(args) > 1 else "../data/pairs.csv"
    with open(pairs_path, newline="", encoding="utf-8") as f:
        pairs = list(csv.DictReader(f))
    print(f"{len(tokens)} color tokens, {len(pairs)} pairs checked")
    print(f"{'foreground':<11}{'background':<12}{'use':<6}{'ratio':>7}  result")
    failures = 0
    for p in pairs:
        fg, bg = tokens[p["foreground"]], tokens[p["background"]]
        r, need = ratio(fg, bg), MINIMUM[p["use"]]
        verdict = "AA pass" if r >= need else f"FAIL (needs {need}:1)"
        print(f"{p['foreground']:<11}{p['background']:<12}{p['use']:<6}"
              f"{r:>6.2f}:1  {verdict}")
        if r < need:
            failures += 1
            which, better = fix(fg, bg, need)
            old = fg if which == "foreground" else bg
            other = bg if which == "foreground" else fg
            print(f"{'':<29}try {which} {better} instead of {old}: "
                  f"{ratio(better, other):.2f}:1")
    print(f"{failures} pair(s) fail WCAG 2.2 AA")


if __name__ == "__main__":
    main(sys.argv[1:])
```

Menggelapkan warna yang *lebih gelap* dari pasangan itu adalah perbaikan paling sederhana yang selalu menggerakkan rasio ke arah yang benar: warna yang lebih terang tetap, yang lebih gelap bertambah gelap, dan selisih luminansnya membesar. Untuk teks putih di atas *amber*, itu berarti latar *amber* yang lebih gelap; untuk teks abu-abu di atas abu-abu muda, teks abu-abu yang lebih gelap. Perancang mungkin memilih perbaikan lain, misalnya rona yang berbeda, tetapi saran ini memperlihatkan seberapa jauh warna itu harus bergeser.

8.6.3 Programnya dalam Java

Kode 8.2 adalah versi Java-nya. JDK tidak mempunyai pengurai JSON, sehingga `loadTokens` menemukan setiap token dengan ekspresi reguler yang mencocokkan nama yang diikuti objek `$value` berisi kolom `hex`. Itu cukup untuk berkas datar berbentuk seperti di atas; *design system* dengan kelompok bersarang, alias, dan ruang warna lain sebaiknya dibaca dengan pustaka JSON sungguhan.

Kode 8.2 `Contrast.java`: pemeriksa yang sama dalam Java.

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/** Check color pairs of a design-token file against WCAG 2.2 contrast
 *  minimums and suggest a darker color where a pair fails.
 *  Usage: java Contrast [tokens.json] [pairs.csv] */
public class Contrast {
    static final Map<String, Double> MINIMUM =
        Map.of("text", 4.5, "large", 3.0, "ui", 3.0);

    /** Reads {name: "#RRGGBB"} from the flat color group of a Design Tokens
     *  (2025.10) file. A regular expression is enough for this file shape;
     *  use a JSON library for anything more general. */
    static Map<String, String> loadTokens(String path) throws IOException {
        String json = Files.readString(Path.of(path));
        Pattern p = Pattern.compile(
            "\"([\\w-]+)\"\\s*:\\s*\\{\\s*\"\\$value\"[^}]*?"
            + "\"hex\"\\s*:\\s*\"(#[0-9A-Fa-f]{6})\"", Pattern.DOTALL);
        Map<String, String> tokens = new LinkedHashMap<>();
        Matcher m = p.matcher(json);
        while (m.find()) tokens.put(m.group(1), m.group(2).toUpperCase());
        return tokens;
    }

    /** sRGB channel 0-255 to linear light. */
    static double channel(int c8) {
        double c = c8 / 255.0;
        return c <= 0.04045 ? c / 12.92 : Math.pow((c + 0.055) / 1.055, 2.4);
    }

    static int part(String hex, int i) {
        return Integer.parseInt(hex.substring(1 + 2 * i, 3 + 2 * i), 16);
    }

    static double luminance(String hex) {
        return 0.2126 * channel(part(hex, 0)) + 0.7152 * channel(part(hex, 1))
            + 0.0722 * channel(part(hex, 2));
    }
```

Kode 8.2 (Lanjutan) *Contrast.java: pemeriksa yang sama dalam Java.*

```
    static double ratio(String fg, String bg) {
        double a = luminance(fg), b = luminance(bg);
        return (Math.max(a, b) + 0.05) / (Math.min(a, b) + 0.05);
    }

    /** Mix a color with black: share 0 keeps it, 1 gives black. */
    static String darken(String hex, double share) {
        int[] c = new int[3];
        for (int i = 0; i < 3; i++)
            c[i] = (int) Math.round(part(hex, i) * (1 - share));
        return String.format("#%02X%02X%02X", c[0], c[1], c[2]);
    }

    /** Darken the darker color of the pair in 1% steps until the pair
     *  reaches the target; returns {which one, new color}. */
    static String[] fix(String fg, String bg, double target) {
        boolean fgDarker = luminance(fg) < luminance(bg);
        String dark = fgDarker ? fg : bg, light = fgDarker ? bg : fg;
        String which = fgDarker ? "foreground" : "background";
        for (int step = 0; step <= 100; step++) {
            String candidate = darken(dark, step / 100.0);
            if (ratio(candidate, light) >= target)
                return new String[] {which, candidate};
        }
        return new String[] {which, null};
    }

    public static void main(String[] args) throws IOException {
        Locale.setDefault(Locale.US);
        Map<String, String> tokens =
            loadTokens(args.length > 0 ? args[0] : "../data/tokens.json");
        List<String> lines = Files.readAllLines(
            Path.of(args.length > 1 ? args[1] : "../data/pairs.csv"));
        List<String> pairs = lines.subList(1, lines.size());
        System.out.println(tokens.size() + " color tokens, " + pairs.size()
            + " pairs checked");
        System.out.printf("%-11s%-12s%-6s%7s  result%n", "foreground",
            "background", "use", "ratio");
        int failures = 0;
        for (String line : pairs) {
            String[] f = line.split(",");
            String fg = tokens.get(f[0]), bg = tokens.get(f[1]);
            double r = ratio(fg, bg), need = MINIMUM.get(f[2]);
            String verdict = r >= need ? "AA pass" : "FAIL (needs " + need + ":1)";
            System.out.printf("%-11s%-12s%-6s%6.2f:1  %s%n", f[0], f[1], f[2], r,
                verdict);
```

Kode 8.2 (Lanjutan) *Contrast.java: pemeriksa yang sama dalam Java.*

```
            if (r < need) {
                failures++;
                String[] fx = fix(fg, bg, need);
                boolean isFg = fx[0].equals("foreground");
                System.out.printf("%-29stry %s %s instead of %s: %.2f:1%n", "",
                    fx[0], fx[1], isFg ? fg : bg, ratio(fx[1], isFg ? bg : fg));
            }
        }
        System.out.println(failures + " pair(s) fail WCAG 2.2 AA");
    }
}
```

8.6.4 Menjalankannya

`python3 contrast.py` (atau `java -cp out Contrast`) mencetak:

```
10 color tokens, 10 pairs checked
foreground background  use     ratio  result
navy       white       text   12.46:1  AA pass
grey       white       text    4.83:1  AA pass
grey       grey-light  text    4.39:1  FAIL (needs 4.5:1)
                              try foreground #69707D instead of #6B7280: 4.53:1
white      teal        text    5.01:1  AA pass
white      amber       large   2.27:1  FAIL (needs 3.0:1)
                              try background #C18A00 instead of #E0A100: 3.04:1
white      green-old   text    2.78:1  FAIL (needs 4.5:1)
                              try background #3A853D instead of #4CAF50: 4.56:1
white      sky-old     large   1.87:1  FAIL (needs 3.0:1)
                              try background #629CB1 instead of #7EC8E3: 3.04:1
amber      white       ui      2.27:1  FAIL (needs 3.0:1)
                              try foreground #C18A00 instead of #E0A100: 3.04:1
teal       teal-light  text    4.18:1  FAIL (needs 4.5:1)
                              try foreground #0E7676 instead of #0F7C7C: 4.52:1
red        white       text    5.44:1  AA pass
6 pair(s) fail WCAG 2.2 AA
```

Gambar 8.2 memperlihatkan pasangan-pasangan yang sama sebagai contoh warna. Tiga hasil perlu diberi catatan. Tombol putih-di-atas-hijau aplikasi lama hanya mencapai 2,78:1, yang menjelaskan keluhan "saya tidak bisa membaca tulisan putih di atas hijau"; hijau yang cukup gelap untuk lulus, `#3A853D`, disarankan. Teks isi abu-abu lulus di atas putih (4,83:1) tetapi *gagal* di atas latar kartu abu-abu muda (4,39:1), jebakan yang umum: sebuah pasangan warna diperiksa, lalu teks yang sama kelak ditempatkan di atas latar yang sedikit berbeda. Dan *teal* bank di atas *teal* muda, kombinasi yang tampak tenang dan terbaca di monitor yang bagus, gagal pada 4,18:1; perbaikannya *teal* yang sedikit lebih gelap, `#0E7676`, yang tidak akan disadari siapa pun sebagai perubahan warna.



Gambar 8.2 Sepuluh pasangan warna aplikasi bank sebagai contoh warna, dengan pemakaian, rasio kontras, dan hasil WCAG 2.2 AA.

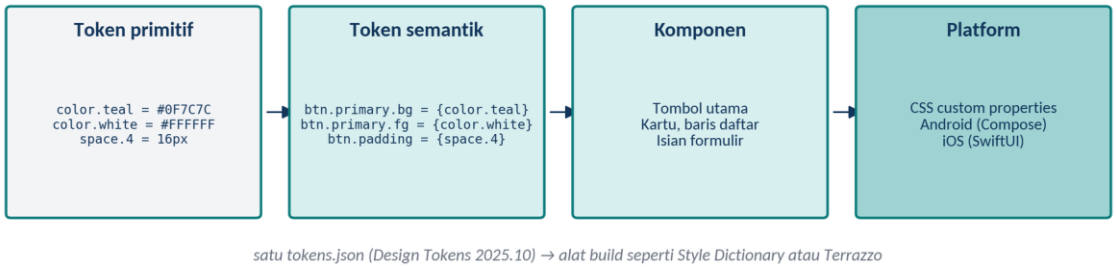
> **PERINGATAN:** Periksa pasangan yang benar-benar dipakai, termasuk teks di atas kartu, di atas gambar, dan pada keadaan nonaktif atau *hover*, bukan hanya teks utama di atas latar utama. Sebagian besar kegagalan kontras pada produk nyata berasal dari kombinasi sekunder yang tidak diperiksa siapa pun.

8.7 *Design System* dan Token

8.7.1 Dari keputusan menjadi sistem

Design system mengubah keputusan seperti di atas menjadi sesuatu yang bertahan lebih lama daripada orang-orang yang membuatnya. Biasanya ia memiliki tiga lapis. *Design token* memuat nilai mentah: warna, ukuran huruf, langkah jarak, radius sudut, durasi animasi. *Komponen* menggabungkan token menjadi bagian yang dapat dipakai ulang: tombol utama, baris daftar, kartu, isian formulir, masing-masing dengan keadaannya (normal, ditekan, nonaktif, terfokus, galat). *Pedoman* menyatakan kapan memakai komponen mana dan mengapa, dengan contoh pemakaian yang benar dan salah.

Token sendiri terbagi dalam dua tingkat, seperti diperlihatkan **Gambar 8.3**. Token *primitif* menamai nilai mentah (`color.teal = #0F7C7C`). Token *semantik* menamai tujuan dan menunjuk ke token primitif (`button.primary.background = {color.teal}`). Komponen hanya memakai token semantik. Perantaraan ini bermanfaat ketika sesuatu berubah: jika pemeriksaan kontras menunjukkan bahwa *teal* harus lebih gelap, satu token primitif berubah dan setiap komponen yang memakainya ikut berubah; jika bank menambahkan tema berkontras tinggi, token semantik menunjuk ke primitif yang berbeda dalam tema itu.



Gambar 8.3 *Design token* dalam *design system*: token primitif memuat nilai, token semantik menamai tujuan, komponen memakai token semantik, dan alat *build* mengubah satu berkas token menjadi kode untuk setiap platform.

Nilai format W3C terletak pada kemampuan satu berkas melayani setiap alat. Design Tokens Community Group mengumumkan spesifikasi 2025.10 dengan dukungan untuk tema dan banyak merek, ruang warna modern seperti Display P3 dan Oklch, alias antartoken, serta pembangkitan kode untuk iOS, Android, web, dan Flutter; implementasi rujukannya meliputi Style Dictionary, Tokens Studio, dan Terrazzo, dan alat desain termasuk Figma, Penpot, Sketch, dan Framer sedang mengadopsinya [6]. Bagi bank itu, artinya `tokens.json` yang sama yang dibaca pemeriksa kontras dapat membangkitkan CSS untuk versi web dan berkas sumber daya aplikasi Android.

8.7.2 Bahasa desain tahun 2026

Ketiga *design system* platform besar sama-sama berganti arah pada 2025, dan tim kecil yang merancang untuk platform-platform itu perlu tahu bagaimana.

Material 3 Expressive. Google memperkenalkan Material 3 Expressive pada Mei 2025 sebagai evolusi sistem Material Design untuk Android dan web, dengan bentuk yang lebih beragam, huruf yang lebih tegas, warna yang lebih kaya, dan gerak berbasis fisika. Google menggambarkannya sebagai hasil 46 studi riset dengan lebih dari 18.000 partisipan; dalam studi pelacakan mata pada sepuluh aplikasi, partisipan menemukan unsur antarmuka kunci hingga empat kali lebih cepat pada desain ekspresif, dan partisipan yang lebih tua menemukan unsur interaktif kunci secepat partisipan yang lebih muda [7]. Itu temuan Google sendiri tentang sistemnya sendiri, dan layak

dibaca dengan kesadaran itu; temuan itu juga mengingatkan bahwa "ekspresif" tidak harus berarti "kurang mudah dipakai" bila ukuran, kontras, dan hierarki dirancang dengan sengaja.

Liquid Glass. Apple mengumumkan Liquid Glass pada 9 Juni 2025 sebagai material perangkat lunak baru untuk iOS 26, iPadOS 26, macOS Tahoe 26, watchOS 26, dan tvOS 26: material tembus pandang yang memantulkan dan membiaskan apa yang ada di belakangnya serta menyesuaikan warnanya dengan isi, dipakai untuk kontrol, bilah tab, bilah samping, dan permukaan sistem [8]. Ketembusan pandang adalah keadaan ketika kontras tidak lagi dapat diperiksa sekali saja, karena latar di belakang kontrol berubah ketika pengguna menggulir. Setelah kritik tentang keterbacaan pada rilis-rilis awal, iOS 26.1 menambahkan pengaturan di Display & Brightness untuk memilih antara tampilan *Clear* yang asli dan tampilan *Tinted* yang menaikkan opasitas dan kontras, di samping pilihan aksesibilitas Reduce Transparency yang sudah ada [9]. Pelajarannya berlaku umum bagi setiap perancang: ketika teks atau ikon berada di atas latar yang berubah-ubah, jamin kontrasnya dengan lapisan penyangga atau rona, dan uji dengan isi sungguhan, bukan dengan latar polos.

Fluent 2. *Design system* Fluent 2 milik Microsoft mencakup Windows, web, serta aplikasi iOS dan Android seperti aplikasi Microsoft 365, dan menerbitkan token, komponen, serta pedoman aksesibilitasnya secara terbuka [10]. Bagi tim yang membangun aplikasi bisnis yang harus terasa serasi di samping alat-alat Microsoft, Fluent 2 adalah sekumpulan keputusan yang siap pakai.

8.7.3 Apa yang sebaiknya dilakukan tim kecil

Bank perkreditan rakyat tidak membutuhkan Material-nya sendiri. Ia membutuhkan sistem yang *kecil*: satu berkas token (mungkin dua puluh warna, sebuah skala huruf, sebuah skala jarak, tiga radius), selusin komponen beserta keadaannya, dan satu halaman aturan, semuanya dibangun di atas konvensi platform sendiri agar nasabah menemukan segala sesuatu di tempat yang sudah diajarkan ponselnya. Tim di BPR Tani Makmur mempertahankan pola navigasi dan kontrol platform, menerapkan token sendiri untuk warna, huruf, dan jarak, serta menulis pedoman satu halaman yang aturan pertamanya berbunyi "setiap pasangan teks dan ikon lulus `contrast.py` sebelum dirilis".

8.8 Arsitektur Informasi

8.8.1 Mengorganisasi, memberi label, menavigasi, mencari

Rosenfeld, Morville, dan Arango membagi arsitektur informasi menjadi empat sistem [1]:

- **Sistem organisasi** menentukan bagaimana isi dikelompokkan: menurut tugas ("bayar", "kirim"), menurut topik ("tabungan", "pinjaman"), menurut kelompok pengguna ("untuk petani", "untuk guru"), menurut abjad, atau menurut waktu. Kebanyakan aplikasi mencampur skema, dan kebanyakan kebingungan berasal dari mencampurnya tanpa menyadarinya.
- **Sistem pelabelan** menentukan nama kelompok dan butir. Label adalah janji tentang apa yang ada di baliknya; "Layanan" tidak menjanjikan sesuatu yang spesifik, "Bayar tagihan" menjanjikan sesuatu yang dapat dikenali nasabah.
- **Sistem navigasi** menentukan cara orang berpindah: tab di bagian bawah layar ponsel untuk tiga sampai lima tujuan terpenting, daftar dan kartu di dalamnya, dan jalan kembali yang jelas.
- **Sistem pencarian** memungkinkan orang yang tahu apa yang diinginkannya melewati navigasi, dan gagal ketika kata pencariannya adalah kata perancang, bukan kata pengguna.

Aplikasi lama bank itu memiliki satu organisasi yang datar (lima belas ikon), label dalam kosakata internal bank ("Mutasi", "Inquiry", "Setor Angsuran"), dan tanpa pencarian. Tim desain bisa saja menciptakan struktur baru; alih-alih, tim bertanya kepada nasabah.

8.8.2 *Card sorting* dan *tree testing*

Dalam *card sorting*, partisipan mengelompokkan kartu, masing-masing menyebut satu butir isi atau satu fungsi, ke dalam tumpukan yang masuk akal bagi mereka. Dalam *card sorting terbuka* mereka juga menamai tumpukannya; dalam *card sorting tertutup* mereka menaruh kartu ke kategori yang sudah ditentukan. *Card sorting* terbuka menemukan cara orang memikirkan suatu ranah dan kata-kata yang mereka pakai; yang tertutup memeriksa struktur yang diusulkan [11]. *Card sorting* dapat dilakukan dengan kartu kertas di kantor cabang atau dengan alat daring; lima belas sampai tiga puluh partisipan dari satu kelompok pengguna biasanya memberi gambaran yang stabil.

Uji kebalikannya adalah *tree testing*: partisipan hanya melihat teks hierarki menu yang diusulkan, tanpa desain visual apa pun, lalu ditanya di mana mereka akan mencari sesuatu ("Di mana Anda akan membayar angsuran pinjaman?"). Bagian partisipan yang mencapai tempat yang benar, dan jalur yang mereka tempuh, memperlihatkan apakah strukturnya berfungsi sebelum layar apa pun dirancang.

Tim bank menjalankan *card sorting* terbuka dengan dua puluh nasabah di dua kantor cabang, dengan lima belas fungsi aplikasi di atas kartu kertas. Setiap partisipan memilah kartu ke dalam tumpukan dan menuliskan nama pada setiap tumpukan. `data/cardsort.csv` mencatat hasilnya, satu baris per partisipan dan kartu, dengan nama yang diberikan partisipan pada tumpukannya:

```
participant,card,group
U01,Transfer to a BPR account,Transfer
U01,Transfer to another bank,Transfer
U01,Scheduled transfer,Bills
```

8.9 Menganalisis *Card Sorting*

8.9.1 Kemiripan antarkartu

Pertanyaan pertama sederhana: seberapa sering setiap pasangan kartu ditaruh di tumpukan yang sama? Jika N partisipan memilah kartu, dan n_{ij} di antaranya menaruh kartu i dan j di tumpukan yang sama, *kemiripan kemunculan bersama* pasangan itu adalah

$$S_{ij} = \frac{n_{ij}}{N} \tag{8.5}$$

sebuah bilangan dari 0 (tidak pernah bersama) sampai 1 (selalu bersama). Nama tumpukan tidak berpengaruh pada ukuran ini, yang penting hanya apakah dua kartu berbagi tumpukan; "Pay" milik seorang partisipan dan "Bills" milik partisipan lain sama-sama hanya berarti "tumpukan yang sama dengan kartu-kartu tagihan lainnya".

8.9.2 Pengelompokan *average linkage*

Kemiripan lima belas kartu membentuk tabel 105 pasangan, terlalu banyak untuk dibaca. *Pengelompokan aglomeratif hierarkis* meringkasnya. Metode ini dimulai dengan setiap kartu dalam kelompoknya sendiri dan berulang kali menggabungkan dua kelompok yang paling mirip, sampai satu kelompok memuat segalanya. Kemiripan dua kelompok A dan B, masing-masing

sekumpulan kartu, didefinisikan dengan *average linkage* sebagai rata-rata kemiripan semua pasangan kartu, satu dari masing-masing kelompok:

$$S(A,B) = \frac{1}{|A|\,|B|}\sum_{i\in A}\sum_{j\in B} S_{ij} \tag{8.6}$$

dengan $|A|$ banyaknya kartu di A. *Average linkage* adalah jalan tengah antara menggabungkan kelompok segera setelah *ada* dua anggota yang mirip (yang merangkai kartu-kartu yang tidak berkaitan) dan hanya bila *semua* anggota mirip (yang sangat ketat). Catatan penggabungannya, bila digambar sebagai pohon, disebut *dendrogram*; memotong pohon itu pada kemiripan tertentu menghasilkan kelompok-kelompok yang, rata-rata, disepakati sekurang-kurangnya oleh bagian partisipan sebesar itu.

8.9.3 Programnya

Kode 8.3 menghitung Persamaan (8.5) untuk semua pasangan, mengelompokkan dengan Persamaan (8.6), dan memotong hasilnya pada tingkat yang dipilih (0,5 secara bawaan: kelompok yang, rata-rata, disepakati sekurang-kurangnya separuh partisipan).

Kode 8.3 `cardsort.py`: kemiripan kemunculan bersama dan pengelompokan *average linkage* untuk *card sorting* terbuka.

```
"""Analyze an open card sort: how often each pair of cards was grouped
together, and average-linkage clustering into a proposed menu.
Usage: python cardsort.py [cardsort.csv] [cut]"""
import csv
import sys


def load(path):
    """Return (cards in first-seen order, {participant: {card: group}})."""
    cards, sorts = [], {}
    with open(path, newline="", encoding="utf-8") as f:
        for r in csv.DictReader(f):
            if r["card"] not in cards:
                cards.append(r["card"])
            sorts.setdefault(r["participant"], {})[r["card"]] = r["group"]
    return cards, sorts


def similarity(cards, sorts):
    """sim[i][j] = share of participants who put cards i and j together."""
    n = len(cards)
    sim = [[0.0] * n for _ in range(n)]
    for groups in sorts.values():
        for i in range(n):
            for j in range(n):
                if groups[cards[i]] == groups[cards[j]]:
                    sim[i][j] += 1 / len(sorts)
    return sim


def link(a, b, sim):
    """Average linkage: mean similarity over all pairs across two clusters."""
    total = 0.0
    for i in a:
        for j in b:
            total += sim[i][j]
    return total / (len(a) * len(b))
```

Kode 8.3 (Lanjutan) *cardsort.py: kemiripan kemunculan bersama dan pengelompokan average linkage untuk card sorting terbuka.*

```
def cluster(sim):
    """Merge the two most similar clusters until one is left.
    Returns the list of merges (similarity, cluster a, cluster b)."""
    clusters = [[i] for i in range(len(sim))]
    merges = []
    while len(clusters) > 1:
        best = None
        for x in range(len(clusters)):
            for y in range(x + 1, len(clusters)):
                s = link(clusters[x], clusters[y], sim)
                if best is None or s > best[0] + 1e-12:
                    best = (s, x, y)
        s, x, y = best
        merges.append((s, clusters[x], clusters[y]))
        clusters = ([c for k, c in enumerate(clusters) if k not in (x, y)]
                    + [clusters[x] + clusters[y]])
    return merges


def cut(n, merges, level):
    """Groups formed by all merges at or above the similarity level."""
    groups = [[i] for i in range(n)]
    for s, a, b in merges:
        if s >= level:
            ga = next(g for g in groups if a[0] in g)
            gb = next(g for g in groups if b[0] in g)
            groups.remove(gb)
            ga.extend(gb)
    return groups


def main(args):
    path = args[0] if args else "../data/cardsort.csv"
    level = float(args[1]) if len(args) > 1 else 0.5
    cards, sorts = load(path)
    sim = similarity(cards, sorts)
    print(f"{len(sorts)} participants sorted {len(cards)} cards")
    pairs = sorted(((sim[i][j], cards[i], cards[j]) for i in range(len(cards))
                    for j in range(i + 1, len(cards))), key=lambda t: -t[0])
    print("Most often together:")
    for s, a, b in pairs[:5]:
        print(f"  {s * 100:3.0f}%  {a} + {b}")
    print("Merges (average linkage), each creating a new group G1, G2, ...:")
    names = {}                          # frozenset of card indexes -> name
```

Kode 8.3 (Lanjutan) *cardsort.py: kemiripan kemunculan bersama dan pengelompokan average linkage untuk card sorting terbuka.*

```
    def label(members):
        return names.get(frozenset(members), cards[members[0]])

    for k, (s, a, b) in enumerate(cluster(sim), 1):
        print(f"  {s:.2f}  G{k} = {label(a)} + {label(b)}")
        names[frozenset(a + b)] = f"G{k}"
    print(f"Proposed groups at similarity >= {level}:")
    for g in cut(len(cards), cluster(sim), level):
        print("  - " + "; ".join(cards[i] for i in g))


if __name__ == "__main__":
    main(sys.argv[1:])
```

Pengelompokan ini ditulis agar jelas, bukan agar cepat: pada setiap langkah program membandingkan setiap pasangan kelompok dengan `link`, yang cukup untuk puluhan kartu dalam *card sorting* tetapi akan terlalu lambat untuk ribuan butir. Ketika dua pasangan sama miripnya, yang pertama ditemukan menang; toleransi kecil `1e-12` memastikan bahwa versi Python dan Java, yang menjumlahkan bilangan yang sama dalam urutan yang sama, mengambil pilihan yang sama.

Kode 8.4 adalah versi Java-nya. Sebuah `record` menyimpan setiap penggabungan; peta dari himpunan indeks kartu ke nama seperti `G3` memungkinkan laporan merujuk kelompok-kelompok sebelumnya.

Kode 8.4 `CardSort.java`: analisis yang sama dalam Java.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
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;

/** Analyze an open card sort: how often each pair of cards was grouped
 *  together, and average-linkage clustering into a proposed menu.
 *  Usage: java CardSort [cardsort.csv] [cut] */
public class CardSort {
    List<String> cards = new ArrayList<>();
    Map<String, Map<String, String>> sorts = new LinkedHashMap<>();
    double[][] sim;

    record Merge(double sim, List<Integer> a, List<Integer> b) {}

    static CardSort load(String path) throws IOException {
        CardSort cs = new CardSort();
        List<String> lines = Files.readAllLines(Path.of(path));
        for (String line : lines.subList(1, lines.size())) {
            if (line.isBlank()) continue;
            String[] f = line.split(",");
            if (!cs.cards.contains(f[1])) cs.cards.add(f[1]);
            cs.sorts.computeIfAbsent(f[0], k -> new HashMap<>()).put(f[1], f[2]);
        }
        int n = cs.cards.size();
        cs.sim = new double[n][n];
        for (Map<String, String> groups : cs.sorts.values())
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    if (groups.get(cs.cards.get(i))
                            .equals(groups.get(cs.cards.get(j))))
                        cs.sim[i][j] += 1.0 / cs.sorts.size();
        return cs;
    }
```

Kode 8.4 (Lanjutan) *CardSort.java: analisis yang sama dalam Java.*

```
    /** Average linkage: mean similarity over all pairs across two clusters. */
    double link(List<Integer> a, List<Integer> b) {
        double total = 0;
        for (int i : a) for (int j : b) total += sim[i][j];
        return total / (a.size() * b.size());
    }

    List<Merge> cluster() {
        List<List<Integer>> clusters = new ArrayList<>();
        for (int i = 0; i < cards.size(); i++) clusters.add(List.of(i));
        List<Merge> merges = new ArrayList<>();
        while (clusters.size() > 1) {
            double best = -1;
            int bx = -1, by = -1;
            for (int x = 0; x < clusters.size(); x++)
                for (int y = x + 1; y < clusters.size(); y++) {
                    double s = link(clusters.get(x), clusters.get(y));
                    if (bx < 0 || s > best + 1e-12) {
                        best = s;
                        bx = x;
                        by = y;
                    }
                }
            List<Integer> a = clusters.get(bx), b = clusters.get(by);
            merges.add(new Merge(best, a, b));
            List<List<Integer>> next = new ArrayList<>();
            for (int k = 0; k < clusters.size(); k++)
                if (k != bx && k != by) next.add(clusters.get(k));
            List<Integer> joined = new ArrayList<>(a);
            joined.addAll(b);
            next.add(joined);
            clusters = next;
        }
        return merges;
    }
```

Kode 8.4 (Lanjutan) *CardSort.java: analisis yang sama dalam Java.*

```
    /** Groups formed by all merges at or above the similarity level. */
    List<List<Integer>> cut(List<Merge> merges, double level) {
        List<List<Integer>> groups = new ArrayList<>();
        for (int i = 0; i < cards.size(); i++)
            groups.add(new ArrayList<>(List.of(i)));
        for (Merge m : merges) {
            if (m.sim() < level) continue;
            List<Integer> ga = null, gb = null;
            for (List<Integer> g : groups) {
                if (g.contains(m.a().get(0))) ga = g;
                if (g.contains(m.b().get(0))) gb = g;
            }
            groups.remove(gb);
            ga.addAll(gb);
        }
        return groups;
    }

    String names(List<Integer> idx) {
        List<String> out = new ArrayList<>();
        for (int i : idx) out.add(cards.get(i));
        return String.join("; ", out);
    }

    public static void main(String[] args) throws IOException {
        Locale.setDefault(Locale.US);
        CardSort cs = load(args.length > 0 ? args[0] : "../data/cardsort.csv");
        double level = args.length > 1 ? Double.parseDouble(args[1]) : 0.5;
        int n = cs.cards.size();
        System.out.println(cs.sorts.size() + " participants sorted " + n
            + " cards");
        List<double[]> pairs = new ArrayList<>();
        for (int i = 0; i < n; i++)
            for (int j = i + 1; j < n; j++)
                pairs.add(new double[] {cs.sim[i][j], i, j});
        pairs.sort((p, q) -> Double.compare(q[0], p[0]));   // stable sort
        System.out.println("Most often together:");
        for (double[] p : pairs.subList(0, 5))
            System.out.printf("  %3.0f%%  %s + %s%n", p[0] * 100,
                cs.cards.get((int) p[1]), cs.cards.get((int) p[2]));
        System.out.println(
            "Merges (average linkage), each creating a new group G1, G2, ...:");
        Map<Set<Integer>, String> labels = new HashMap<>();
        List<Merge> merges = cs.cluster();
        for (int k = 0; k < merges.size(); k++) {
            Merge m = merges.get(k);
```

Kode 8.4 (Lanjutan) *CardSort.java: analisis yang sama dalam Java.*

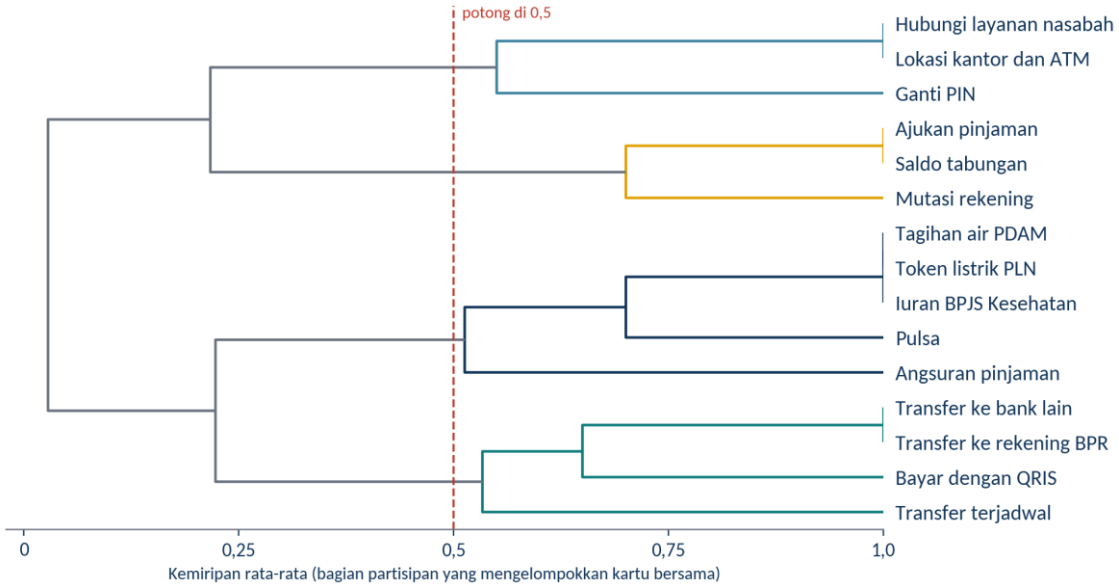
```
            String la = labels.getOrDefault(new HashSet<>(m.a()),
                cs.cards.get(m.a().get(0)));
            String lb = labels.getOrDefault(new HashSet<>(m.b()),
                cs.cards.get(m.b().get(0)));
            System.out.printf("  %.2f  G%d = %s + %s%n", m.sim(), k + 1, la, lb);
            Set<Integer> joined = new HashSet<>(m.a());
            joined.addAll(m.b());
            labels.put(joined, "G" + (k + 1));
        }
        System.out.println("Proposed groups at similarity >= " + level + ":");
        for (List<Integer> g : cs.cut(merges, level))
            System.out.println("  - " + cs.names(g));
    }
}
```

8.9.4 Membaca hasilnya

`python3 cardsort.py` (atau `java -cp out CardSort`) mencetak:

```
20 participants sorted 15 cards
Most often together:
  100%  Transfer to a BPR account + Transfer to another bank
  100%  PLN electricity token + PDAM water bill
  100%  PLN electricity token + BPJS Health contribution
  100%  PDAM water bill + BPJS Health contribution
  100%  Savings balance + Apply for a loan
Merges (average linkage), each creating a new group G1, G2, ...:
  1.00  G1 = Transfer to a BPR account + Transfer to another bank
  1.00  G2 = PLN electricity token + PDAM water bill
  1.00  G3 = BPJS Health contribution + G2
  1.00  G4 = Savings balance + Apply for a loan
  1.00  G5 = Branch and ATM locations + Contact customer service
  0.70  G6 = Phone credit + G3
  0.70  G7 = Account statement + G4
  0.65  G8 = Pay by QRIS + G1
  0.55  G9 = Change PIN + G5
  0.53  G10 = Scheduled transfer + G8
  0.51  G11 = Loan installment + G6
  0.22  G12 = G10 + G11
  0.22  G13 = G7 + G9
  0.03  G14 = G12 + G13
Proposed groups at similarity >= 0.5:
  - Scheduled transfer; Pay by QRIS; Transfer to a BPR account; Transfer to another bank
  - Account statement; Savings balance; Apply for a loan
  - Loan installment; Phone credit; BPJS Health contribution; PLN electricity token; PDAM
water bill
  - Change PIN; Branch and ATM locations; Contact customer service
```

Gambar 8.4 menggambar penggabungan itu sebagai dendrogram: makin ke kanan dua kartu bergabung, makin banyak partisipan yang menaruhnya bersama. Memotong pada 0,5 menghasilkan empat kelompok.



Gambar 8.4 Dendrogram *card sorting* terbuka (20 partisipan, 15 kartu, *average linkage*); memotong pada kemiripan 0,5 menghasilkan empat kelompok, dan angsuran pinjaman bergabung dengan tagihan.

Tiga kelompok sesuai dugaan seorang bankir: transfer (termasuk pembayaran QRIS dan transfer terjadwal), tabungan dan pinjaman (saldo, mutasi, pengajuan pinjaman), dan bantuan (PIN, lokasi, layanan nasabah). Kelompok keempat memuat kejutannya: *Angsuran pinjaman* bergabung

dengan tagihan (listrik, air, BPJS, pulsa) pada kemiripan 0,51, bukan dengan kelompok pinjaman. Pengamatan lebih dekat pada kemiripannya memperlihatkan pembelahannya: angsuran dikelompokkan bersama tagihan air oleh 55% partisipan dan bersama saldo tabungan oleh 45%. Bagi banyak nasabah, angsuran adalah sesuatu yang Anda *bayar* setiap bulan, seperti listrik; pinjaman adalah sesuatu yang Anda *punyai*. Aplikasi lama bank menaruh "Setor Angsuran" di bawah "Pinjaman", tempat lebih dari separuh nasabah tidak akan mencarinya.

Tim menarik tiga keputusan desain dari *card sorting* ini:

1. Navigasi bawah mendapat empat tab yang sesuai dengan empat kelompok, dengan kata-kata partisipan sendiri sebagai label: *Kirim uang*, *Bayar*, *Tabungan & pinjaman*, dan *Bantuan*. Nama-nama tumpukan memperlihatkan kata-kata yang dipakai nasabah; "Bayar tagihan" dan "Bills" adalah nama yang paling sering untuk tumpukan tagihan, dan *Bayar* yang lebih pendek dipilih agar muat di tab.
2. *Angsuran pinjaman* muncul di bawah *Bayar*, dan juga sebagai tautan di halaman pinjaman itu sendiri, karena 45% nasabah akan mencarinya di sana. Mencantumkan silang butir yang terbelah di antara dua kelompok adalah obat IA yang baku.
3. Label kartu yang membuat partisipan ragu selama pemilahan diganti: "Mutasi" menjadi "Riwayat transaksi" di antarmuka, sementara "mutasi" dipertahankan sebagai kata kunci pencarian karena nasabah lama mengenalnya.

Tree testing dengan dua puluh nasabah baru kemudian memeriksa strukturnya: untuk tugas "bayar angsuran pinjaman Anda", 18 dari 20 menemukannya pada percobaan pertama, sebagian besar di bawah *Bayar*.

> **WAWASAN KUNCI:** *Card sorting* tidak merancang menu; ia memperlihatkan cara nasabah berpikir. Kelompok-kelompoknya mengusulkan pengelompokan, nama tumpukannya mengusulkan label, dan butir yang terbelah di antara kelompok memperlihatkan di mana tautan silang dibutuhkan. Perancang tetap yang memutuskan, dan *tree testing* memeriksa keputusannya.

8.10 Menguji Program

Kode 8.5 memeriksa rumus kontras pada batas-batasnya (21:1 untuk hitam di atas putih dalam urutan mana pun, 1:1 untuk warna identik), pembaca token dan rasio *navy* di atas putih yang dihitung dengan tangan, bahwa perbaikan yang disarankan mencapai targetnya, serta analisis *card sorting*: empat belas penggabungan untuk lima belas kartu, empat kelompok pada 0,5 dengan angsuran di antara tagihan, dan dua kelompok pada 0,2.

Kode 8.5 `test_design.py`: pengujian pemeriksa kontras dan analisis *card sorting*.

```
"""Tests for contrast.py and cardsort.py.
Run: python -m unittest test_design.py"""
import unittest
import cardsort
from contrast import darken, fix, load_tokens, ratio


class DesignTest(unittest.TestCase):
    def test_ratio_limits(self):
        self.assertAlmostEqual(ratio("#000000", "#FFFFFF"), 21.0)
        self.assertAlmostEqual(ratio("#777777", "#777777"), 1.0)
        self.assertAlmostEqual(ratio("#FFFFFF", "#000000"), 21.0)

    def test_tokens_file(self):
        tokens = load_tokens("../data/tokens.json")
        self.assertEqual(tokens["navy"], "#12355B")
        self.assertAlmostEqual(ratio(tokens["navy"], tokens["white"]), 12.46,
                               places=2)

    def test_fix_reaches_target(self):
        which, color = fix("#FFFFFF", "#E0A100", 3.0)
        self.assertEqual(which, "background")
        self.assertGreaterEqual(ratio("#FFFFFF", color), 3.0)
        self.assertEqual(darken("#FFFFFF", 1.0), "#000000")

    def test_card_sort_groups(self):
        cards, sorts = cardsort.load("../data/cardsort.csv")
        sim = cardsort.similarity(cards, sorts)
        merges = cardsort.cluster(sim)
        self.assertEqual(len(merges), len(cards) - 1)
        groups = cardsort.cut(len(cards), merges, 0.5)
        self.assertEqual(len(groups), 4)
        bills = next(g for g in groups
                     if cards.index("PDAM water bill") in g)
        self.assertIn(cards.index("Loan installment"), bills)
        self.assertEqual(len(cardsort.cut(len(cards), merges, 0.2)), 2)


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

```
Ran 4 tests in 0.002s

OK
```

Kode 8.6 melakukan pemeriksaan yang sama dalam Java.

Kode 8.6 `DesignTest.java`: pemeriksaan yang sama dalam Java.

```java
import java.io.IOException;
import java.util.List;
import java.util.Map;

/** Tests for Contrast and CardSort, without a test library.
 *  Run: java -cp out DesignTest */
public class DesignTest {
    static int passed = 0;

    static void check(boolean ok, String name) {
        if (!ok) throw new AssertionError("FAILED: " + name);
        passed++;
    }

    public static void main(String[] args) throws IOException {
        check(Math.abs(Contrast.ratio("#000000", "#FFFFFF") - 21.0) < 1e-9, "21:1");
        check(Math.abs(Contrast.ratio("#777777", "#777777") - 1.0) < 1e-9, "1:1");
        Map<String, String> tokens = Contrast.loadTokens("../data/tokens.json");
        check(tokens.get("navy").equals("#12355B"), "token read");
        check(Math.abs(Contrast.ratio(tokens.get("navy"), tokens.get("white"))
            - 12.46) < 0.005, "navy on white");
        String[] fx = Contrast.fix("#FFFFFF", "#E0A100", 3.0);
        check(fx[0].equals("background")
            && Contrast.ratio("#FFFFFF", fx[1]) >= 3.0, "fix");
        check(Contrast.darken("#FFFFFF", 1.0).equals("#000000"), "darken");
        CardSort cs = CardSort.load("../data/cardsort.csv");
        List<CardSort.Merge> merges = cs.cluster();
        check(merges.size() == cs.cards.size() - 1, "merge count");
        List<List<Integer>> groups = cs.cut(merges, 0.5);
        check(groups.size() == 4, "four groups");
        int bill = cs.cards.indexOf("PDAM water bill");
        int loan = cs.cards.indexOf("Loan installment");
        boolean together = false;
        for (List<Integer> g : groups)
            if (g.contains(bill) && g.contains(loan)) together = true;
        check(together, "installment with bills");
        check(cs.cut(merges, 0.2).size() == 2, "two groups at 0.2");
        System.out.println(passed + " checks passed");
    }
}
```

```
10 checks passed
```

8.11 Menerapkannya dalam Kenyataan

Tiga langkah yang sama berlaku untuk layanan kecil apa pun.

Untuk desain visual: daftarkan tiga hal terpenting di setiap layar dan buat urutan visualnya sesuai; kelompokkan dengan kedekatan dan wilayah bersama; pakai satu warna untuk “dapat diketuk”; pilih ukuran huruf dari skala modular (Persamaan (8.1)) dan jarak dari skala 4 atau 8 piksel; lakukan uji menyipit.

Untuk warna: simpan setiap warna dalam berkas token; daftarkan setiap pasangan yang benar-benar dipakai; jalankan `contrast.py` pada setiap perubahan; jangan pernah memakai warna saja untuk makna; uji latar tembus pandang atau bergambar dengan isi sungguhan.

Untuk *design system*: mulailah kecil (token, selusin komponen, satu halaman aturan), bangun di atas konvensi platform, dan beri sistem itu seorang pemilik, karena sistem tanpa pemilik akan membusuk menjadi folder berisi berkas-berkas lama.

Untuk arsitektur informasi: jalankan *card sorting* terbuka dengan 15 sampai 30 pengguna sungguhan; analisis dengan Persamaan (8.5) dan Persamaan (8.6); ambil label dari nama

tumpukan partisipan; cantumkan silang butir yang terbelah; periksa hasilnya dengan *tree testing* sebelum merancang layar.

Tiga contoh. **Situs web puskesmas** dapat memilah layanannya bersama pasien di ruang tunggu; pasien kemungkinan besar akan mengelompokkan "imunisasi", "posyandu", dan "KIA" bersama, apa pun bunyi bagan organisasinya. **Papan menu dan halaman pemesanan warung** (Bab 7) dapat memakai prinsip Gestalt secara langsung: lauk dikelompokkan menurut jenis dalam wilayah bersama, harga disejajarkan dalam satu kolom, menu spesial hari itu dengan satu-satunya warna aksen. **Pengaturan aplikasi ojek** dapat diuji dengan *tree testing* bersama pengemudi untuk menemukan di mana mereka mengharapkan "ganti pelat kendaraan" dan "berhenti menerima pesanan".

8.12 IMK dalam Praktik: Profitina

Produk Profitina menggambarkan masing-masing dari tiga gagasan bab ini.

Dasbor sebagai kokpit. Dasbor Profitina mengumpulkan, dalam satu permintaan, semua hal yang membutuhkan perhatian pengguna: notifikasi balasan, suka, *repost*, pesan langsung, sematan, status umpan balik, dan pembelian PROFIT, kanal yang diikuti, peringkat, dan aktivitas, dengan setiap butir dapat diklik menuju bagiannya sendiri. Itu keputusan arsitektur informasi, mengorganisasi menurut *apa yang membutuhkan saya sekarang* alih-alih *modul mana yang menghasilkannya*, sekaligus masalah desain visual: banyak jenis butir harus dapat dipindai sekaligus, sehingga pengelompokan, tata letak butir yang konsisten, dan hierarki yang jelas antara yang belum dan sudah dibaca lebih penting daripada hiasan.

Penghitung yang bermakna. Kanal menampilkan penghitung "N baru", dan penghitung itu hanya menghitung kiriman tingkat atas, bukan setiap balasan. Pilihan tentang *apa* yang dihitung adalah keputusan pelabelan: angka di samping kanal menjanjikan "ada sebanyak ini hal baru untuk dilihat", dan menghitung balasan akan membesar-besarkan janji itu dan melatih pengguna untuk mengabaikannya.

Arsitektur untuk mesin sekaligus manusia. Situs pemasaran Profitina dwibahasa (Indonesia dan Inggris) dan menerbitkan `/llms.txt`, peta situs berupa teks polos untuk mesin jawaban AI, di samping data terstruktur (schema.org) yang dibantu produk ini untuk ditambahkan pelanggannya. Arsitektur informasi pada 2026 memiliki dua khalayak: orang yang menavigasi situs dan sistem AI yang membacanya atas nama mereka. Organisasi yang jelas dan label yang jujur melayani keduanya. Profitina memberi saran dan penggunalah yang memutuskan; struktur yang terbaca itulah yang memungkinkan pengguna memutuskan dengan baik.

8.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Semuanya sama keras.** Ketika setiap unsur tebal, berwarna, atau besar, tidak ada yang menonjol. Tentukan urutan kepentingannya dan biarkan sebagian besar hal tenang.
- **Warna acak untuk ikon.** Warna harus menandakan makna (dapat diketuk, berhasil, peringatan), bukan menghias. Hal yang serupa harus tampak serupa.
- **Celah yang tidak mengelompokkan.** Jika jarak antarkelompok tidak lebih besar daripada jarak di dalam kelompok, kedekatan gagal. Pakai skala jarak dengan langkah yang jelas berbeda.
- **Terlalu banyak ukuran huruf.** Ukuran yang berbeda satu piksel tampak seperti kekeliruan. Pakai skala modular.

- **Memeriksa kontras dengan mata.** Hitunglah (Persamaan (8.4)); periksa setiap pasangan yang dipakai, termasuk teks di atas kartu dan permukaan tembus pandang.
- **Warna sebagai satu-satunya tanda.** Selalu tambahkan teks atau ikon.
- **Meniru gaya platform tanpa pengaturan aksesibilitasnya.** Hormati pengaturan ukuran teks sistem, pengurangan ketembusan, dan mode gelap.
- ***Design system* tanpa token atau pemilik.** Nilai yang ditulis langsung akan melenceng; sistem yang tidak dipelihara siapa pun akan ditinggalkan.
- **Mengorganisasi menurut bagan organisasi.** Nasabah tidak mengenal divisi-divisi bank. Tanyakan kepada mereka dengan *card sorting*.
- **Label internal.** "Mutasi", "Inquiry", dan "Setor" adalah kata-kata bank. Pakai kata-kata nasabah dan simpan kata lama sebagai kata kunci pencarian.
- **Memaksa setiap butir ke satu tempat.** Ketika partisipan membelah sebuah butir di antara kelompok, cantumkan secara silang.

8.14 Ringkasan Bab

- Desain visual membuat struktur terlihat. Prinsip Gestalt kedekatan, kesamaan, wilayah bersama, dan ketertutupan (**Gambar 8.1**) menciptakan kelompok; kontras ukuran, ketebalan, warna, dan posisi menciptakan hierarki.
- Huruf harus cukup besar, dalam baris yang nyaman, dan dipilih dari skala modular (Persamaan (8.1)); jarak mengikuti skala 4 atau 8 piksel dan sebuah grid.
- WCAG 2.2 (juga ISO/IEC 40500:2025) mensyaratkan 4,5:1 untuk teks isi dan 3:1 untuk teks besar serta bagian UI. Rasio kontras diturunkan dari kanal sRGB yang dilinearkan (Persamaan (8.2)), luminans relatif (Persamaan (8.3)), dan Persamaan (8.4). WCAG 3.0 masih berupa draf.
- `contrast.py` dan `Contrast.java` membaca berkas W3C Design Tokens 2025.10, memeriksa setiap pasangan, dan menyarankan perbaikan; pada aplikasi bank, enam dari sepuluh pasangan gagal, termasuk tombol putih-di-atas-hijau yang lama (2,78:1) (**Gambar 8.2**).
- *Design system* memadukan token, komponen, dan pedoman; token primitif dan semantik (**Gambar 8.3**) membuat perubahan dan tema menjadi aman. Material 3 Expressive, Liquid Glass, dan Fluent 2 adalah bahasa platform tahun 2026; material tembus pandang menjadikan kontras perhatian yang terus-menerus.
- Arsitektur informasi mengorganisasi, memberi label, menavigasi, dan mencari. *Card sorting* terbuka, yang dianalisis dengan kemiripan kemunculan bersama (Persamaan (8.5)) dan *average linkage* (Persamaan (8.6)), mengusulkan empat kelompok untuk aplikasi bank dan menunjukkan bahwa nasabah menaruh angsuran pinjaman di bawah tagihan (**Gambar 8.4**).

8.15 Soal Latihan

1. Hitung dengan tangan rasio kontras abu-abu `#6B7280` di atas putih, lalu periksa terhadap 4,83:1 dari program. Apakah lulus untuk teks isi? Untuk teks besar?
2. Dengan skala modular berdasar 16 px dan rasio 1,2, daftarkan ukuran-ukurannya untuk $k = -1$ sampai 4, dibulatkan ke piksel utuh. Bandingkan dengan skala rasio 1,25 di Subbab 8.4.2: mana yang memberi tingkatan yang lebih jelas di layar ponsel yang kecil?
3. Tambahkan pasangan `amber,navy,ui` (ikon *amber* di atas bilah *navy*) ke `pairs.csv` lalu jalankan pemeriksanya. Jelaskan hasilnya dengan Persamaan (8.4).

4. Pemeriksa kontras menggelapkan warna yang lebih gelap. Tulis varian yang *mencerahkan* warna yang lebih terang (mencampurnya dengan putih) dan bandingkan kedua saran untuk `grey` di atas `grey-light`. Kapan perancang lebih memilih masing-masing?
5. Jalankan `cardsort.py` dengan potongan 0,6 dan 0,7. Berapa kelompok yang Anda peroleh, dan kartu mana yang keluar dari kelompoknya? Apa artinya bagi kestabilan penempatan kartu-kartu itu?
6. Jelaskan dengan kata-kata Anda sendiri mengapa *single linkage* (kemiripan pasangan yang *paling* mirip) dapat menggabungkan kelompok transfer dan tagihan terlalu dini, dengan kartu *Transfer terjadwal* sebagai contoh. Ubah `link` agar memakai *single linkage* dan periksa penjelasan Anda.
7. Rancang *tree testing* untuk struktur empat tab bank yang baru: tulis lima tugas, termasuk satu tentang angsuran pinjaman dan satu tentang kata lama "mutasi", lalu sebutkan hasil seperti apa yang akan membuat Anda mengubah strukturnya.
8. Seorang perancang mengusulkan teks putih di atas bilah tembus pandang bergaya Liquid Glass di atas foto sawah. Jelaskan bagaimana Anda akan memeriksa dan menjamin kontras bilah ini.

8.16 Penyelesaian Soal Latihan

1. Kanalnya 0x6B = 107, 0x72 = 114, 0x80 = 128; dibagi 255: 0,4196, 0,4471, 0,5020. Dilinearkan dengan Persamaan (8.2): $(0.4746/1.055)^{2.4} = 0.4499^{2.4} = 0.147$; $(0.5021/1.055)^{2.4} = 0.4759^{2.4} = 0.168$; $(0.5570/1.055)^{2.4} = 0.5279^{2.4} = 0.216$. $L = 0.2126 \times 0.147 + 0.7152 \times 0.168 + 0.0722 \times 0.216 = 0.0313 + 0.1202 + 0.0156 = 0.167$. $\mathrm{CR} = 1.05/0.217 = 4.83$. Pasangan ini lulus untuk teks isi (4,5) dan teks besar (3), tetapi dengan selisih tipis: di atas latar yang sedikit lebih gelap daripada putih ia gagal, seperti diperlihatkan pasangan dengan `grey-light`.

2. 16×1.2^k untuk $k = -1, \ldots, 4$: 13,3, 16, 19,2, 23,0, 27,6, 33,2, dibulatkan 13, 16, 19, 23, 28, 33. Skala 1,25 memberi 13, 16, 20, 25, 31, 39. Di layar kecil, 1,25 memberi tingkatan yang lebih sedikit tetapi lebih jelas berbeda serta ukuran puncak yang lebih besar untuk saldo; 1,2 memberi lebih banyak tingkatan, yang membantu layar yang padat teks tetapi berisiko menghasilkan ukuran yang tampak serupa (16 dan 19).

3. Tambahkan baris `amber,navy,ui`. Program melaporkan rasio 5,50:1, `AA pass`. Menurut Persamaan (8.4), $(0.4134 + 0.05)/(0.0343 + 0.05) = 0.4634/0.0843 = 5.50$. *Amber*, yang gagal di atas putih, lulus di atas *navy*: kontras adalah sifat pasangan, tidak pernah sifat satu warna.

4. Fungsi `lighten(hex, share)` menghitung `round(c + (255 - c) * share)` untuk setiap kanal, dan gelung perbaikan mencerahkan warna yang lebih terang sampai rasionya tercapai. Untuk teks abu-abu di atas `grey-light` (`#F3F4F6`), mencerahkan latar mencapai 4,5:1 di dekat putih murni (sekitar `#FDFDFE`), yang mengubah tampilan kartu; menggelapkan teks menjadi `#69707D` nyaris tidak terlihat. Perancang mencerahkan latar ketika warna kartu tidak penting bagi desain, dan menggelapkan teks ketika latarnya bagian dari merek atau warna teks harus konsisten di semua layar.

5. Pada 0,6, *Transfer terjadwal* (bergabung pada 0,53), *Angsuran pinjaman* (0,51), dan *Ganti PIN* (0,55) menjadi kelompok sendiri-sendiri, sehingga ada tujuh kelompok; pada 0,7, *Bayar dengan QRIS* (0,65) juga terpisah dan kelompoknya menjadi delapan. Keempat kartu itulah yang penempatannya tidak disepakati partisipan. Kartu-kartu itu calon untuk dicantumkan silang, diberi label yang lebih jelas, atau diamati lebih dekat dalam *tree testing*, sementara kartu yang bergabung pada 1,00 dapat ditempatkan dengan yakin.

6. *Single linkage* mengambil kemiripan *terbesar* antara kartu mana pun di satu kelompok dan kartu mana pun di kelompok lain. *Transfer terjadwal* ditaruh bersama tagihan oleh beberapa partisipan (mereka melihatnya sebagai "membayar rutin"), sehingga kemiripannya dengan kartu tagihan tinggi; begitu ia berada di kelompok transfer, *single linkage* menganggap seluruh kelompok transfer dekat dengan tagihan, dan kedua kelompok bergabung pada tingkat yang tinggi, meskipun sebagian besar kartu transfer tidak pernah dikelompokkan bersama tagihan. Mengganti `link` dengan nilai maksimum `sim[i][j]` atas semua pasangan memperlihatkan kelompok transfer dan tagihan bergabung pada 0,55, jauh di atas 0,22 pada *average linkage*. Inilah kelemahan *chaining* (perangkaian) pada *single linkage*.

7. Tugas: "Bayar angsuran pinjaman bulan ini", "Lihat uang yang masuk ke rekening Anda minggu lalu" (menguji "Riwayat transaksi" dan kata lama "mutasi"), "Kirim Rp200.000 ke rekening anak Anda di bank lain", "Beli token listrik Rp50.000", "Cari jam buka kantor cabang terdekat". Keberhasilan berarti mencapai butir yang benar pada jalur pertama; catat tab pertama yang dipilih. Tugas dengan keberhasilan percobaan pertama di bawah sekitar 80%, atau satu tab salah yang dipilih banyak partisipan, mengisyaratkan perubahan: tautan silang, tab yang diganti namanya, atau butir yang dipindah. Jika sebagian besar partisipan mencari "mutasi" di bawah *Bantuan*, misalnya, riwayat transaksi membutuhkan label yang lebih jelas atau tautan dari saldo.

8. Kontras di atas foto tidak dapat dihitung sekali saja, karena latarnya berubah-ubah. Jaminlah dengan desain: taruh lapisan penyangga di bawah teks (rona *navy* dengan opasitas yang cukup tinggi sehingga bagian paling terang dari foto mana pun di belakangnya tetap memberi sekurang-kurangnya 4,5:1 dengan teks putih), atau bayangan maupun garis tepi teks, dan hormati pengaturan Reduce Transparency serta Tinted pada sistem. Periksa dengan menghitung rasio warna teks terhadap latar gabungan yang *paling terang*, yaitu rona yang dicampur di atas putih murni, dan dengan menguji tangkapan layar berisi foto sungguhan yang terang. Pendekatan yang sama itulah yang diterapkan pilihan Tinted iOS 26.1 pada tingkat sistem.

Daftar Pustaka

[1] L. Rosenfeld, P. Morville, and J. Arango, *Information Architecture: For the Web and Beyond*, 4th ed. Sebastopol, CA, USA: O'Reilly Media, 2015.

[2] J. Wagemans, J. H. Elder, M. Kubovy, S. E. Palmer, M. A. Peterson, M. Singh, and R. von der Heydt, "A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure–ground organization," *Psychological Bulletin*, vol. 138, no. 6, pp. 1172–1217, 2012.

[3] W3C, *Web Content Accessibility Guidelines (WCAG) 2.2*, W3C Recommendation, Oct. 5, 2023, latest version accessed Sep. 2026. [Online]. Available: https://www.w3.org/TR/WCAG22/

[4] W3C Web Accessibility Initiative, "WCAG 2.2 Approved as an ISO Standard," Oct. 21, 2025. [Online]. Available: https://www.w3.org/WAI/news/2025-10-21/wcag22-iso

[5] W3C Web Accessibility Initiative, "For Review: WCAG 3 Working Draft – September 2026," Sep. 10, 2026. [Online]. Available: https://www.w3.org/WAI/news/2026-09-10/wcag3/

[6] W3C Design Tokens Community Group, "Design Tokens specification reaches first stable version," Oct. 28, 2025. [Online]. Available: https://www.w3.org/community/design-tokens/2025/10/28/design-tokens-specification-reaches-first-stable-version/

[7] Google Design, "Expressive Design: Google's UX Research," 2025. [Online]. Available: https://design.google/library/expressive-material-design-google-research

[8] Apple Newsroom, "Apple introduces a delightful and elegant new software design," Jun. 9, 2025. [Online]. Available: https://www.apple.com/newsroom/2025/06/apple-introduces-a-delightful-and-elegant-new-software-design/

[9] Engadget, "How to adjust the Liquid Glass effect in iOS 26.1," Nov. 9, 2025. [Online]. Available: https://www.engadget.com/mobile/smartphones/how-to-adjust-the-liquid-glass-effect-in-ios-261-203634681.html

[10] Microsoft, "Fluent 2 Design System," accessed Sep. 2026. [Online]. Available: https://fluent2.microsoft.design/

[11] D. Spencer, *Card Sorting: Designing Usable Categories*. Brooklyn, NY, USA: Rosenfeld Media, 2009.

Bab 9

Gaya Interaksi: GUI, Sentuh, Suara, Gestur

Segelas es teh bisa dipesan dengan banyak cara: berseru dari seberang meja, menunjuk gambar, mencentang formulir kertas, mengetik perintah, mengetuk ponsel, atau berbicara kepada pengeras suara di atas meja. Setiap cara adalah sebuah *interaction style* (gaya interaksi), dan setiap gaya membuat sebagian hal mudah dan sebagian lain sulit. Bab ini mengikuti sebuah warung kecil di Surabaya yang ingin menerima pesanan dari tablet di meja kasir, lalu memakainya untuk menjelaskan gaya-gaya interaksi utama masa kini: bahasa perintah, menu dan formulir, manipulasi langsung pada antarmuka grafis, sentuhan, gestur, suara, dan antarmuka spasial yang baru. Di akhir bab Anda akan mampu menyebut gaya-gaya itu dan mengatakan apa yang dituntut masing-masing dari pengguna; menjelaskan jurang eksekusi dan jurang evaluasi serta mengapa manipulasi langsung mempersempit keduanya; memakai hukum Fitts dan versinya untuk sentuhan jari guna menalar ukuran target; menerapkan aturan ukuran target WCAG 2.2 serta platform Apple dan Google; merancang percakapan suara yang meminta konfirmasi ketika ragu; serta membangun, dalam Java dan Python, satu inti pesanan dengan tiga antarmuka — baris perintah, baris "suara" yang diketik, dan kios sentuh — beserta program yang mengukur setiap target ketuk kios itu di peramban sungguhan.

9.1 Meja Kasir Bu Tini

Bu Tini mengelola warung dekat sebuah kampus di Surabaya: lima menu, satu juru masak, satu pembantu, dan pada tengah hari kerumunan mahasiswa, pengemudi ojek, dan pegawai kantor yang semuanya ingin memesan bersamaan. Amati meja kasirnya sepuluh menit dan Anda akan melihat semua gaya interaksi dalam bab ini bekerja. Seorang pelanggan tetap berseru "*Es teh dua, Bu!*" dari pintu, dan Bu Tini menjawab "*Dua, ya?*" untuk memastikan — perintah suara dengan konfirmasi lisan. Pelanggan baru yang belum mengenal menu menunjuk kartu menu berlaminasi yang bergambar — pemilihan menu, dilakukan dengan gestur. Seorang pengemudi yang terburu-buru mengacungkan dua jari sambil mengangguk ke arah stoples teh, dan itu pun berhasil, karena Bu Tini dapat melihat dia dan stoplesnya sekaligus. Para pegawai kantor yang memesan untuk satu ruangan menulis pesanan pada secarik kertas dengan satu baris untuk setiap menu — sebuah formulir. Dan para mahasiswa bertanya apakah mereka bisa memesan dan membayar dari ponsel.

Keponakan Bu Tini, Dimas, mahasiswa informatika, menawarkan diri membuat kios tablet untuk meja kasir. Gagasan pertamanya sederhana: tampilkan menu di layar, biarkan orang mengetuk. Lalu pertanyaan-pertanyaan bermunculan. Perlukah kios juga memahami kata-kata yang diketik atau diucapkan, bagi pelanggan yang lebih suka berkata "nasi goreng dua" daripada mencari tombol? Seberapa besar tombol harus dibuat agar ibu jari yang belepotan sambal tetap mengenai menu yang benar? Apa yang harus dilakukan kios ketika mendengar sesuatu yang tidak dipahaminya — menebak, atau bertanya? Dan bagaimana nasib pelanggan lansia Bu Tini, yang sebagian belum pernah menyentuh tablet?

Itulah pertanyaan-pertanyaan bab ini, dan maknanya jauh melampaui satu warung. Menurut survei Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) yang dirilis Mei 2026, 81,72% penduduk Indonesia — sekitar 235 juta orang — kini memakai internet [1]. Hampir semuanya menjumpai banyak gaya interaksi dalam sehari: menu tombol di samping layar ATM, peta aplikasi ojek daring yang digeser dengan satu jari, pesan suara di aplikasi obrolan, kode QR yang dipindai di kasir. Perancang yang hanya mengenal satu gaya akan memaksa setiap tugas masuk ke gaya itu, dan banyak pengguna akan menanggung akibatnya.

9.2 Gagasannya dalam Bahasa Sehari-hari

9.2.1 Apa itu gaya interaksi

Gaya interaksi adalah cara seseorang memberi tahu komputer apa yang harus dilakukan dan melihat apa yang telah dilakukan komputer. Ia menjawab dua pertanyaan kecil. *Bagaimana saya mengatakannya?* — dengan mengetik kata, memilih dari daftar, mengisi bagian kosong, menunjuk dan memindahkan objek, menyentuh, bergestur, berbicara. *Bagaimana saya melihat hasilnya?* — sebagai baris teks, gambar yang berubah, bunyi, getaran.

Buku teks klasik bidang ini, *Designing the User Interface* karya Shneiderman dan rekan-rekannya, mengelompokkan cara-cara "mengatakan" itu menjadi lima gaya utama: manipulasi langsung, pemilihan menu, pengisian formulir, bahasa perintah, dan bahasa alami [2]. Sentuhan, gestur, suara, dan interaksi spasial lebih tepat dipandang sebagai *saluran* baru bagi gaya-gaya lama daripada gaya baru: aplikasi ponsel sebagian besar menawarkan manipulasi langsung dan menu melalui sentuhan; *voice assistant* menawarkan bahasa alami melalui ucapan; *headset* seperti Apple Vision Pro menawarkan manipulasi langsung melalui mata dan cubitan jari.

> **DEFINISI.** *Gaya interaksi* adalah sekelompok cara pengguna menyampaikan niat kepada sistem dan menangkap tanggapannya. Lima gaya utamanya adalah *bahasa perintah* (pengguna mengetik perintah dengan sintaksis tetap), *pemilihan menu* (pengguna memilih dari pilihan yang terdaftar), *pengisian formulir* (pengguna mengisi ruas berlabel), *manipulasi langsung* (pengguna bertindak atas objek yang terlihat dan langsung melihat akibatnya), dan *bahasa alami* (pengguna menulis atau berbicara seperti kepada orang). *Modalitas* adalah saluran indrawi yang dipakai: papan ketik, penunjuk, sentuhan, gestur, suara, tatapan.

9.2.2 Peta gaya-gaya interaksi

Tabel 9.1 menjajarkan gaya-gaya itu, beserta pertanyaan yang menentukan pilihan di antaranya: apa yang harus *diingat* pengguna, dan apa yang harus *dilihatnya*? Bahasa perintah meminta pengguna memanggil kata yang tepat dari ingatan (*recall*); menu membiarkan pengguna mengenalinya dalam daftar (*recognition*). Mengenali lebih mudah daripada mengingat, sehingga menu cocok bagi pengguna sesekali; mengingat lebih cepat setelah dikuasai, sehingga para ahli menyukai perintah dan pintasan papan ketik.

Tabel 9.1 Lima gaya interaksi utama, apa yang dituntut masing-masing dari pengguna, dan di mana masing-masing muncul di meja kasir Bu Tini.

Gaya	Pengguna…	Unggul ketika	Lemah ketika	Di warung
Bahasa perintah	mengetik kata bersintaksis tetap	pengguna ahli, tugas berulang, skrip	pemula; salah sintaksis	`add esteh 2` di laptop
Pemilihan menu	memilih dari daftar	pengguna sesekali; pilihan sedikit	pilihan banyak, pohon dalam	kartu menu, ATM
Pengisian formulir	mengisi bagian kosong berlabel	data terstruktur (alamat, slip pesanan)	ruas yang tak dipahami pengguna	slip pesanan kantor
Manipulasi langsung	bertindak atas objek terlihat	melihat dan membatalkan hasil seketika	tindakan abstrak, objek banyak	mengetuk menu di kios
Bahasa alami	berbicara atau menulis bebas	tangan atau mata sibuk, menu tak dikenal	ambiguitas, bising, privasi	"*Es teh dua, Bu!*"

Tidak ada gaya yang unggul di mana-mana. ATM memakai menu karena bank tidak dapat melatih nasabahnya; pemrogram memakai perintah karena dua puluh kata yang sama diketik seribu kali sehari; kios memakai manipulasi langsung karena menunya sedikit dan gambarnya jelas; pelanggan tetap di pintu memakai bahasa alami karena tangannya memegang helm. Sistem yang baik sering memadukan beberapa gaya: pesanan yang sama dapat diberikan dengan lebih dari satu cara, dan sistem menampilkan hasilnya dalam satu gambaran bersama.

9.2.3 Contoh yang dikerjakan: segelas teh, lima cara

Ambil pesanan "dua es teh" dan hitung apa yang diminta setiap gaya.

1. **Perintah.** Ketik `add esteh 2` lalu tekan Enter: 13 ketukan tombol. Pengguna harus tahu kode `esteh`, kata kerja `add`, dan urutan kata-katanya. Salah ketik seperti `add esteh x` menghasilkan galat.
2. **Menu.** Pada menu bernomor, tekan 1 (minuman), 1 (es teh), lalu 2 untuk jumlahnya: tiga tekanan. Pengguna harus membaca daftar, tetapi tidak perlu mengingat apa pun.
3. **Formulir.** Temukan baris "Es teh" pada slip dan tulis 2 di kotaknya: satu ruas. Formulir juga memperlihatkan semua hal lain yang bisa dipesan.
4. **Manipulasi langsung.** Ketuk tombol "Es teh" dua kali dan lihat baris pesanan menjadi "2 x Es teh". Hasilnya terlihat setelah setiap ketukan, dan tombol minus membatalkan kekeliruan.
5. **Bahasa alami.** Ucapkan "*es teh dua*". Tiga kata — tetapi sistem kini harus *memahaminya*, dan ia mungkin mendengar "*es teh*" sebagai "*estate*", atau "*dua*" sebagai "*juga*".

Hitungan ini memperlihatkan tawar-menawarnya: gaya yang sedikit menuntut ingatan pengguna banyak menuntut layar, dan gaya yang paling sedikit menuntut pengguna (berbicara) paling banyak menuntut sistem. Sisa bab ini membuat tawar-menawar itu tepat.

9.3 Teori dan Buktinya

9.3.1 Dua jurang, dan mengapa manipulasi langsung mempersempitnya

Norman menggambarkan setiap pemakaian alat sebagai penyeberangan dua jurang [3]. **Jurang eksekusi** (*gulf of execution*) adalah jarak antara apa yang diinginkan pengguna ("dua teh") dan apa yang diizinkan sistem untuk dilakukan (mengetik perintah? mencari tombol?). **Jurang evaluasi** (*gulf of evaluation*) adalah jarak antara apa yang ditampilkan sistem dan apa yang perlu diketahui pengguna ("pesanan saya sudah masuk?"). Antarmuka yang baik menjembatani keduanya: ia membuat tindakan yang mungkin terlihat dan memperlihatkan keadaan sistem

dengan jelas setelah setiap tindakan. Bab 1 telah memperkenalkan kedua jurang ini secara umum; di sini keduanya menjelaskan gaya-gaya interaksi.

Bahasa perintah membiarkan jurang eksekusi tetap lebar — pengguna harus menerjemahkan keinginannya ke dalam sintaksis — dan sering membiarkan jurang evaluasi lebar pula, karena perintah dapat berhasil tanpa kabar. Shneiderman menciptakan istilah **manipulasi langsung** (*direct manipulation*) untuk antarmuka yang mempersempit kedua jurang sekaligus [2]: objek yang diminati terus terlihat; pengguna bertindak atasnya dengan tindakan fisik (menunjuk, mengetuk, menyeret), bukan sintaksis rumit; dan tindakan bersifat cepat, bertahap, dan dapat dibatalkan, dengan akibat yang langsung terlihat. Lembar kerja, program menggambar, peta di aplikasi ojek daring, dan kios dalam bab ini semuanya manipulasi langsung. Harganya adalah ruang layar dan sulitnya menyatakan tindakan abstrak: "setiap Jumat pukul dua belas, pesan sepuluh teh untuk kantor" mudah diketik sebagai perintah tetapi canggung dinyatakan dengan menyeret.

9.3.2 Peristiwa, keadaan, dan kecepatan umpan balik

Setiap antarmuka grafis, sentuh, atau suara dibangun dengan cara yang sama di bawahnya: ia menunggu **peristiwa** (*event*) — tekanan tombol, ketukan, ucapan yang selesai, balasan jaringan — dan menjalankan **penangan** (*handler*) untuk setiap peristiwa, yang mengubah **keadaan** (*state*) program dan menggambar ulang apa yang dilihat pengguna. Struktur berbasis peristiwa inilah yang membuat antarmuka tidak pernah "berjalan dari atas ke bawah" seperti program *batch*; ia berputar dalam gelung, dan penggunalah yang menentukan apa yang terjadi berikutnya. Kios kita memiliki keadaan yang mungil (keranjang) dan tiga peristiwa (tambah, kurangi, bayar); baris suara kita memiliki satu keadaan lagi, yaitu pesanan yang *menunggu konfirmasi*, yang diperkenalkan Subbab 9.5.

Seberapa cepat penangan harus menampilkan hasilnya? Penelitian sejak Miller pada 1968 yang dirangkum Nielsen memberikan tiga batas yang masih dipakai hingga kini [4]. Sekitar **0,1 detik** adalah batas agar pengguna merasa sistem bereaksi *seketika*, seperti tombol yang berubah warna di bawah jari; sekitar **1 detik** adalah batas agar alur pikiran pengguna tidak terputus; sekitar **10 detik** adalah batas untuk menjaga perhatian pengguna tetap pada tugas. Lewat satu detik, tunjukkan bahwa sesuatu sedang terjadi; lewat sepuluh detik, tunjukkan kemajuan dan biarkan pengguna mengerjakan hal lain. Batas-batas ini bukan hukum alam, tetapi merupakan pedoman yang andal, dan karena itulah aplikasi yang dibuat dengan baik langsung bereaksi pada ketukan dan mengerjakan pekerjaan lambatnya belakangan.

9.3.3 Sentuhan: jari bukan penunjuk tetikus

Penunjuk tetikus selebar satu piksel; bantalan jari menyentuh kaca dengan bidang beberapa milimeter dan menutupi target yang ditekannya sendiri. Dua akibat mengikutinya: target harus lebih besar, dan tawar-menawar antara kecepatan dan ketepatan berubah.

Hukum Fitts meramalkan waktu untuk menunjuk sebuah target dari jarak dan lebarnya. Dalam bentuk yang dipakai di IMK ("formulasi Shannon"), waktu MT untuk mengenai target selebar W pada jarak D adalah

$$MT = a + b\log_2\left(\frac{D}{W} + 1\right) \tag{9.1}$$

dengan a dan b tetapan yang diukur untuk suatu perangkat dan orang, sedangkan logaritmanya, yang disebut **indeks kesulitan** ID dan diukur dalam bit, menangkap seberapa sulit target itu [5].

Bab 2 telah menurunkan hukum ini untuk tetikus. Persamaan (9.1) menyatakan bahwa memperkecil lebar target menjadi setengahnya kira-kira sama mahalnya dengan menggandakan jaraknya: target kecil adalah target lambat.

Pada layar sentuh, hukum ini perlu satu koreksi. Bi, Li, dan Zhai menunjukkan bahwa sentuhan jari memiliki ketidaktepatan *mutlak* yang tidak mengecil betapa pun pelan pengguna bergerak: titik kontak jari tetap tersebar di sekitar titik yang dibidik, sehati-hati apa pun [6]. **Hukum FFitts** (Fitts untuk jari) mereka membagi sebaran titik sentuh yang teramati, dengan simpangan baku σ, menjadi bagian yang disebabkan kecepatan gerakan dan bagian σ_a yang disebabkan jari itu sendiri, lalu hanya menyimpan bagian pertama di dalam indeks kesulitan:

$$ID_f = \log_2\left(\frac{D}{\sqrt{2\pi e\,(\sigma^2 - \sigma_a^2)}} + 1\right) \tag{9.2}$$

Makna praktis Persamaan (9.2) sederhana: di bawah ukuran tertentu, memperkecil target sentuh tidak sekadar memperlambat pengguna, tetapi menimbulkan *kesalahan* yang tidak dapat dicegah oleh kehati-hatian apa pun. Karena itulah setiap platform sentuh menetapkan ukuran target minimum.

> **CONTOH.** Pada halaman kios selebar 360 piksel, tombol "Bayar" dan tautan bahasa sama-sama berjarak sekitar $D = 300$ piksel CSS dari tempat ibu jari beristirahat. Versi pertama halaman menggambar tautan bahasa sebagai sebaris teks kecil setinggi 13 piksel saja; tombol menu setinggi 64 piksel. Dengan Persamaan (9.1): untuk tautan, $ID = \log_2(300/13 + 1) = \log_2(24.08) = 4.59$ bit; untuk tombol menu, $ID = \log_2(300/64 + 1) = \log_2(5.69) = 2.51$ bit. Tautan itu hampir dua bit lebih sulit — dan menurut Persamaan (9.2), target 13 piksel juga mendekati ukuran ketika sebaran jari sendiri menyebabkan meleset.

Tabel 9.2 menghimpun ukuran minimum yang berlaku pada September 2026. **Web Content Accessibility Guidelines (WCAG) 2.2** dari W3C, yang disahkan sebagai standar internasional ISO/IEC 40500:2025 pada Oktober 2025, menambahkan kriteria keberhasilan 2.5.8 *Target Size (Minimum)* pada tingkat AA: target penunjuk harus berukuran sekurang-kurangnya 24 kali 24 piksel CSS, kecuali jaraknya cukup sehingga lingkaran 24 piksel yang berpusat padanya tidak menyentuh target lain, atau ia berada di tengah kalimat, atau berlaku salah satu dari beberapa pengecualian lain [7]. Kriteria yang lebih lama, 2.5.5 pada tingkat AAA, meminta 44 kali 44. Human Interface Guidelines Apple menganjurkan bidang ketuk sekurang-kurangnya 44 kali 44 poin di iPhone dan iPad [8]; Material Design Google meminta sekurang-kurangnya 48 kali 48 piksel bebas-kerapatan (dp), sekitar 9 mm, untuk target sentuh di Android [9].

Tabel 9.2 Ukuran minimum target sentuh dan penunjuk per September 2026 (KK = kriteria keberhasilan).

Sumber	Target minimum	Tingkat atau status
WCAG 2.2, KK 2.5.8 *Target Size (Minimum)*	24 × 24 px CSS, atau pengecualian jarak	AA (diwajibkan sebagian besar regulasi)
WCAG 2.2, KK 2.5.5 *Target Size (Enhanced)*	44 × 44 px CSS	AAA
Apple Human Interface Guidelines	44 × 44 pt	pedoman platform
Material Design 3 (Android)	48 × 48 dp (sekitar 9 mm)	pedoman platform

9.3.4 Gestur: ampuh, tetapi tak terlihat

Gestur adalah gerakan yang membawa perintah: usap untuk menghapus obrolan, cubit untuk memperbesar peta, tekan lama untuk memunculkan menu, tarik ke bawah untuk menyegarkan.

Gestur cepat bagi yang mengenalnya dan tak terlihat bagi yang tidak — tidak ada apa pun di layar yang berkata "Anda bisa mengusap di sini". Norman menyebut isyarat yang memberi tahu pengguna bahwa suatu tindakan mungkin dilakukan sebagai **penanda** (*signifier*) [3], dan gestur biasanya tidak memilikinya. Karena itu perancang mengikuti tiga aturan. Pertama, gestur hendaknya menjadi pintasan, bukan satu-satunya jalan: aplikasi obrolan yang menghapus dengan usapan juga menyediakan butir hapus di menu. Kedua, gestur hendaknya mengikuti konvensi platform, karena pengguna mempelajarinya sekali per ponsel, bukan sekali per aplikasi. Ketiga, gestur tidak boleh menyingkirkan orang yang tidak mampu melakukannya.

Aturan terakhir itu kini menjadi bagian WCAG 2.2. Kriteria keberhasilan 2.5.1 *Pointer Gestures* mensyaratkan bahwa apa pun yang dilakukan dengan gestur multititik atau berbasis lintasan (cubitan, usapan sepanjang lintasan) juga dapat dilakukan dengan satu penunjuk tanpa lintasan, misalnya ketukan; kriteria 2.5.7 *Dragging Movements*, yang baru di WCAG 2.2, mensyaratkan bahwa apa pun yang dilakukan dengan menyeret juga dapat dilakukan dengan ketukan atau klik tunggal [7]. Peta yang hanya dapat diperbesar dengan mencubit, atau daftar yang hanya dapat diurutkan ulang dengan menyeret, gagal; tambahkan tombol plus dan minus, atau panah naik dan turun, maka ia lulus. Bab 11 kembali ke kriteria-kriteria ini dari sisi pengguna dengan disabilitas motorik.

9.3.5 Suara: alami, tetapi ambigu

Suara adalah antarmuka tertua sekaligus, bagi komputer, salah satu yang tersulit. Ia cocok untuk saat tangan dan mata sedang sibuk — memasak, berkendara, memegang helm — dan bagi orang yang kesulitan dengan layar, termasuk banyak lansia dan orang dengan penglihatan rendah. *Voice assistant* utama berubah dengan cepat: pada Agustus 2026 Google mengumumkan bahwa, mulai 4 September 2026, Gemini secara bertahap menggantikan Google Assistant di ponsel dan tablet Android, jam Wear OS, dan Android Auto [10], dan *assistant* yang dibangun di atas model bahasa besar kini menerima permintaan panjang yang dirumuskan longgar alih-alih frasa perintah yang baku. Penelitian di CHI 2026 menjajaki bagaimana *voice assistant* dapat membantu orang dewasa dengan disabilitas intelektual mengerjakan tugas sehari-hari secara mandiri [11]; temuan lama bahwa suara menolong sebagian pengguna dan membuat frustrasi sebagian lain tetap benar.

Mesin apa pun yang ada di bawahnya, sebuah **antarmuka pengguna suara** (*voice user interface*, VUI) menghadapi tiga masalah yang tidak dihadapi layar sentuh. Ia *tidak memiliki menu yang terlihat*: pengguna tidak tahu apa yang boleh diucapkan. Ia *tidak pasti*: pengenal ucapan dapat salah dengar, dan kata-katanya dapat ambigu — "*es teh dua*" bisa berarti dua teh atau satu teh untuk meja dua. Dan ia *linear serta sekejap*: daftar dua puluh menu yang dibacakan sudah terlupa sebelum selesai. Jawaban desain klasiknya adalah sebagai berikut.

- **Konfirmasi sebanding dengan risiko dan keraguan.** Konfirmasi *implisit* mengulang apa yang dipahami sambil tetap melanjutkan ("Dua es teh — ada lagi?"); konfirmasi *eksplisit* berhenti dan bertanya ("Maksudnya satu kopi? Katakan ya untuk memastikan"). Pakai konfirmasi eksplisit ketika pengenal ragu atau ketika kesalahan mahal akibatnya (membayar, mengirim uang, menghapus).
- **Katakan apa yang bisa dikatakan.** Ketika sistem tidak paham, jangan hanya berkata "Maaf?"; berikan contoh permintaan yang baik ("Silakan sebut nama menu, misalnya *nasi goreng*").
- **Sediakan jalan kembali.** Setiap tindakan lisan hendaknya dapat dibatalkan dengan suara ("*batal*") dan terlihat di layar bila ada layar.

"*Dua, ya?*" milik Bu Tini persis merupakan konfirmasi eksplisit. Subbab 9.5 memasukkannya ke dalam program dengan sebuah bilangan, yaitu **keyakinan** (*confidence*) c suatu tafsiran: bagian kata dalam ucapan yang dipahami oleh pengurai,

$$c = \frac{k}{m} \tag{9.3}$$

dengan m banyaknya kata yang terdengar dan k banyaknya kata yang cocok dengan menu, bilangan, kata niat, atau kata pengisi yang tak berbahaya. Pengenal ucapan sungguhan melaporkan keyakinan dari model akustik dan model bahasanya; Persamaan (9.3) adalah pengganti sederhana yang dapat dijelaskan dengan peran yang sama. Ketika c di bawah suatu ambang, sistem bertanya alih-alih bertindak.

> **WAWASAN KUNCI:** Antarmuka suara yang tidak pernah bertanya sedang menebak, dan antarmuka suara yang selalu bertanya itu melelahkan. Keterampilannya terletak pada memilih kapan bertanya: ketika sistem ragu, dan ketika tebakan yang salah mahal akibatnya.

9.3.6 Bahasa desain masa kini, dan interaksi di ruang

Platform-platform besar menerbitkan **bahasa desain** yang menetapkan rupa dan perilaku kontrolnya, dan pada 2025 bahasa-bahasa itu berubah cukup nyata. Pada 15 September 2025 Apple merilis iOS 26 beserta saudara-saudaranya dengan **Liquid Glass**, bahan tembus pandang untuk kontrol dan navigasi yang memantulkan dan membiaskan apa yang ada di belakangnya serta memberi konten bagian layar yang lebih luas [12]. **Material 3 Expressive** dari Google, yang diumumkan Mei 2025, menambahkan lebih banyak bentuk, warna, kontras huruf, dan gerak yang memantul; Google melaporkan bahwa ia bertumpu pada 46 studi dengan lebih dari 18.000 partisipan, dan di dalamnya orang menemukan elemen antarmuka kunci hingga empat kali lebih cepat pada desain ekspresif [13]. **Fluent 2** dari Microsoft adalah sistem desain bersama untuk Windows dan Microsoft 365 [14]. Bagi perancang aplikasi kecil, pelajarannya sederhana tetapi nyata: ikuti kontrol dan gestur platform, karena pengguna sudah mempelajarinya; periksa keterbacaan gaya-gaya baru yang tembus pandang dan penuh warna, karena kaca yang indah dapat menurunkan kontras (Bab 11 mengukur kontras).

Komputasi spasial (*spatial computing*) membawa manipulasi langsung ke tiga dimensi. Di visionOS, pengguna *menatap* kontrol untuk membidiknya dan *mencubit* ibu jari dengan jari lain untuk mengaktifkannya; tangan boleh beristirahat di pangkuan [15]. Inilah hukum Fitts dengan mata sebagai penunjuk, dan pedoman Apple karenanya meminta bidang target dan jarak antartarget yang lapang di ruang. Bab 15 mempelajari antarmuka spasial, *wearable*, dan *ambient* secara mendalam; untuk saat ini cukuplah melihat bahwa dua jurang yang sama dan tawar-menawar yang sama antara mengingat dan mengenali berlaku pula di sana.

9.4 Metode Praktis: Memilih dan Memeriksa Gaya

Ketika merancang fitur baru, pilih gayanya dengan sengaja, lalu periksa. **Tabel 9.3** adalah lembar kerja satu halaman yang melakukan keduanya; isilah untuk setiap tugas penting.

Tabel 9.3 Lembar kerja untuk memilih dan memeriksa gaya interaksi, diisi untuk kios warung.

Langkah	Pertanyaan	Kios Bu Tini
1. Pengguna	Siapa yang mengerjakan tugas, seberapa sering, dengan keterampilan apa?	Mahasiswa (setiap hari), pengemudi (terburu-buru), pelanggan lansia (jarang menyentuh layar)
2. Konteks	Tangan, mata, kebisingan, cahaya, privasi?	Satu tangan sering sibuk; bising di siang hari; jari berminyak
3. Bentuk tugas	Pilihan sedikit atau banyak? Konkret atau abstrak?	Lima menu, konkret: cocok untuk manipulasi langsung
4. Gaya utama	Gaya mana yang paling cocok dengan langkah 1–3?	Kios sentuh dengan tombol bergambar yang besar
5. Jalur kedua	Alternatif mana yang melayani pengguna yang gagal dilayani gaya utama?	Pesanan lisan atau ketikan dengan konfirmasi; pembantu di meja kasir
6. Umpan balik	Apa yang dilihat pengguna dalam 0,1 dtk, 1 dtk, 10 dtk?	Baris pesanan langsung diperbarui; total selalu terlihat
7. Kesalahan	Bagaimana kekeliruan terlihat dan dibatalkan?	Tombol minus per baris; konfirmasi eksplisit saat ragu
8. Periksa	Ukuran target, alternatif gestur, kontras sudah diukur?	Audit setiap target pada 24, 44, dan 48 px (Subbab 9.5.4)

Langkah 1 sampai 3 berasal dari riset pengguna di Bab 5; langkah 8 adalah tempat program bab ini membantu, karena ukuran target dapat diukur secara otomatis di peramban sungguhan.

9.5 Membangunnya Langkah demi Langkah

Program ini memiliki satu **inti pesanan** dan tiga antarmuka, seperti diperlihatkan **Gambar 9.1**. Inti mengenal menu, menyimpan keranjang, menghitung total, dan menafsirkan dua jenis masukan: perintah yang ketat dan kata-kata bebas. Antarmuka hanya menerjemahkan antara manusia dan inti. Pemisahan ini adalah keputusan desain terpenting dalam bab ini: apa pun gayanya, harga, aturan, dan totalnya sama, dan gaya baru — pengenal ucapan sungguhan, bot WhatsApp — dapat ditambahkan tanpa menyentuh inti.



Gambar 9.1 Satu inti pesanan dengan tiga antarmuka: gaya perintah, gaya suara yang diketik beserta aturan konfirmasinya, dan kios manipulasi langsung semuanya memakai menu, keranjang, dan total yang sama.

9.5.1 Menu dan inti pesanan

Menunya berupa berkas CSV kecil, **Kode 9.1**. Selain kode, dua nama, dan harga, setiap menu mencantumkan *alias* yang dipakai pelanggan untuknya — kosakata gaya suara:

Kode 9.1 `menu.csv`: lima menu, harganya, dan kata-kata yang dipakai pelanggan untuk menyebutnya.

```
code,name_en,name_id,price,aliases
esteh,Iced tea,Es teh,5000,es teh|teh|iced tea|tea
kopi,Coffee,Kopi,6000,kopi|coffee
nasgor,Fried rice,Nasi goreng,15000,nasi goreng|nasgor|fried rice
mie,Fried noodles,Mi goreng,13000,mi goreng|mie goreng|mie|noodles
tempe,Fried tempeh,Tempe goreng,2000,tempe goreng|tempe|tempeh
```

Inti Python dalam **Kode 9.2** terdiri atas tiga bagian. `load_menu` membaca CSV. Kelas `Cart` menyimpan jumlah per menu dan menolak menu yang tak dikenal serta jumlah di luar 1 sampai 20. Dua pengurai mengubah masukan menjadi tindakan: `parse_command` menerima sintaksis ketat `verb code [quantity]`, dan `parse_utterance` membaca kata-kata bebas. Pengurai ucapan memindai kata dari kiri ke kanan; pada setiap posisi ia mencari alias *terpanjang* yang dimulai di sana (sehingga “nasi goreng” mengalahkan alias yang lebih pendek), mengambil kata bilangan tepat sebelum atau tepat sesudah menu sebagai jumlahnya, dan menandai setiap kata sebagai dipahami atau tidak. Kata seperti “*batal*” atau “cancel” mengubah niat menjadi menghapus; kata pengisi seperti “*tolong*” dan “*ya*” dipahami tetapi tidak membawa makna. Keyakinan menurut Persamaan (9.3) adalah bagian kata yang dipahami.

Kode 9.2 `order_core.py`: menu, keranjang, dan dua pengurai untuk gaya perintah dan gaya suara.

```
"""Inti pesanan warung, dipakai bersama oleh antarmuka perintah, suara, dan kios."""
import csv
from pathlib import Path

DATA = Path(__file__).resolve().parent.parent / "data"
NUMBERS = {"satu": 1, "dua": 2, "tiga": 3, "empat": 4, "lima": 5,
           "one": 1, "two": 2, "three": 3, "four": 4, "five": 5}
ADD = {"pesan", "tambah", "mau", "minta", "order", "add", "want"}
REMOVE = {"hapus", "batal", "kurangi", "remove", "cancel"}
TOTAL = {"berapa", "total", "much"}
FILLER = {"tolong", "saya", "aku", "ya", "dong", "sama", "dan", "yang",
          "please", "i", "a", "and", "the", "me", "give", "how"}
CONFIRM_BELOW = 0.6


def load_menu(path=DATA / "menu.csv"):
    with open(path, encoding="utf-8") as f:
        rows = list(csv.DictReader(f))
    for r in rows:
        r["price"] = int(r["price"])
        r["aliases"] = r["aliases"].split("|")
    return {r["code"]: r for r in rows}


class Cart:
    def __init__(self, menu):
        self.menu = menu
        self.qty = {}

    def add(self, code, n=1):
        if code not in self.menu:
            raise ValueError("unknown item: " + code)
        if not 1 <= n <= 20:
            raise ValueError("quantity must be 1..20")
        self.qty[code] = self.qty.get(code, 0) + n

    def remove(self, code, n=1):
        left = self.qty.get(code, 0) - n
        if left > 0:
            self.qty[code] = left
        else:
            self.qty.pop(code, None)
```

Kode 9.2 (Lanjutan) *order_core.py: menu, keranjang, dan dua pengurai untuk gaya perintah dan gaya suara.*

```
    def lines(self, lang="en"):
        out = []
        for code, n in self.qty.items():
            item = self.menu[code]
            out.append((code, item["name_" + lang], n, n * item["price"]))
        return out

    def total(self):
        return sum(n * self.menu[c]["price"] for c, n in self.qty.items())


def rupiah(x):
    return "Rp" + format(x, ",").replace(",", ".")


def parse_command(line):
    """Gaya perintah: 'add esteh 2', 'remove kopi', 'show'."""
    parts = line.split()
    if not parts:
        raise ValueError("empty command")
    verb, args = parts[0], parts[1:]
    if verb in ("add", "remove") and 1 <= len(args) <= 2:
        n = 1
        if len(args) == 2:
            if not args[1].isdigit():
                raise ValueError("quantity must be a number: " + args[1])
            n = int(args[1])
        return verb, args[0], n
    if verb in ("show", "total") and not args:
        return verb, None, 0
    raise ValueError("unknown command: " + line)


def parse_utterance(text, menu):
    """Gaya suara: kata bebas -> (maksud, [(kode, n)], keyakinan)."""
    words = text.lower().replace(",", " ").split()
    known = [False] * len(words)
    intent = "add"
    items = []
    i = 0
    while i < len(words):
        match = None
        for code, item in menu.items():
            for alias in item["aliases"]:
                a = alias.split()
                if words[i:i + len(a)] == a and (
```

Kode 9.2 (Lanjutan) *order_core.py: menu, keranjang, dan dua pengurai untuk gaya perintah dan gaya suara.*

```
                    match is None or len(a) > len(match[1])):
                match = (code, a)
        if match:
            n, j = 1, i + len(match[1])
            if i > 0 and words[i - 1] in NUMBERS:
                n = NUMBERS[words[i - 1]]
            elif j < len(words) and words[j] in NUMBERS:
                n = NUMBERS[words[j]]
                known[j] = True
                j += 1
            items.append((match[0], n))
            for k in range(i, i + len(match[1])):
                known[k] = True
            i = j
            continue
        w = words[i]
        if w in REMOVE:
            intent = "remove"
        elif w in TOTAL:
            intent = "total"
        if w in ADD | REMOVE | TOTAL | FILLER or w in NUMBERS:
            known[i] = True
        i += 1
    conf = sum(known) / len(known) if known else 0.0
    return intent, items, round(conf, 2)
```

Inti Java dalam **Kode 9.3** memiliki rancangan yang sama. *Record* Java menampung butir menu, baris keranjang, dan sebuah tafsiran (`Heard`); keranjang memakai `LinkedHashMap` agar baris-baris tetap dalam urutan penambahannya, persis seperti kamus Python.

Kode 9.3 `OrderCore.java`: menu, keranjang, dan pengurai 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.*;

/** Inti pesanan warung, dipakai bersama oleh antarmuka perintah, suara, kios. */
public class OrderCore {
    static final Path DATA = Path.of("..", "data");
    static final Map<String, Integer> NUMBERS = Map.of("satu", 1, "dua", 2,
        "tiga", 3, "empat", 4, "lima", 5, "one", 1, "two", 2, "three", 3,
        "four", 4, "five", 5);
    static final Set<String> ADD = Set.of("pesan", "tambah", "mau", "minta",
        "order", "add", "want");
    static final Set<String> REMOVE = Set.of("hapus", "batal", "kurangi",
        "remove", "cancel");
    static final Set<String> TOTAL = Set.of("berapa", "total", "much");
    static final Set<String> FILLER = Set.of("tolong", "saya", "aku", "ya",
        "dong", "sama", "dan", "yang", "please", "i", "a", "and", "the", "me",
        "give", "how");
    static final double CONFIRM_BELOW = 0.6;

    record Item(String code, String nameEn, String nameId, int price,
                List<String> aliases) {
        String name(String lang) { return lang.equals("id") ? nameId : nameEn; }
    }

    record Line(String code, String name, int qty, int subtotal) {}

    record Heard(String intent, List<Map.Entry<String, Integer>> items,
                 double confidence) {}

    static Map<String, Item> loadMenu() throws IOException {
        Map<String, Item> menu = new LinkedHashMap<>();
        List<String> rows = Files.readAllLines(DATA.resolve("menu.csv"),
                                               StandardCharsets.UTF_8);
        for (String row : rows.subList(1, rows.size())) {
            String[] f = row.split(",");
            menu.put(f[0], new Item(f[0], f[1], f[2], Integer.parseInt(f[3]),
                                    List.of(f[4].split("\\|"))));
        }
        return menu;
    }
```

Kode 9.3 (Lanjutan) *OrderCore.java: menu, keranjang, dan pengurai yang sama dalam Java.*

```
    static class Cart {
        final Map<String, Item> menu;
        final Map<String, Integer> qty = new LinkedHashMap<>();

        Cart(Map<String, Item> menu) { this.menu = menu; }

        void add(String code, int n) {
            if (!menu.containsKey(code))
                throw new IllegalArgumentException("unknown item: " + code);
            if (n < 1 || n > 20)
                throw new IllegalArgumentException("quantity must be 1..20");
            qty.merge(code, n, Integer::sum);
        }

        void remove(String code, int n) {
            int left = qty.getOrDefault(code, 0) - n;
            if (left > 0) qty.put(code, left); else qty.remove(code);
        }

        List<Line> lines(String lang) {
            List<Line> out = new ArrayList<>();
            qty.forEach((c, n) -> out.add(new Line(c, menu.get(c).name(lang), n,
                                                   n * menu.get(c).price())));
            return out;
        }

        int total() {
            int t = 0;
            for (var e : qty.entrySet())
                t += e.getValue() * menu.get(e.getKey()).price();
            return t;
        }
    }

    static String rupiah(int x) {
        return "Rp" + String.format(Locale.US, "%,d", x).replace(',', '.');
    }
```

Kode 9.3 (Lanjutan) *OrderCore.java: menu, keranjang, dan pengurai yang sama dalam Java.*

```
    /** Gaya perintah: "add esteh 2", "remove kopi", "show". */
    static String[] parseCommand(String line) {
        String[] p = line.trim().split("\\s+");
        String verb = p[0];
        if ((verb.equals("add") || verb.equals("remove"))
                && p.length >= 2 && p.length <= 3) {
            String n = "1";
            if (p.length == 3) {
                if (!p[2].matches("\\d+"))
                    throw new IllegalArgumentException(
                        "quantity must be a number: " + p[2]);
                n = p[2];
            }
            return new String[] {verb, p[1], n};
        }
        if ((verb.equals("show") || verb.equals("total")) && p.length == 1)
            return new String[] {verb, null, "0"};
        throw new IllegalArgumentException("unknown command: " + line);
    }

    /** Gaya suara: kata bebas -> maksud, item, dan keyakinan. */
    static Heard parseUtterance(String text, Map<String, Item> menu) {
        String[] w = text.toLowerCase().replace(",", " ").trim().split("\\s+");
        boolean[] known = new boolean[w.length];
        String intent = "add";
        List<Map.Entry<String, Integer>> items = new ArrayList<>();
        int i = 0;
        while (i < w.length) {
            String code = null;
            int len = 0;
            for (Item item : menu.values())
                for (String alias : item.aliases()) {
                    String[] a = alias.split(" ");
                    if (i + a.length <= w.length && a.length > len
                        && Arrays.equals(Arrays.copyOfRange(w, i, i + a.length),
                                         a)) {
                        code = item.code();
                        len = a.length;
                    }
                }
            if (code != null) {
                int n = 1, j = i + len;
                if (i > 0 && NUMBERS.containsKey(w[i - 1])) {
                    n = NUMBERS.get(w[i - 1]);
                } else if (j < w.length && NUMBERS.containsKey(w[j])) {
                    n = NUMBERS.get(w[j]);
```

Kode 9.3 (Lanjutan) *OrderCore.java: menu, keranjang, dan pengurai yang sama dalam Java.*

```
                    known[j] = true;
                    j++;
                }
                items.add(Map.entry(code, n));
                for (int k = i; k < i + len; k++) known[k] = true;
                i = j;
                continue;
            }
            if (REMOVE.contains(w[i])) intent = "remove";
            else if (TOTAL.contains(w[i])) intent = "total";
            if (ADD.contains(w[i]) || REMOVE.contains(w[i])
                || TOTAL.contains(w[i]) || FILLER.contains(w[i])
                || NUMBERS.containsKey(w[i]))
                known[i] = true;
            i++;
        }
        int k = 0;
        for (boolean b : known) if (b) k++;
        double conf = Math.round(100.0 * k / w.length) / 100.0;
        return new Heard(intent, items, conf);
    }
}
```

9.5.2 Baris perintah dan baris suara yang diketik

Antarmuka pertama membaca berkas sesi baris demi baris. Baris yang diawali `say` diperlakukan sebagai ucapan (diketik, agar program berjalan di mana saja dan keluarannya dapat diperiksa); setiap baris lain adalah perintah. Bagian yang menarik adalah keadaan dialog `pending`. Ketika sebuah ucapan menyebut menu tetapi keyakinannya di bawah `CONFIRM_BELOW` (0,6), antarmuka tidak bertindak: ia menyimpan tafsirannya dan bertanya “Did you mean …? Say ‘yes’ to confirm.” Jika ucapan berikutnya diawali *ya*, *iya*, atau *yes*, pesanan yang tersimpan dijalankan; ucapan lain membuangnya. Ketika tidak ada menu yang ditemukan sama sekali, antarmuka meminta nama menu. **Kode 9.4** dan **Kode 9.5** memperlihatkan kedua versinya.

Kode 9.4 `cli.py`: gaya perintah dan gaya suara yang diketik, dengan konfirmasi eksplisit di bawah ambang keyakinan.

```
"""Antarmuka gaya perintah dan gaya suara (diketik) untuk pesanan warung."""
import sys
from order_core import (CONFIRM_BELOW, Cart, load_menu, parse_command,
                        parse_utterance, rupiah)


def run(lines):
    menu = load_menu()
    cart = Cart(menu)
    pending = None  # pesanan berkeyakinan rendah, menunggu "ya"
    for line in lines:
        line = line.strip()
        if not line:
            continue
        print(">", line)
        if line.startswith("say "):
            words = line[4:].lower().split()
            if pending and words and words[0] in ("ya", "yes", "iya"):
                intent, items = pending
                pending = None
            else:
                pending = None
                intent, items, conf = parse_utterance(line[4:], menu)
                heard = ", ".join(f"{n} x {menu[c]['name_en']}"
                                  for c, n in items)
                print(f"  heard: {intent} [{heard or 'nothing'}] "
                      f"confidence {conf}")
                if not items:
                    print("  -> Sorry, which menu item? Please say its name.")
                    continue
                if conf < CONFIRM_BELOW:
                    # Keyakinan rendah: tanya dulu sebelum bertindak.
                    pending = (intent, items)
                    print(f"  -> Did you mean {heard}? Say 'yes' to confirm.")
                    continue
            for c, n in items:
                if intent == "remove":
                    cart.remove(c, n)
                else:
                    cart.add(c, n)
            print(f"  -> OK. Total {rupiah(cart.total())}")
            continue
        try:
            verb, code, n = parse_command(line)
            if verb == "add":
                cart.add(code, n)
```

Kode 9.4 (Lanjutan) *cli.py: gaya perintah dan gaya suara yang diketik, dengan konfirmasi eksplisit di bawah ambang keyakinan.*

```
            elif verb == "remove":
                cart.remove(code, n)
            else:
                for _, name, q, sub in cart.lines():
                    print(f"  {q:>2} x {name:<14}{rupiah(sub):>10}")
            print(f"  total {rupiah(cart.total())}")
        except ValueError as e:
            print("  error:", e)


if __name__ == "__main__":
    src = open(sys.argv[1], encoding="utf-8") if len(sys.argv) > 1 else sys.stdin
    run(src)
```

Kode 9.5 `Cli.java`: antarmuka perintah dan suara yang sama dalam Java.

```
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.*;

/** Antarmuka gaya perintah dan gaya suara (diketik) untuk pesanan warung. */
public class Cli {
    public static void main(String[] args) throws Exception {
        var menu = OrderCore.loadMenu();
        var cart = new OrderCore.Cart(menu);
        OrderCore.Heard pending = null; // pesanan berkeyakinan rendah, tunggu "ya"
        for (String raw : Files.readAllLines(Path.of(args[0]),
                                             StandardCharsets.UTF_8)) {
            String line = raw.strip();
            if (line.isEmpty()) continue;
            System.out.println("> " + line);
            if (line.startsWith("say ")) {
                String first = line.substring(4).toLowerCase().split("\\s+")[0];
                OrderCore.Heard h;
                if (pending != null
                        && List.of("ya", "yes", "iya").contains(first)) {
                    h = pending;
                    pending = null;
                } else {
                    pending = null;
                    h = OrderCore.parseUtterance(line.substring(4), menu);
                    List<String> parts = new ArrayList<>();
                    for (var e : h.items())
                        parts.add(e.getValue() + " x "
                                + menu.get(e.getKey()).nameEn());
                    String heard = String.join(", ", parts);
                    System.out.println("  heard: " + h.intent() + " ["
                        + (heard.isEmpty() ? "nothing" : heard) + "] confidence "
                        + h.confidence());
                    if (h.items().isEmpty()) {
                        System.out.println(
                            "  -> Sorry, which menu item? Please say its name.");
                        continue;
                    }
                    if (h.confidence() < OrderCore.CONFIRM_BELOW) {
                        // Keyakinan rendah: tanya dulu sebelum bertindak.
                        pending = h;
                        System.out.println("  -> Did you mean " + heard
                            + "? Say 'yes' to confirm.");
                        continue;
                    }
```

Kode 9.5 (Lanjutan) *Cli.java: antarmuka perintah dan suara yang sama dalam Java.*

```
                }
                for (var e : h.items()) {
                    if (h.intent().equals("remove"))
                        cart.remove(e.getKey(), e.getValue());
                    else
                        cart.add(e.getKey(), e.getValue());
                }
                System.out.println("  -> OK. Total "
                                   + OrderCore.rupiah(cart.total()));
                continue;
            }
            try {
                String[] c = OrderCore.parseCommand(line);
                int n = Integer.parseInt(c[2]);
                if (c[0].equals("add")) cart.add(c[1], n);
                else if (c[0].equals("remove")) cart.remove(c[1], n);
                else for (var l : cart.lines("en"))
                    System.out.printf("  %2d x %-14s%10s%n", l.qty(), l.name(),
                                      OrderCore.rupiah(l.subtotal()));
                System.out.println("  total " + OrderCore.rupiah(cart.total()));
            } catch (IllegalArgumentException e) {
                System.out.println("  error: " + e.getMessage());
            }
        }
    }
}
```

Berkas sesi mencampur kedua gaya dan memuat tiga masalah yang disengaja: perintah dengan jumlah yang salah, ucapan tanpa menu di dalamnya, dan ucapan yang menunya jelas tetapi kata-kata lainnya tidak.

Kode 9.6 `session.txt`: sesi tertulis berisi perintah, ucapan, dan tiga masalah yang disengaja.

```
add esteh 2
add nasgor
add kopi x
say tolong nasi goreng dua sama es teh satu ya
say minta yang anget anget dong
say kopi item yang kentel banget
say ya
remove esteh 1
show
```

9.5.3 Kios sentuh

Kios adalah halaman web dengan satu tombol besar per menu dan satu tombol minus per baris pesanan. Setiap tombol adalah formulir HTML mungil yang mengirim ke `/add`, `/remove`, atau `/pay`; peladen mengubah keranjang lalu mengarahkan kembali ke halaman (pola *post/redirect/get*, yang mencegah muat ulang memesan dua kali). Karena berupa HTML biasa tanpa skrip, kios ini berjalan di ponsel termurah, dengan papan ketik, dan dengan pembaca layar. Templat halaman, yang dipakai bersama kedua peladen, menetapkan ukuran-ukuran yang penting: tombol menu setinggi sekurang-kurangnya 64 piksel, tombol minus 48 kali 48, tombol bayar setinggi 56, dan cincin fokus berwarna amber yang terlihat bagi pengguna papan ketik. Versi pertama, **Kode 9.7**, juga memiliki tautan bahasa yang kecil di bagian bawah, digambar pada 70% ukuran teks — jenis target yang sangat umum di situs sungguhan, dan cacat yang disengaja yang akan ditemukan oleh audit.

Kode 9.7 `kiosk_v1.html`: versi pertama templat halaman kios, dengan tempat-tempat isian yang diisi oleh peladen.

```
<!DOCTYPE html>
<html lang="{{LANG}}">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>{{TITLE}}</title>
<style>
body { font-family: "DejaVu Sans", sans-serif; margin: 0; color: #1f2937;
       background: #f3f4f6; }
header { background: #12355B; color: #fff; padding: 12px 16px; }
h1 { font-size: 1.3rem; margin: 0; }
main { padding: 12px 16px; }
.menu { display: grid; grid-template-columns: 1fr 1fr; gap: 10px; }
.menu button { width: 100%; min-height: 64px;
       border: 2px solid #0F7C7C; border-radius: 12px; background: #fff; }
.menu button span { display: block; color: #0F7C7C; font-weight: bold; }
ul { list-style: none; padding: 0; }
li { display: flex; justify-content: space-between; align-items: center;
     background: #fff; margin: 6px 0; padding: 8px; border-radius: 8px; }
.minus { min-width: 48px; min-height: 48px; font-size: 1.4rem;
         border: 2px solid #C0392B; border-radius: 50%; background: #fff; }
.total { font-size: 1.2rem; font-weight: bold; }
.pay { width: 100%; min-height: 56px; font-size: 1.1rem; color: #fff;
       background: #0F7C7C; border: 0; border-radius: 12px; }
.tiny { font-size: 0.7rem; }
button { font: inherit; }
button:focus-visible { outline: 3px solid #E0A100; outline-offset: 2px; }
</style>
</head>
<body>
<header><h1>{{TITLE}}</h1></header>
<main>
<h2>{{MENU}}</h2>
<div class="menu">{{ITEMS}}</div>
<h2>{{ORDER}}</h2>
<ul>{{LINES}}</ul>
<p class="total">{{TOTAL}}</p>
<form method="post" action="/pay"><input type="hidden" name="lang" value="{{LANG}}">
<button class="pay" type="submit">{{PAY}}</button></form>
<p><a class="tiny" href="/?lang={{OTHER}}">{{SWITCH}}</a></p>
</main>
</body>
</html>
```

Peladen Flask dalam **Kode 9.8** mengisi templat: ia membangun tombol menu dan baris pesanan, meloloskan (*escape*) setiap nama, dan menukar teks bahasa Inggris dan Indonesia melalui parameter `lang`. Setiap tombol minus membawa `aria-label` seperti “Kurangi satu: Es teh”, sehingga pembaca layar mengumumkan lebih dari sekadar “minus”.

Kode 9.8 `kiosk.py`: kios manipulasi langsung yang dilayani Flask pada porta 4090.

```
"""Kios manipulasi langsung: ketuk hidangan untuk tambah, minus untuk kurangi."""
import sys
from html import escape
from flask import Flask, redirect, request
from order_core import DATA, Cart, load_menu, rupiah

TEXT = {"en": {"TITLE": "Warung Bu Tini", "MENU": "Tap to add",
               "ORDER": "Your order", "PAY": "Pay", "TOTAL": "Total",
               "SWITCH": "Bahasa Indonesia", "EMPTY": "Nothing yet",
               "LESS": "Remove one"},
        "id": {"TITLE": "Warung Bu Tini", "MENU": "Ketuk untuk menambah",
               "ORDER": "Pesanan Anda", "PAY": "Bayar", "TOTAL": "Total",
               "SWITCH": "English", "EMPTY": "Belum ada pesanan",
               "LESS": "Kurangi satu"}}
app = Flask(__name__)
menu = load_menu()
cart = Cart(menu)
name = sys.argv[1] if len(sys.argv) > 1 else "kiosk.html"
template = (DATA / name).read_text(encoding="utf-8")


def page(lang):
    t = TEXT[lang]
    items = "".join(
        f'<form method="post" action="/add"><input type="hidden" name="lang" '
        f'value="{lang}"><button name="code" value="{c}">'
        f'{escape(m["name_" + lang])}<span>{rupiah(m["price"])}</span>'
        f"</button></form>" for c, m in menu.items())
    lines = "".join(
        f'<li><span>{n} x {escape(name)} = {rupiah(sub)}</span>'
        f'<form method="post" action="/remove"><input type="hidden" '
        f'name="lang" value="{lang}"><button class="minus" name="code" '
        f'value="{c}" aria-label="{t["LESS"]}: {escape(name)}">-</button>'
        f"</form></li>" for c, name, n, sub in cart.lines(lang))
    html = template
    for key, val in {**t, "LANG": lang, "OTHER": "en" if lang == "id" else "id",
                     "ITEMS": items, "LINES": lines or f"<li>{t['EMPTY']}</li>",
                     "TOTAL": f"{t['TOTAL']}: {rupiah(cart.total())}"}.items():
        html = html.replace("{{" + key + "}}", val)
    return html


def lang_of(values):
    return "id" if values.get("lang") == "id" else "en"
```

Kode 9.8 (Lanjutan) *kiosk.py: kios manipulasi langsung yang dilayani Flask pada porta 4090.*

```
@app.get("/")
def home():
    return page(lang_of(request.args))


@app.post("/add")
def add():
    cart.add(request.form["code"])
    return redirect("/?lang=" + lang_of(request.form), code=303)


@app.post("/remove")
def remove():
    cart.remove(request.form["code"])
    return redirect("/?lang=" + lang_of(request.form), code=303)


@app.post("/pay")
def pay():
    cart.qty.clear()
    return redirect("/?lang=" + lang_of(request.form), code=303)


if __name__ == "__main__":
    app.run(port=4090)
```

Peladen Java dalam **Kode 9.9** hanya memakai pustaka standar: `com.sun.net.httpserver.HttpServer`, pengurai formulir kecil, dan templat yang sama. Ia mendengarkan pada porta 4091, dan untuk keranjang yang sama ia menghasilkan halaman yang sama persis, bita demi bita, dengan peladen Python — fakta yang diandalkan pengujian di Subbab 9.7.

Kode 9.9 `Kiosk.java`: kios yang sama dengan peladen HTTP bawaan JDK pada porta 4091.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.util.*;

/** Kios manipulasi langsung: ketuk hidangan untuk tambah, minus untuk kurangi. */
public class Kiosk {
    static final Map<String, Map<String, String>> TEXT = Map.of(
        "en", Map.of("TITLE", "Warung Bu Tini", "MENU", "Tap to add",
            "ORDER", "Your order", "PAY", "Pay", "TOTAL", "Total",
            "SWITCH", "Bahasa Indonesia", "EMPTY", "Nothing yet",
            "LESS", "Remove one"),
        "id", Map.of("TITLE", "Warung Bu Tini", "MENU", "Ketuk untuk menambah",
            "ORDER", "Pesanan Anda", "PAY", "Bayar", "TOTAL", "Total",
            "SWITCH", "English", "EMPTY", "Belum ada pesanan",
            "LESS", "Kurangi satu"));
    static Map<String, OrderCore.Item> menu;
    static OrderCore.Cart cart;
    static String template;

    static String esc(String s) {
        return s.replace("&", "&amp;").replace("<", "&lt;")
                .replace(">", "&gt;").replace("\"", "&quot;");
    }

    static String page(String lang) {
        var t = TEXT.get(lang);
        StringBuilder items = new StringBuilder(), lines = new StringBuilder();
        for (var m : menu.values())
            items.append("<form method=\"post\" action=\"/add\">"
                + "<input type=\"hidden\" name=\"lang\" value=\"" + lang
                + "\"><button name=\"code\" value=\""
                + m.code() + "\">" + esc(m.name(lang)) + "<span>"
                + OrderCore.rupiah(m.price()) + "</span></button></form>");
        for (var l : cart.lines(lang))
            lines.append("<li><span>" + l.qty() + " x " + esc(l.name()) + " = "
                + OrderCore.rupiah(l.subtotal()) + "</span><form method=\"post\" "
                + "action=\"/remove\"><input type=\"hidden\" name=\"lang\" value=\""
                + lang + "\"><button class=\"minus\" name=\"code\" value=\""
                + l.code() + "\" aria-label=\"" + t.get("LESS") + ": "
                + esc(l.name()) + "\">-</button></form></li>");
        Map<String, String> v = new HashMap<>(t);
```

Kode 9.9 (Lanjutan) *Kiosk.java: kios yang sama dengan peladen HTTP bawaan JDK pada porta 4091.*

```
        v.put("LANG", lang);
        v.put("OTHER", lang.equals("id") ? "en" : "id");
        v.put("ITEMS", items.toString());
        v.put("LINES", lines.length() > 0 ? lines.toString()
                                          : "<li>" + t.get("EMPTY") + "</li>");
        v.put("TOTAL", t.get("TOTAL") + ": " + OrderCore.rupiah(cart.total()));
        String html = template;
        for (var e : v.entrySet())
            html = html.replace("{{" + e.getKey() + "}}", e.getValue());
        return html;
    }

    static Map<String, String> form(String body) {
        Map<String, String> f = new HashMap<>();
        if (body == null || body.isEmpty()) return f;
        for (String pair : body.split("&")) {
            String[] kv = pair.split("=", 2);
            f.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
                  kv.length > 1 ? URLDecoder.decode(kv[1], StandardCharsets.UTF_8)
                                : "");
        }
        return f;
    }

    static void handle(HttpExchange ex) throws IOException {
        String path = ex.getRequestURI().getPath();
        if (ex.getRequestMethod().equals("POST")) {
            var f = form(new String(ex.getRequestBody().readAllBytes(),
                                    StandardCharsets.UTF_8));
            String lang = "id".equals(f.get("lang")) ? "id" : "en";
            switch (path) {
                case "/add" -> cart.add(f.get("code"), 1);
                case "/remove" -> cart.remove(f.get("code"), 1);
                case "/pay" -> cart.qty.clear();
                default -> { }
            }
            ex.getResponseHeaders().set("Location", "/?lang=" + lang);
            ex.sendResponseHeaders(303, -1);  // kembali ke halaman sesudah POST
            ex.close();
            return;
        }
        var q = form(ex.getRequestURI().getRawQuery());
        byte[] body = page("id".equals(q.get("lang")) ? "id" : "en")
                          .getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", "text/html; charset=utf-8");
        ex.sendResponseHeaders(200, body.length);
```

Kode 9.9 (Lanjutan) *Kiosk.java: kios yang sama dengan peladen HTTP bawaan JDK pada porta 4091.*

```
        ex.getResponseBody().write(body);
        ex.close();
    }

    public static void main(String[] args) throws IOException {
        menu = OrderCore.loadMenu();
        cart = new OrderCore.Cart(menu);
        String name = args.length > 0 ? args[0] : "kiosk.html";
        template = Files.readString(OrderCore.DATA.resolve(name));
        HttpServer server = HttpServer.create(new InetSocketAddress(4091), 0);
        server.createContext("/", Kiosk::handle);
        server.start();
        System.out.println("Kiosk on http://127.0.0.1:4091/");
    }
}
```

9.5.4 Mengukur setiap target di peramban sungguhan

Ukuran target harus diukur, bukan ditebak, karena CSS, fon, dan lebar ponsel menentukan ukuran yang benar-benar ditemui jari. Program audit dalam **Kode 9.10** memakai Playwright untuk membuka kios di peramban Chromium sungguhan dengan *viewport* selebar 360 piksel (lebar Android yang umum), mengetuk “Iced tea” dua kali dan “Fried rice” sekali seperti pembeli, lalu meminta peramban memberikan persegi panjang setiap tautan, tombol, dan ruas formulir. Ia mencetak lebar dan tinggi setiap target serta apakah sisi terpendeknya di bawah 24, 44, atau 48 piksel, dan dapat menyimpan tangkapan layar satu halaman penuh. Program audit hanya ada satu, ditulis dalam Python, karena ia mengendalikan peramban; ia mengaudit peladen Java dan Python sama baiknya.

Kode 9.10 `audit_targets.py`: mengukur setiap elemen kios yang sedang berjalan yang dapat diketuk pada lebar ponsel dan membandingkannya dengan tiga pedoman.

```
"""Ukur setiap elemen yang dapat diketuk terhadap pedoman ukuran target."""
import sys
from playwright.sync_api import sync_playwright

RULES = [("WCAG 2.5.8 (AA)", 24), ("WCAG 2.5.5 / Apple", 44),
         ("Material", 48)]
JS = """() => [...document.querySelectorAll('a, button, input, select')]
  .filter(e => e.type !== 'hidden')
  .map(e => { const r = e.getBoundingClientRect();
    return [e.tagName.toLowerCase(),
            (e.getAttribute('aria-label') || e.innerText || '').trim(),
            Math.round(r.width), Math.round(r.height)]; })"""


def audit(url, shot=None, actions=()):
    with sync_playwright() as p:
        browser = p.chromium.launch()
        pg = browser.new_page(viewport={"width": 360, "height": 740},
                              device_scale_factor=2)
        pg.goto(url)
        for selector in actions:  # ketuk beberapa tombol dulu, seperti pembeli
            with pg.expect_navigation():
                pg.click(selector)
        rows = pg.evaluate(JS)
        if shot:
            pg.screenshot(path=shot, full_page=True)
        browser.close()
    fails = {name: 0 for name, _ in RULES}
    for tag, label, w, h in rows:
        size = min(w, h)
        bad = [name for name, px in RULES if size < px]
        for name in bad:
            fails[name] += 1
        label = label.replace("\n", " ")[:22]
        print(f"{tag:<7}{label:<23}{w:>4} x {h:<4}"
              + ("below " + bad[0] if bad else "ok"))
    print(f"{len(rows)} targets; smaller than "
          + ", ".join(f"{px} px: {fails[n]}" for n, px in RULES))


if __name__ == "__main__":
    url = sys.argv[1] if len(sys.argv) > 1 else "http://127.0.0.1:4090/"
    audit(url, sys.argv[2] if len(sys.argv) > 2 else None,
          ['button[value="esteh"]', 'button[value="esteh"]',
           'button[value="nasgor"]'])
```

9.6 Menjalankannya

Jalankan sesi melalui antarmuka Python dengan `python cli.py ../data/session.txt`; antarmuka Java, `java -cp out Cli ../data/session.txt`, mencetak baris-baris yang persis sama:

```
> add esteh 2
  total Rp10.000
> add nasgor
  total Rp25.000
> add kopi x
  error: quantity must be a number: x
> say tolong nasi goreng dua sama es teh satu ya
  heard: add [2 x Fried rice, 1 x Iced tea] confidence 1.0
  -> OK. Total Rp60.000
> say minta yang anget anget dong
  heard: add [nothing] confidence 0.6
  -> Sorry, which menu item? Please say its name.
> say kopi item yang kentel banget
  heard: add [1 x Coffee] confidence 0.4
  -> Did you mean 1 x Coffee? Say 'yes' to confirm.
> say ya
  -> OK. Total Rp66.000
> remove esteh 1
  total Rp61.000
> show
   2 x Iced tea        Rp10.000
   3 x Fried rice      Rp45.000
   1 x Coffee           Rp6.000
  total Rp61.000
```

Bacalah transkrip itu sebagai percakapan. Dua perintah biasa berhasil dan menampilkan total berjalan setelah masing-masing. Perintah `add kopi x` ditolak dengan pesan yang menyebut masalahnya — kegagalan khas gaya perintah, yaitu salah sintaksis. Ucapan pertama, "*tolong nasi goreng dua sama es teh satu ya*", dipahami sepenuhnya: dua menu, jumlah sesudah menu, kata pengisi dikenali, keyakinan 1,0, sehingga pesanan dijalankan hanya dengan konfirmasi implisit (total yang baru). Ucapan kedua, "*minta yang anget anget dong*", tidak memuat menu; sistem tidak menebak, tetapi meminta nama menu. Ucapan ketiga, "*kopi item yang kentel banget*", menyebut kopi tetapi memuat dua kata yang tidak dikenal pengurai, sehingga keyakinannya $c = 2/5 = 0.4$ menurut Persamaan (9.3): di bawah ambang, sistem bertanya "Did you mean 1 x Coffee?", dan "*ya*" dari pelanggan menuntaskannya. "*Dua, ya?*" milik Bu Tini telah menjadi aturan dalam kode.

Sekarang jalankan versi pertama kios (`python kiosk.py kiosk_v1.html`, atau `java -cp out Kiosk kiosk_v1.html`) dan jalankan audit di terminal kedua:

```
button Iced tea Rp5.000        159 x 64  ok
button Coffee Rp6.000          159 x 64  ok
button Fried rice Rp15.000     159 x 64  ok
button Fried noodles Rp13.000  159 x 64  ok
button Fried tempeh Rp2.000    159 x 64  ok
button Remove one: Iced tea     48 x 48  ok
button Remove one: Fried rice   48 x 48  ok
button Pay                     328 x 56  ok
a      Bahasa Indonesia         99 x 13  below WCAG 2.5.8 (AA)
9 targets; smaller than 24 px: 1, 44 px: 1, 48 px: 1
```

Delapan dari sembilan target berukuran cukup besar. Tombol menu berukuran 159 kali 64 piksel — dua kolom di layar 360 piksel — dan tombol minus tepat 48 kali 48. Tautan bahasa berukuran 99 kali 13: di bawah ketiga ukuran. Apakah itu kegagalan WCAG? Belum tentu pada tingkat AA,

dan hal ini patut dijelaskan dengan cermat. Kriteria 2.5.8 memiliki pengecualian *jarak*: target yang terlalu kecil tetap lulus jika lingkaran berdiameter 24 piksel yang berpusat padanya tidak berpotongan dengan target lain atau lingkaran target lain [7]. Tautan itu berdiri sendiri di bagian bawah halaman, sehingga kemungkinan besar lulus 2.5.8 berkat jarak. Namun ia gagal pada kriteria 44 piksel 2.5.5 dan kedua pedoman platform, ia adalah satu-satunya target yang akan meleset bagi pelanggan lansia yang jarinya kurang tepat, dan menurut Contoh di Subbab 9.3.3 ia hampir dua bit lebih sulit dikenai daripada tombol menu. Ia harus diperbaiki.

9.7 Mengujinya

Inti pesanan layak mendapat pengujian tersendiri, karena setiap antarmuka mempercayainya. **Kode 9.11** memeriksa total, apa yang terjadi ketika yang dikurangi lebih banyak daripada yang dipesan, bilangan sebelum dan sesudah menu, niat membatalkan, kasus keyakinan rendah, dan jumlah yang salah; **Kode 9.12** memeriksa hal-hal yang sama dalam Java tanpa pustaka pengujian apa pun.

Kode 9.11 `test_order_core.py`: pengujian inti pesanan untuk pytest.

```
"""Pemeriksaan sederhana inti pesanan (jalankan: python -m pytest -q)."""
import pytest
from order_core import (CONFIRM_BELOW, Cart, load_menu, parse_command,
                        parse_utterance)

menu = load_menu()


def test_total_and_remove():
    cart = Cart(menu)
    cart.add("esteh", 2)
    cart.add("nasgor")
    assert cart.total() == 25000
    cart.remove("esteh", 5)
    assert "esteh" not in cart.qty


def test_numbers_before_and_after():
    assert parse_utterance("tolong nasi goreng dua", menu)[1] == [("nasgor", 2)]
    assert parse_utterance("tiga es teh", menu)[1] == [("esteh", 3)]


def test_intent_and_confidence():
    assert parse_utterance("batal kopi", menu)[0] == "remove"
    assert parse_utterance("kopi item yang kentel banget", menu)[2] < CONFIRM_BELOW


def test_bad_quantity():
    with pytest.raises(ValueError):
        parse_command("add kopi x")
```

Kode 9.12 `OrderCoreTest.java`: pemeriksaan yang sama dalam Java biasa.

```java
import java.util.*;

/** Pemeriksaan sederhana inti pesanan (jalankan: java -cp out OrderCoreTest). */
public class OrderCoreTest {
    static int failed = 0;

    static void check(boolean ok, String what) {
        System.out.println((ok ? "PASS " : "FAIL ") + what);
        if (!ok) failed++;
    }

    public static void main(String[] args) throws Exception {
        var menu = OrderCore.loadMenu();
        var cart = new OrderCore.Cart(menu);
        cart.add("esteh", 2);
        cart.add("nasgor", 1);
        check(cart.total() == 25000, "total of 2 iced tea + 1 fried rice");
        cart.remove("esteh", 5);
        check(!cart.qty.containsKey("esteh"), "removing too many deletes line");
        var h = OrderCore.parseUtterance("tolong nasi goreng dua", menu);
        check(h.items().get(0).getValue() == 2, "number after the dish");
        h = OrderCore.parseUtterance("tiga es teh", menu);
        check(h.items().get(0).getValue() == 3, "number before the dish");
        h = OrderCore.parseUtterance("batal kopi", menu);
        check(h.intent().equals("remove"), "cancel word gives remove intent");
        h = OrderCore.parseUtterance("kopi item yang kentel banget", menu);
        check(h.confidence() < OrderCore.CONFIRM_BELOW, "unclear words: confirm");
        try {
            OrderCore.parseCommand("add kopi x");
            check(false, "bad quantity rejected");
        } catch (IllegalArgumentException e) {
            check(true, "bad quantity rejected");
        }
        System.out.println(failed == 0 ? "all passed" : failed + " failed");
    }
}
```

`python -m pytest -q` melaporkan:

```
4 passed in 0.01s
```

dan `java -cp out OrderCoreTest` mencetak:

```
PASS total of 2 iced tea + 1 fried rice
PASS removing too many deletes line
PASS number after the dish
PASS number before the dish
PASS cancel word gives remove intent
PASS unclear words: confirm
PASS bad quantity rejected
all passed
```

Dua pemeriksaan tambahan layak dilakukan sekali dengan tangan. Ambil halaman dari kedua peladen dengan keranjang kosong dan bandingkan bitanya (`curl -s http://127.0.0.1:4090/?lang=id | md5sum`, dan hal yang sama untuk porta 4091): kedua *hash* sama, sehingga audit atas satu peladen berlaku pula untuk yang lain. Dan jalankan audit terhadap peladen Java: ia mencetak sembilan baris yang sama seperti terhadap Flask.

9.8 Membuatnya Lebih Baik

9.8.1 Target yang dapat dikenai semua orang

Perbaikan untuk tautan bahasa adalah memberinya bidang sentuh sungguhan tanpa memperbesar teksnya: tampilkan sebagai blok sebaris (*inline block*) dengan bantalan dan tinggi minimum 48 piksel. Templat akhir `kiosk.html` mengubah satu aturan CSS dan satu nama kelas:

```
.lang { display: inline-block; min-height: 48px; padding: 12px 16px;
        box-sizing: border-box; }
<p><a class="lang" href="/?lang={{OTHER}}">{{SWITCH}}</a></p>
```

Jalankan salah satu peladen tanpa argumen (kini ia memuat `kiosk.html`) dan jalankan audit lagi, kali ini sambil menyimpan tangkapan layar:

```
button Iced tea Rp5.000        159 x 64  ok
button Coffee Rp6.000          159 x 64  ok
button Fried rice Rp15.000     159 x 64  ok
button Fried noodles Rp13.000  159 x 64  ok
button Fried tempeh Rp2.000    159 x 64  ok
button Remove one: Iced tea     48 x 48  ok
button Remove one: Fried rice   48 x 48  ok
button Pay                     328 x 56  ok
a      Bahasa Indonesia        173 x 48  ok
9 targets; smaller than 24 px: 0, 44 px: 0, 48 px: 0
```

Kini setiap target memenuhi ketiga pedoman. **Gambar 9.2** adalah tangkapan layar kios yang sedang berjalan, dalam bahasa Indonesia, setelah ketiga ketukan.



Gambar 9.2 Kios akhir pada lebar 360 piksel CSS setelah mengetuk “Es teh” dua kali dan “Nasi goreng” sekali: dua kolom tombol menu setinggi 64 piksel, tombol minus 48 piksel pada setiap baris pesanan, dan tautan bahasa dengan bidang sentuh 48 piksel.

9.8.2 Bertanya pada saat yang tepat

Aturan konfirmasi di Subbab 9.5.2 sengaja dibuat sederhana, dan menjalankan lebih banyak ucapan melaluinya memperlihatkan batas-batasnya. "*es teh manis dua gelas*" menghasilkan satu es teh dengan keyakinan 0,6 — tepat pada ambang, sehingga dijalankan, dan hasilnya *salah*, karena kata *dua* terpisah dari menu oleh *manis*. "*berapa totalnya*" tidak dikenali sebagai pertanyaan tentang total, karena pengurai mengenal *total* tetapi tidak *totalnya*, kata yang sama dengan akhiran *-nya*. Kedua kegagalan itu mengajarkan hal yang umum. Skor keyakinan mengukur seberapa banyak yang dipahami, bukan apakah pemahamannya benar; dan antarmuka suara untuk pengguna Indonesia harus mampu menghadapi imbuhan serta campuran bahasa Indonesia, Jawa, dan Inggris yang benar-benar dipakai pelanggan. Soal Latihan 9.4 dan 9.5 memperbaiki keduanya. Untuk tindakan yang mahal — membayar, di kios sungguhan — aturan yang benar lebih ketat: selalu konfirmasi secara eksplisit, berapa pun keyakinannya.

9.9 Memakainya Sungguhan

Pola yang sama — satu inti, beberapa antarmuka, target yang diukur — dapat disesuaikan untuk banyak usaha kecil.

- **Menu Anda sendiri.** Ganti `menu.csv` dengan menu, harga, dan, yang terpenting, kata-kata yang benar-benar diucapkan pelanggan Anda. Dengarkan di meja kasir selama seminggu dan catat alias-aliasnya; daftar itulah antarmuka suara Anda.
- **Antarmuka tambahan.** Bot WhatsApp atau Telegram adalah antarmuka bahasa alami: berikan setiap pesan yang masuk kepada `parse_utterance` dan balas dengan teks konfirmasi. Pengenal ucapan sungguhan (di perangkat pada Android dan iOS, atau model sumber terbuka yang dijalankan lokal) adalah antarmuka lain; keluaran teksnya masuk ke pengurai yang sama, dan keyakinannya sendiri dapat digabungkan dengan Persamaan (9.3).
- **Pelanggan lansia.** Pertahankan pembantu di meja kasir sebagai sebuah antarmuka juga. Antarmuka yang memaksa semua orang memakai tablet telah memilih gaya yang *melawan* sebagian penggunanya; kios hendaknya memperpendek antrean, bukan menggantikan manusia.
- **Sebelum mencetak kode QR.** Jalankan audit pada lebar 320 dan 360 piksel, di ponsel termurah yang dapat Anda temukan, di bawah sinar matahari, dengan ibu jari. Pengukuran otomatis menangkap ukuran; hanya manusia yang menangkap segala hal lainnya.
- **Pembayaran.** Apa pun yang memindahkan uang memerlukan konfirmasi eksplisit, total yang jelas, dan jalan pembatalan hingga saat pembayaran. Bab 12 membahas mengapa menyembunyikan hal-hal ini, sekalipun tidak sengaja, menghancurkan kepercayaan.

9.10 IMK dalam Praktik: Profitina

Profitina, platform visibilitas AI (GEO) asal Indonesia yang mengukur seberapa siap situs web sebuah merek untuk dikutip oleh mesin jawaban AI, memakai beberapa gaya interaksi dalam satu produk, dan pilihan-pilihannya mengikuti penalaran bab ini. **Dasbor** dirancang sebagai kokpit: satu permintaan mengumpulkan notifikasi (balasan, suka, *repost*, pesan langsung, sematan, status umpan balik, pembelian), kanal yang diikuti, peringkat, dan aktivitas, dan setiap butir dapat diklik langsung menuju bagiannya. Itulah manipulasi langsung yang diterapkan pada aliran peristiwa: objek yang diminati terlihat, dan bertindak atasnya cukup dengan satu ketukan, bukan menelusuri menu.

Kotak tulis komunitas mengikuti batas 0,1 detik di Subbab 9.3.2 dengan *pengiriman optimistis*: ketika pengguna menekan kirim, kotak langsung dikosongkan dan kiriman langsung muncul,

sedangkan permintaan jaringan berlangsung sesudahnya; jika pengiriman gagal, teksnya dipulihkan sehingga tidak ada ketikan yang hilang. Pengguna melihat akibatnya dalam waktu yang terasa seketika, dan kegagalan yang jarang dapat dipulihkan. Tindakan **"Share to X"** sengaja dibuat manual dan per butir, dengan antrean dan kuota harian: tindakan yang berakibat publik tetap berada di bawah kendali manusia yang eksplisit, prinsip yang sama dengan konfirmasi eksplisit bagi perintah suara yang mahal. Dan **mesin konten**, yang menyusun draf konten GEO dengan model Qwen2.5 yang dijalankan secara lokal, menyajikan draf untuk disunting pengguna alih-alih langsung menerbitkannya: bahasa alami dipakai untuk *mengusulkan*, dan manipulasi langsung untuk *memutuskan*. Latihan praktik buku ini memakai studi kasus laundry, bukan produk itu sendiri; Bab 16 membangunnya.

9.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Satu gaya untuk semua orang.** Kios yang hanya sentuh menyingkirkan pelanggan yang tidak dapat atau tidak mau memakainya; layanan yang hanya suara menyingkirkan pengguna tuli dan tempat yang bising. Sediakan jalur kedua (Langkah 5 pada **Tabel 9.3**).
- **Gestur tanpa alternatif.** Hapus hanya dengan usapan, perbesar hanya dengan cubitan, dan urut ulang hanya dengan seretan tidak terlihat bagi banyak pengguna dan gagal pada kriteria WCAG 2.2 2.5.1 dan 2.5.7. Tambahkan tombol yang terlihat.
- **Mengukur target dengan mata.** Tautan yang tampak wajar di monitor besar perancang tingginya 13 piksel di ponsel. Ukurlah di peramban pada lebar ponsel, seperti yang dilakukan **Kode 9.10**.
- **Menyamakan pengecualian jarak dengan lulus.** Target kecil yang lulus 2.5.8 berkat jarak mungkin tetap sulit dikenai; bidik 44–48 piksel untuk apa pun yang penting.
- **Menebak alih-alih bertanya.** Antarmuka suara yang bertindak atas tafsiran berkeyakinan rendah akan memesan menu yang salah; yang mengonfirmasi segalanya akan menjengkelkan. Konfirmasi ketika ragu atau ketika tindakannya mahal.
- **Menganggap keyakinan sebagai kebenaran.** Banyaknya kata yang dipahami tidak berarti jumlah yang benar terbaca, seperti diperlihatkan "*es teh manis dua gelas*". Uji pengurai dengan ucapan nyata dari pelanggan nyata.
- **Logika di antarmuka.** Jika harga atau aturan dihitung terpisah di setiap antarmuka, gaya-gaya itu akan saling berselisih. Pertahankan satu inti.
- **Umpan balik yang lambat.** Tombol yang tidak menampilkan apa pun selama satu detik terasa rusak. Tampilkan akibatnya dalam 0,1 detik dan kerjakan pekerjaan lambat sesudahnya.

9.12 Ringkasan Bab

- Gaya interaksi adalah cara pengguna menyampaikan niat dan melihat hasil. Lima gaya utamanya adalah bahasa perintah, pemilihan menu, pengisian formulir, manipulasi langsung, dan bahasa alami; sentuhan, gestur, suara, dan tatapan adalah modalitas yang membawanya (**Tabel 9.1**).
- Mengenali (menu, objek yang terlihat) lebih mudah daripada mengingat (perintah); mengingat lebih cepat bagi ahli. Manipulasi langsung mempersempit kedua jurang Norman dengan menjaga objek tetap terlihat dan tindakan tetap cepat, bertahap, dan dapat dibatalkan.
- Antarmuka bersifat berbasis peristiwa. Umpan balik dalam sekitar 0,1 detik terasa seketika, dalam 1 detik menjaga alur pikiran, dan lewat 10 detik kehilangan perhatian.
- Hukum Fitts (Persamaan (9.1)) membuat target kecil menjadi lambat; hukum FFitts (Persamaan (9.2)) menambahkan ketidaktepatan mutlak jari, yang membuat target sentuh yang

sangat kecil rawan salah. WCAG 2.2 mensyaratkan 24 × 24 px CSS pada tingkat AA (dengan pengecualian) dan 44 × 44 pada AAA; Apple menganjurkan 44 pt, Material 48 dp (**Tabel 9.2**).

- Gestur memerlukan alternatif yang terlihat; kriteria WCAG 2.2 2.5.1 dan 2.5.7 mensyaratkan alternatif satu penunjuk bagi gestur lintasan dan penyeretan.
- Antarmuka suara tidak memiliki menu yang terlihat, tidak pasti, dan sekejap. Konfirmasi secara implisit ketika yakin, secara eksplisit ketika ragu (Persamaan (9.3)) atau ketika tindakannya mahal, dan selalu berikan contoh serta jalan pembatalan.
- Program bab ini membangun satu inti pesanan dengan tiga antarmuka (**Gambar 9.1**), kios yang dilayani secara identik oleh Flask dan oleh peladen HTTP JDK, serta audit berbasis peramban yang menemukan dan memastikan perbaikan tautan yang terlalu kecil (**Gambar 9.2**).

9.13 Soal Latihan

1. Dengan Persamaan (9.1), hitung indeks kesulitan target selebar 24, 44, dan 48 piksel pada jarak 300 piksel. Berapa bit yang dihemat dengan beralih dari 24 ke 48 piksel, dan mengapa penghematan itu lebih kecil daripada beralih dari 13 ke 24?
2. Golongkan setiap langkah menarik tunai di ATM bertombol samping (masukkan kartu, pilih bahasa, masukkan PIN, pilih "penarikan", pilih jumlah, konfirmasi) menurut gaya interaksi pada **Tabel 9.1**. Langkah mana yang akan Anda ubah menjadi manipulasi langsung pada ATM layar sentuh, dan mana yang akan Anda biarkan?
3. Untuk setiap situasi berikut, tentukan apakah Anda memakai konfirmasi implisit atau eksplisit, dan tuliskan kalimat yang diucapkan sistem: (a) menambah satu es teh lewat suara dengan keyakinan 1,0; (b) membayar Rp61.000 lewat suara; (c) membatalkan seluruh pesanan lewat suara.
4. Ucapan "*es teh manis dua gelas*" disalahpahami (Subbab 9.8.2). Ubah `parse_utterance` agar kata bilangan dalam jarak dua kata *sesudah* menu diambil sebagai jumlahnya, dan agar kata *manis*, *gelas*, dan *porsi* dihitung sebagai dipahami. Tunjukkan hasil baru untuk ucapan ini dan pastikan pengujian tetap lulus.
5. Buat "*berapa totalnya*" berfungsi: perluas pengurai agar kata yang berakhiran *-nya* juga dicari tanpa akhiran itu. Imbuhan bahasa Indonesia apa lagi yang mungkin dipakai pelanggan warung?
6. Apa yang terjadi pada sesi **Kode 9.6** jika `CONFIRM_BELOW` diubah dari 0,6 menjadi 0,5? Menjadi 0,3? Jawablah dengan bernalar dari nilai keyakinan dalam transkrip, lalu periksa dengan menjalankannya.
7. Peta di sebuah aplikasi pesan-antar hanya dapat diperbesar dengan mencubit, dan alamat tersimpan hanya dapat diurutkan ulang dengan menyeret. Kriteria WCAG 2.2 mana yang dilanggarnya, dan perubahan terkecil apa yang membuatnya lulus?

9.14 Penyelesaian Soal Latihan

1. $ID_{24} = \log_2(300/24 + 1) = \log_2(13.5) = 3.75$ bit; $ID_{44} = \log_2(300/44 + 1) = \log_2(7.82) = 2.97$ bit; $ID_{48} = \log_2(300/48 + 1) = \log_2(7.25) = 2.86$ bit. Beralih dari 24 ke 48 piksel menghemat $3.75 - 2.86 = 0.89$ bit. Beralih dari 13 ke 24 piksel menghemat $4.59 - 3.75 = 0.84$ bit padahal lebarnya hanya bertambah 11 piksel: karena indeksnya logaritmis, setiap *penggandaan* lebar menghemat kira-kira sama besar (sedikit di bawah satu bit ketika D/W besar), sehingga keuntungan terbesar ada pada target terkecil. Selain itu, menurut

Persamaan (9.2), target 13 piksel berada dekat ketidaktepatan mutlak jari, tempat kesalahan — bukan hanya waktu — meningkat.

2. Memasukkan kartu adalah tindakan fisik (manipulasi langsung atas benda di dunia nyata). Memilih bahasa, memilih "penarikan", dan memilih jumlah adalah pemilihan menu dengan tombol samping. Memasukkan PIN adalah pengisian formulir (satu ruas) pada papan tombol. Konfirmasi adalah pemilihan menu dengan dua pilihan. Pada ATM layar sentuh, pemilihan jumlah paling diuntungkan oleh manipulasi langsung: tombol besar untuk jumlah yang umum ditambah ruas "jumlah lain", dengan hasilnya (jumlah dan biaya) terlihat sebelum konfirmasi. Pemasukan PIN sebaiknya tetap pada papan tombol fisik atau terlindung demi privasi; PIN lewat gestur atau suara akan terdengar atau terlihat orang lain.

3. (a) Implisit, karena sistem yakin dan tindakannya murah serta dapat dibatalkan: "Es teh ditambahkan — total Rp5.000. Ada lagi?" (b) Eksplisit, karena membayar itu mahal dan sulit dibatalkan, berapa pun keyakinannya: "Anda akan membayar Rp61.000 untuk 2 es teh, 3 nasi goreng, dan 1 kopi. Katakan 'bayar' untuk memastikan atau 'batal'." (c) Eksplisit, karena menghapus segalanya: "Batalkan seluruh pesanan berisi 6 butir? Katakan 'ya, batalkan' untuk memastikan."

4. Dua perubahan pada `order_core.py`: tambahkan `"manis", "gelas", "porsi"` ke `FILLER`, dan ganti pencarian bilangan sesudah menu dengan pencarian pada dua kata berikutnya:

```
        elif any(w in NUMBERS for w in words[j:j + 2]):
            k = next(k for k in range(j, j + 2) if words[k] in NUMBERS)
            n = NUMBERS[words[k]]
            known[k] = True
            j = k + 1
```

Dengan perubahan ini `parse_utterance("es teh manis dua gelas", menu)` mengembalikan `('add', [('esteh', 2)], 0.8)`: jumlahnya kini 2. Keyakinannya $4/5 = 0.8$, bukan 1,0, karena pemindaian melompat dari menu langsung melewati *manis* ke bilangan, sehingga *manis* tidak pernah dicari; menandai kata yang dilompati sebagai dipahami bila ia kata pengisi akan menaikkan keyakinan menjadi 1,0. Keempat pengujian yang ada tetap lulus, karena dalam "*tolong nasi goreng dua*" bilangannya adalah kata tepat berikutnya. Perubahan yang sama di `OrderCore.java` mengganti cabang `else if` dengan gelung atas `j` dan `j + 1`. Perhatikan risiko yang ditimbulkannya: dalam "*nasi goreng es teh dua*", *dua* berada tiga kata sesudah *nasi goreng*, di luar jendela, sehingga pengurai memberikan satu nasi goreng dan dua es teh, sesuai maksud pelanggan; tetapi dalam "*nasi goreng teh dua*", jendela menjangkau *dua* dari nasi goreng, dan pengurai mengembalikan dua nasi goreng dengan keyakinan 0,75 serta kehilangan tehnya sama sekali. Ambiguitas tidak dapat dihapus oleh jendela yang lebih cerdik, hanya dikurangi — dan ditangkap oleh konfirmasi.

5. Sebelum mencari kata di `ADD`, `REMOVE`, `TOTAL`, dan `FILLER`, buang akhiran *-nya* bila kata itu lebih panjang dari lima huruf: `if w.endswith("nya") and len(w) > 5: w = w[:-3]`. Maka "*totalnya*" menjadi *total*, niatnya `total`, dan keyakinannya 1,0. Imbuhan lain yang terdengar di warung antara lain akhiran *-kan* dan *-in* (*pesankan*, *tambahin*), awalan *di-* (*dibungkus*), reduplikasi (*anget-anget*), dan kata-kata Jawa (*ngunjuk*, *mbak*, *nggih*). Pemotong kata dasar (*stemmer*) seperti yang ada di pustaka Sastrawi menangani imbuhan bahasa Indonesia baku; kata-kata daerah harus dikumpulkan dengan mendengarkan.

6. Nilai keyakinan dalam transkrip adalah 1,0 (ucapan pertama), 0,6 (tanpa menu), dan 0,4 (kopi). Dengan 0,5 tidak ada yang berubah: 0,4 tetap di bawah 0,5, sehingga kopi tetap memerlukan "*ya*", dan ucapan tanpa menu ditolak karena tidak ada menunya, bukan karena keyakinannya. Dengan

0,3, ucapan kopi langsung dijalankan tanpa konfirmasi, dan "*say ya*" berikutnya menjadi ucapan tanpa menu yang dijawab "Sorry, which menu item?". Menjalankan sesi dengan `CONFIRM_BELOW = 0.3` mencetak baris-baris persis seperti itu.

7. Perbesaran yang hanya dengan cubitan gagal pada kriteria keberhasilan 2.5.1 *Pointer Gestures* (gestur multititik tanpa alternatif satu penunjuk); pengurutan ulang yang hanya dengan seretan gagal pada 2.5.7 *Dragging Movements*. Perubahan terkecilnya: tambahkan tombol perbesar dan perkecil yang terlihat pada peta, dan tambahkan tombol "naikkan" dan "turunkan" (atau butir menu "pindahkan ke atas") pada setiap alamat tersimpan. Tombol-tombol itu sendiri harus berukuran sekurang-kurangnya 24 kali 24 piksel CSS untuk memenuhi 2.5.8, dan 44 atau lebih agar nyaman.

Daftar Pustaka

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

[2] B. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 6th ed. Boston, MA, USA: Pearson, 2016.

[3] D. A. Norman, *The Design of Everyday Things*, revised and expanded ed. New York, NY, USA: Basic Books, 2013.

[4] J. Nielsen, "Response Times: The 3 Important Limits," Nielsen Norman Group, diperbarui 2014, berdasarkan J. Nielsen, *Usability Engineering*, bab 5. San Francisco, CA, USA: Morgan Kaufmann, 1993. [Daring]. Tersedia: https://www.nngroup.com/articles/response-times-3-important-limits/

[5] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," *Journal of Experimental Psychology*, vol. 47, no. 6, pp. 381–391, 1954.

[6] X. Bi, Y. Li, and S. Zhai, "FFitts law: Modeling finger touch with Fitts' law," in *Proc. CHI '13*, Paris, France, 2013, pp. 1363–1372, doi: 10.1145/2470654.2466180.

[7] W3C, "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, 12 Des. 2024; disahkan sebagai ISO/IEC 40500:2025, Okt. 2025. [Daring]. Tersedia: https://www.w3.org/TR/WCAG22/

[8] Apple Inc., "Accessibility," *Human Interface Guidelines*, diakses Sep. 2026. [Daring]. Tersedia: https://developer.apple.com/design/human-interface-guidelines/accessibility

[9] Google, "Accessibility basics: Touch and pointer target size," *Material Design 3*, diakses Sep. 2026. [Daring]. Tersedia: https://m3.material.io/foundations/designing/structure

[10] Business Today, "Google Assistant to be replaced by Gemini starting September on Android and Wear OS," 6 Agu. 2026. [Daring]. Tersedia: https://www.businesstoday.in/technology/news/story/google-assistant-to-be-replaced-by-gemini-starting-september-on-android-and-wearos-547570-2026-08-06

[11] "I Can Do It: Exploring Voice Assistants for Adults with Intellectual Disabilities," in *Proc. 2026 CHI Conf. Human Factors in Computing Systems (CHI '26)*, 2026, doi: 10.1145/3772318.3790573.

[12] Apple Inc., "New versions of Apple's software platforms are available today," Apple Newsroom, 15 Sep. 2025. [Daring]. Tersedia: https://www.apple.com/newsroom/2025/09/new-versions-of-apples-software-platforms-are-available-today/

[13] Google Design, "Expressive Design: Google's UX Research," 2025. [Daring]. Tersedia: https://design.google/library/expressive-material-design-google-research

[14] Microsoft, "Fluent 2 Design System," diakses Sep. 2026. [Daring]. Tersedia: https://fluent2.microsoft.design/

[15] Apple Inc., “Designing for visionOS,” *Human Interface Guidelines*, diakses Sep. 2026. [Daring]. Tersedia: https://developer.apple.com/design/human-interface-guidelines/designing-for-visionos

Bab 10

UX Seluler dan Responsif

Sebagian besar orang yang akan memakai antarmuka Anda akan memegangnya dengan satu tangan. Di Indonesia, yang dipegang sering kali ponsel Android murah, di bawah terik matahari, dengan koneksi yang cepat di pusat kota tetapi lambat beberapa kilometer dari sana, sambil mengerjakan hal lain. Bab ini membahas perancangan untuk tangan itu, layar itu, dan jaringan itu. Bab ini mengikuti halaman pemesanan sebuah pangkalan ojek kecil yang versi pertamanya dirancang di komputer desktop dan nyaris tak dapat dipakai di ponsel para pengemudi dan pelanggannya, lalu membangunnya ulang dengan pendekatan *mobile-first* (seluler lebih dulu). Di akhir bab Anda akan mampu membaca statistik seluler Indonesia dengan benar dan menjelaskan mengapa dua sumber tepercaya tampak berselisih; menjelaskan piksel CSS, piksel perangkat, dan *viewport*, serta menghitung seberapa kecil halaman desktop menjadi di ponsel; membangun tata letak cair yang *mobile-first* dengan titik henti (*breakpoint*) yang dipilih oleh konten; menempatkan tindakan penting dalam jangkauan ibu jari; mempercepat formulir di ponsel dengan jenis masukan dan token *autocomplete* yang tepat; memperkirakan waktu muat dari bobot halaman, waktu pulang-pergi, dan lebar pita serta menjaga halaman dalam anggaran kinerja; menerapkan kriteria WCAG 2.2 yang paling penting di ponsel; dan menguji semua itu secara otomatis pada empat lebar layar di peramban sungguhan, dengan peladen dalam Java dan Python.

10.1 Ponsel Mas Yoga

Pak Darto mengelola sebuah pangkalan ojek di Jalan Kenari: sebuah bangku di bawah pohon, beberapa sepeda motor, dan empat pengemudi yang bergiliran. Sebagian besar perjalanan masih dimulai dengan seseorang yang berjalan ke bangku itu, tetapi makin banyak pelanggan yang ingin memesan lebih dahulu — seorang ibu yang harus tiba di puskesmas pukul delapan, seorang mahasiswa yang masuk kelas pukul sembilan. Keponakan Pak Darto membuatkan halaman web pemesanan kecil untuk pangkalan itu di komputer desktopnya. Di monitor tampilannya rapi: tabel pengemudi dengan nomor pelat dan statusnya, formulir berisi empat ruas, serta tautan untuk menelepon atau mengirim pesan ke pangkalan.

Lalu Mas Yoga, salah seorang pengemudi, membukanya di ponsel sambil menunggu di bawah pohon. Halaman itu muncul sebagai kartu pos kelabu yang mungil. Teksnya terlalu kecil untuk dibaca tanpa diperbesar; setelah diperbesar, ia harus menggulir ke samping untuk melihat kolom status; ruas nomor telepon membuka papan ketik huruf lengkap, bukan papan angka; tautan “Telepon” begitu kecil sehingga ibu jarinya dua kali mengenai “WhatsApp”. Seorang pelanggan, Bu Ani, menyerah dan menelepon pangkalan — halaman itu telah mengubah tugas dua menit menjadi tugas yang menjengkelkan, dan menjadikan panggilan telepon satu-satunya antarmuka yang andal.

Tidak ada yang *salah* pada halaman itu dalam ukuran desktop. Kegagalannya terletak pada asumsi desain: halaman itu mengandaikan layar lebar, tetikus, meja yang tenang, dan koneksi kabel yang cepat. Yang dimiliki Mas Yoga adalah layar sempit, ibu jari, dunia yang bergerak, dan jaringan seluler. Bab ini membahas penggantian asumsi-asumsi itu dengan asumsi yang benar.

10.2 Gagasannya dalam Bahasa Sehari-hari

Antarmuka yang **responsif** adalah satu halaman yang menyesuaikan tata letaknya dengan layar yang dijumpainya, bukan satu halaman untuk desktop ditambah "situs seluler" yang terpisah. Istilah ini diciptakan Ethan Marcotte pada 2010 untuk gabungan tiga teknik: kisi cair yang ukurannya dinyatakan dalam proporsi, bukan piksel tetap; gambar yang lentur; dan *media query* CSS yang mengubah tata letak pada lebar-lebar tertentu [1]. **Mobile-first**, istilah yang dipopulerkan Luke Wroblewski, membalik kebiasaan lama merancang untuk desktop lalu memampatkannya: rancang dahulu untuk layar yang paling kecil dan paling terbatas, tempat hanya hal yang benar-benar penting yang muat, lalu *tambahkan* tata letak untuk layar yang lebih besar [2].

Keterbatasan ponsel lebih dari sekadar ukuran. Ponsel dipakai dengan satu tangan dan satu ibu jari; penggunanya sering terinterupsi; sinar matahari memudarkan warna yang kontrasnya rendah; jaringan berubah-ubah dari cepat sampai nyaris tak ada; dan prosesor ponsel murah beberapa kali lebih lambat daripada prosesor laptop. Desain seluler menganggap serius semua hal ini. Kabar baiknya, merancang untuk keterbatasan-keterbatasan ini hampir selalu memperbaiki versi desktop pula: halaman yang jelas di ponsel 360 piksel akan jelas di monitor 1280 piksel.

> **DEFINISI.** Desain *responsif* adalah satu antarmuka yang tata letaknya menyesuaikan diri dengan ukuran dan kemampuan perangkat, memakai proporsi cair, media yang lentur, dan aturan bersyarat (*media query* atau *container query*). *Mobile-first* adalah praktik merancang layar terkecil dan kondisi terlemah lebih dahulu lalu memperkaya desain untuk layar yang lebih besar dan kondisi yang lebih baik. *Breakpoint* (titik henti) adalah lebar tempat tata letak berubah.

10.3 Teori dan Buktinya

10.3.1 Kenyataan seluler Indonesia, dibaca dengan cermat

Tiga sumber menggambarkan pengguna digital Indonesia, dan ketiganya mengukur hal yang berbeda. Survei Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) yang dirilis Mei 2026 menghitung *orang*: 81,72% penduduk, sekitar 235 juta orang, adalah pengguna internet [3]. StatCounter menghitung *tayangan halaman* pada situs web yang memasang kode pelacaknya. Untuk Agustus 2026 ia melaporkan bahwa 38,98% tayangan halaman di Indonesia berasal dari perangkat seluler, 60,59% dari desktop, dan 0,43% dari tablet [4]; di antara tayangan seluler, Android memperoleh 77,07% dan iOS 22,88% [5]. Dan Speedtest Global Index dari Ookla, sebagaimana dilaporkan Bloomberg Technoz, memberikan kecepatan unduh seluler *median* Indonesia untuk Februari 2026 sebesar 59,18 Mbit/s, dengan unggah 17,97 Mbit/s dan latensi 21 ms [6].

Sekilas angka StatCounter bertentangan dengan pengalaman sehari-hari: bukankah sebagian besar orang Indonesia menjelajah web lewat ponsel? Jawabannya terletak pada apa yang dihitung. StatCounter mengukur tayangan halaman pada situs yang memakai pelacaknya, dan seorang pengguna desktop di kantor dapat melihat banyak halaman situs semacam itu dalam sehari, sementara pengguna ponsel dapat menghabiskan harinya di aplikasi — obrolan, video, ojek daring — yang sama sekali bukan halaman web. Jadi angka itu menyatakan sesuatu yang benar tentang *tayangan halaman web di situs-situs itu*, bukan tentang orang. Pelajaran bagi perancang bersifat umum: sebelum mengutip suatu angka, tanyakan *apa* yang dihitung, *di mana*,

dan *bagaimana*. Untuk halaman pemesanan ojek, yang penggunanya adalah para pengemudi dan tetangga yang memegang ponsel, fakta yang relevan adalah yang diperlihatkan Mas Yoga kepada kita, dan riset pengguna yang relevan adalah jenis yang diuraikan di Bab 5.

Angka Ookla memerlukan kecermatan yang sama. Median 59 Mbit/s berarti separuh pengujian yang diukur lebih cepat dan separuh lebih lambat; ia tidak menyatakan apa pun tentang sepersepuluh yang paling lambat, tempat pengguna perdesaan, sel yang padat, dan sinyal dalam ruangan berada. Karena itulah Subbab 10.3.6 merancang terhadap profil yang sengaja dibuat lambat, bukan terhadap median.

10.3.2 Piksel, viewport, dan mengapa halaman desktop menciut

Layar ponsel memiliki jauh lebih banyak *piksel perangkat* daripada *piksel CSS* sebuah halaman web. Peramban memetakan satu piksel CSS ke beberapa piksel perangkat menurut **rasio piksel perangkat** (*device pixel ratio*) r ponsel itu (2, 3, atau lebih pada ponsel masa kini). Jika sebuah layar selebar w_{dev} piksel perangkat, lebar yang tersedia bagi halaman dalam piksel CSS adalah

$$w_{css} = \frac{w_{dev}}{r} \tag{10.1}$$

Ponsel selebar 1080 piksel perangkat dengan $r = 3$ menyediakan lebar 360 piksel CSS. Perancang bekerja dalam piksel CSS, dan karena itulah buku ini menyebut "ponsel 360 piksel".

Viewport adalah persegi panjang tempat halaman ditata. Untuk menampilkan halaman yang hanya dirancang bagi desktop, peramban seluler secara bawaan menata halaman dalam *viewport* maya yang lebar — biasanya 980 piksel CSS — lalu memperkecilnya agar muat di layar. Halaman dapat mematikan perilaku ini dengan satu baris di bagian kepalanya, `<meta name="viewport" content="width=device-width, initial-scale=1">`, yang memberi tahu peramban agar menata halaman pada lebar perangkat itu sendiri [7]. Tanpa baris itu, semua yang ada di halaman dikalikan dengan faktor skala

$$s = \frac{w_{vp}}{w_{layout}} \tag{10.2}$$

dengan w_{vp} lebar layar dalam piksel CSS dan w_{layout} lebar tempat halaman ditata. Teks atau tombol setinggi h_{css} piksel lalu tampak bagi pengguna dengan ukuran efektif

$$h_{eff} = s \cdot h_{css} \tag{10.3}$$

> **CONTOH.** Halaman pertama Pak Darto tidak memiliki baris *viewport*, sehingga ponsel 360 piksel menatanya pada 980 piksel. Menurut Persamaan (10.2), $s = 360/980 = 0.367$. Tulisan kecil di bawah formulir, berukuran 11 piksel CSS, menurut Persamaan (10.3) tampak sebesar $0.367 \times 11 = 4.04$ piksel — kira-kira setinggi tanda koma dalam kalimat ini. Tautan "Telepon", setinggi 12 piksel, menjadi target sebesar $0.367 \times 12 = 4.4$ piksel. Di ponsel 320 piksel keadaannya lebih buruk: $s = 320/980 = 0.327$ dan tulisan kecil itu menciut menjadi 3,6 piksel.

10.3.3 Tata letak cair dan titik henti yang dipilih konten

Tata letak responsif dimulai sebagai satu kolom yang mengisi lebar layar, dengan ukuran dalam satuan relatif (`%`, `rem`, `fr`) dan `max-width` pada konten agar baris tidak menjadi terlalu panjang di monitor besar. ***Media query*** seperti `@media (min-width: 700px) { … }` lalu menambahkan tata

letak bila ada ruang: dua kolom kartu pengemudi, formulir dua kolom. ***Container query*** yang lebih baru memungkinkan sebuah komponen menyesuaikan diri dengan lebar wadahnya, bukan lebar seluruh jendela, yang cocok untuk komponen yang dipakai ulang seperti kartu yang muncul di bilah samping yang sempit maupun di kolom utama yang lebar [7].

Di mana titik henti harus diletakkan? Bukan pada lebar model-model ponsel tertentu, yang berubah setiap tahun, melainkan di tempat *konten* membutuhkannya: lebarkan jendela perlahan-lahan dan tambahkan titik henti ketika tata letak mulai tampak janggal — baris terlalu panjang, kartu terlalu lebar, ruang kosong terbuang. Untuk halaman pemesanan, satu kolom enak dibaca hingga sekitar 700 piksel; di atas itu, kartu pengemudi yang berdampingan lebih hemat ruang. Titik henti mengikuti konten, bukan daftar perangkat.

10.3.4 Tangan dan ibu jari

Bagaimana orang sebenarnya memegang ponsel? Steven Hoober mengamati 1.333 orang yang memakai ponsel di tempat umum dan mendapati bahwa 49% memegang ponsel dengan satu tangan dan mengoperasikannya dengan ibu jari, 36% menopangnya dengan satu tangan dan memakai jari atau ibu jari tangan lain, dan 15% memegangnya dengan dua tangan dan mengetik dengan kedua ibu jari [8]. Orang juga sering berganti pegangan, bahkan dalam satu tugas. Dua aturan praktis mengikutinya. Pertama, tindakan yang paling sering dan paling penting hendaknya berada dalam jangkauan mudah ibu jari, yang pada ponsel besar berarti bagian bawah dan tengah layar — maka lahirlah bilah navigasi bawah dan tombol aksi mengambang. Kedua, tindakan yang merusak atau jarang dipakai boleh diletakkan lebih jauh, tempat ia tidak tersentuh tanpa sengaja.

Dua tindakan terpenting Mas Yoga adalah “Telepon” dan “WhatsApp”. Halaman responsif meletakkan keduanya pada bilah yang menetap di dasar layar, masing-masing selebar separuh layar dan setinggi 52 piksel: mudah dijangkau ibu jari dalam pemakaian satu tangan, mustahil tertukar satu sama lain. Aturan ukuran target di Bab 9 (sekurang-kurangnya 24 piksel CSS pada tingkat AA WCAG, 44 sampai 48 agar nyaman) berlaku tanpa perubahan.

10.3.5 Formulir di ponsel

Mengetik di atas kaca itu lambat dan rawan salah, sehingga formulir seluler hendaknya meminta sesedikit mungkin dan membantu sisanya. HTML menyediakan tiga penolong sederhana [9]. **Jenis masukan** (*input type*) memilih papan ketik: `type="tel"` membuka papan angka untuk nomor telepon, `type="email"` menambahkan tombol @. **Token** `autocomplete` memberi tahu peramban arti ruas itu, sehingga peramban dapat menawarkan nama, nomor telepon, atau alamat pengguna yang tersimpan dengan satu ketukan: `autocomplete="name"`, `"tel"`, `"street-address"`. Atribut `inputmode` memilih papan ketik tanpa mengubah jenis ruasnya, misalnya `inputmode="numeric"` untuk kode angka. Kriteria keberhasilan WCAG 2.2 1.3.5 *Identify Input Purpose* (tingkat AA) mensyaratkan bahwa ruas yang mengumpulkan informasi tentang pengguna menyatakan tujuannya dengan cara yang dapat dibaca mesin seperti ini [10], karena pengisian otomatis paling menolong orang dengan disabilitas motorik dan kognitif.

Label bahkan lebih penting di ponsel daripada di desktop. *Placeholder* di dalam ruas lenyap begitu pengguna mulai mengetik, sehingga label harus berupa `<label>` sungguhan di atas ruas, dihubungkan dengannya melalui `for` dan `id`; dengan begitu ketukan pada label juga memfokuskan ruas, dan pembaca layar mengumumkannya.

10.3.6 Kecepatan: *vitals*, perjalanan pulang-pergi, dan anggaran

Core Web Vitals dari Google meringkas pemuatan, daya tanggap, dan kestabilan visual halaman dalam tiga ukuran, masing-masing dinilai "baik" pada persentil ke-75 kunjungan nyata: **Largest Contentful Paint** (LCP), waktu hingga elemen terlihat terbesar tergambar, hendaknya paling lama 2,5 detik; **Interaction to Next Paint** (INP), jeda dari interaksi hingga layar diperbarui, hendaknya paling lama 200 ms (INP menggantikan First Input Delay yang lebih lama pada Maret 2024); dan **Cumulative Layout Shift** (CLS), seberapa banyak konten melompat-lompat selama pemuatan, hendaknya paling besar 0,1 [11].

Berapa lama sebuah halaman tiba dapat diperkirakan dengan model sederhana. Sebelum bita pertama kembali, ponsel memerlukan beberapa **perjalanan pulang-pergi** (*round trip*) ke peladen: untuk mencari nama (DNS), membuka koneksi (TCP), menyepakati enkripsi (TLS), dan mengirim permintaan. Jika setiap perjalanan pulang-pergi memakan RTT detik dan jumlahnya k, dan halaman beserta semua yang diperlukan untuk menggambarnya berbobot S kilobita yang tiba dengan lebar pita B kilobit per detik, maka waktunya kira-kira

$$t \approx k \cdot RTT + \frac{8S}{B} \tag{10.4}$$

Faktor 8 mengubah bita menjadi bit. Model ini mengabaikan banyak rincian — unduhan paralel, kompresi, *cache*, waktu menjalankan skrip — tetapi menangkap dua hal yang dikendalikan perancang: berapa banyak perjalanan pulang-pergi yang diperlukan halaman dan berapa banyak bita yang dikirimnya.

Untuk menguji halaman di bawah tekanan yang realistis, alat Lighthouse yang tertanam di Chrome secara bawaan menyimulasikan kunjungan seluler dengan waktu pulang-pergi 150 ms, laju 1.638,4 kbit/s (sekitar 1,6 Mbit/s), dan prosesor empat kali lebih lambat daripada komputer penguji [12]. Profil itu jauh lebih lambat daripada median Indonesia, dan memang disengaja: halaman yang baik di bawah profil itu baik bagi hampir semua orang. **Anggaran kinerja** (*performance budget*) menjadikannya aturan desain: tetapkan, sebelum merancang, berapa kilobita yang boleh ditimbang sebuah halaman agar Persamaan (10.4) tetap di bawah 2,5 detik pada profil yang dipilih, dan perlakukan setiap foto, fon, dan skrip sebagai belanja dari anggaran itu.

10.3.7 Kriteria WCAG yang paling terasa di ponsel

Beberapa kriteria WCAG 2.2 [10] ada terutama karena layar kecil dan pembesaran:

- **1.4.10 Reflow (AA).** Konten harus dapat dipakai pada lebar 320 piksel CSS tanpa menggulir ke dua arah (tabel data dan peta dikecualikan). Ini setara dengan jendela desktop 1280 piksel yang diperbesar 400%, sehingga juga melayani pengguna desktop dengan penglihatan rendah.
- **1.4.4 Resize Text (AA).** Teks harus dapat diperbesar hingga 200% tanpa kehilangan konten atau fungsi.
- **1.3.4 Orientation (AA).** Konten tidak boleh dikunci pada posisi tegak atau mendatar kecuali memang esensial; pengemudi yang ponselnya terpasang di dudukan dasbor memakai posisi mendatar.
- **1.3.5 Identify Input Purpose (AA)**, seperti di atas, dan **2.5.8 Target Size (Minimum) (AA)**, seperti di Bab 9.

10.4 Metode Praktis: Daftar Periksa *Mobile-First*

Tabel 10.1 adalah daftar periksa yang dijalankan pada setiap halaman sebelum ditayangkan. Kolom pertama berisi pertanyaan, kolom kedua cara memeriksanya, dan kolom ketiga nilai halaman pertama Pak Darto.

Tabel 10.1 Daftar periksa *mobile-first*, dengan hasil untuk versi pertama halaman pemesanan.

Pemeriksaan	Cara memeriksa	Halaman pemesanan pertama
Ada baris *viewport*	Lihat di kepala halaman	Tidak ada: halaman diperkecil ke 37%
Reflow pada 320 px	Buka pada 320 px; tanpa gulir ke samping	Diperkecil alih-alih ditata ulang
Teks isi sekurang-kurangnya 16 px CSS	Ukur ukuran terhitung × skala	Efektif 3,6–4,0 px di ponsel
Target sekurang-kurangnya 24 px, sebaiknya 44–48	Ukur persegi panjang × skala	Efektif 3,9–4,4 px
Tindakan utama dalam jangkauan ibu jari	Lihat separuh bawah layar	Telepon dan WhatsApp di atas, mungil
Papan ketik dan *autocomplete* tepat	Periksa `type` dan `autocomplete` ruas	Ruas telepon berupa teks biasa
Label di atas ruas, terhubung	Ketuk setiap label; ruas terfokus	Teks lepas di depan setiap ruas
Bobot dalam anggaran	Jumlahkan sumber daya; terapkan Persamaan (10.4)	2.492 kB: 12,8 dtk pada profil uji
Berfungsi mendatar dan pada teks 200%	Putar layar; perbesar teks	Belum diperiksa

Sebagian besar baris dapat diotomatiskan, dan Subbab 10.5 melakukannya. Dua baris yang tidak dapat diotomatiskan — apakah tindakan berada dalam jangkauan ibu jari dan apakah halaman masuk akal bagi penggunanya — memerlukan manusia, dengan metode Bab 5 dan Bab 7.

10.5 Membangunnya Langkah demi Langkah

Program ini melayani dua versi halaman pemesanan yang sama, mengukurnya di peramban sungguhan pada empat lebar, dan memperkirakan waktu muatnya pada tiga jaringan.

10.5.1 Dua templat, satu perangkat kata

Kedua versi adalah templat HTML yang diisi dengan kata-kata yang sama dari tabel kecil berisi teks bahasa Inggris dan Indonesia serta daftar pengemudi yang sama. Versi pertama, **Kode 10.1**, khas halaman yang dirancang di desktop: lebar badan tetap 960 piksel, fon 13 piksel, sebuah tabel, masukan selebar 300 piksel dengan teks lepas di depannya, ruas telepon berjenis `text` biasa, dan tautan panggilan dalam cetakan 11 piksel. Ia tidak memiliki baris *viewport*.

Kode 10.1 `naive.html`: versi pertama halaman pemesanan yang hanya untuk desktop.

```
<!DOCTYPE html>
<html lang="{{LANG}}">
<head>
<meta charset="utf-8">
<title>{{TITLE}}</title>
<style>
body { width: 960px; margin: 0 auto; font: 13px Arial, sans-serif; }
h1 { font-size: 20px; }
table { width: 940px; border-collapse: collapse; }
td, th { border: 1px solid #999; padding: 2px 6px; }
input { width: 300px; font-size: 12px; }
.small { font-size: 11px; }
</style>
</head>
<body>
<h1>{{TITLE}} &ndash; {{TAG}}</h1>
<table><tr><th>{{DRIVERS}}</th><th>{{PLATE}}</th><th>{{STATUS}}</th></tr>
{{ROWS}}</table>
<form method="post" action="/book">
<p>{{NAME}} <input name="name"> {{PHONE}} <input name="phone"></p>
<p>{{FROM}} <input name="from"> {{TO}} <input name="to">
<input type="submit" value="{{BOOK}}"></p>
</form>
<p class="small"><a href="tel:+620000000000">{{CALL}}</a> |
<a href="https://wa.me/620000000000">{{WA}}</a></p>
</body>
</html>
```

Versi responsif, **Kode 10.2**, ditulis *mobile-first*. Gaya dasarnya untuk layar sempit: satu kolom, fon dasar 16 piksel (`1rem`), kartu pengemudi alih-alih tabel, label di atas masukan selebar penuh setinggi 48 piksel, dan bilah yang menetap di dasar layar berisi dua tindakan utama. Satu *media query* menambahkan dua kolom mulai 700 piksel. Ruas telepon memiliki `type="tel"` dan `autocomplete="tel"`; ruas nama dan penjemputan memiliki token *autocomplete*-nya sendiri.

Kode 10.2 `responsive.html`: tulisan ulang *mobile-first* dengan baris *viewport*, tata letak cair, ruas berlabel, dan bilah tindakan dalam jangkauan ibu jari.

```
<!DOCTYPE html>
<html lang="{{LANG}}">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>{{TITLE}}</title>
<style>
* { box-sizing: border-box; }
body { margin: 0; font: 1rem/1.5 "DejaVu Sans", sans-serif; color: #1f2937;
       background: #f3f4f6; padding-bottom: 88px; }
header { background: #12355B; color: #fff; padding: 12px 16px; }
h1 { font-size: 1.3rem; margin: 0; }
header p { margin: 0; }
main { max-width: 960px; margin: 0 auto; padding: 0 16px; }
.drivers { list-style: none; padding: 0; display: grid; gap: 8px; }
.drivers li { background: #fff; border-radius: 8px; padding: 8px 12px; }
.ready { color: #0F7C7C; font-weight: bold; }
.trip { color: #6B7280; }
label { display: block; margin-top: 12px; font-weight: bold; }
input { width: 100%; min-height: 48px; font-size: 1rem; padding: 8px;
        border: 2px solid #6B7280; border-radius: 8px; }
button { width: 100%; min-height: 52px; margin-top: 16px; font: inherit;
         color: #fff; background: #0F7C7C; border: 0; border-radius: 12px; }
nav { position: fixed; bottom: 0; left: 0; right: 0; display: flex;
      gap: 8px; padding: 12px 16px; background: #fff;
      box-shadow: 0 -2px 6px rgba(0, 0, 0, .15); }
nav a { flex: 1; min-height: 52px; display: flex; align-items: center;
        justify-content: center; border-radius: 12px; font-weight: bold;
        text-decoration: none; color: #12355B; border: 2px solid #12355B; }
@media (min-width: 700px) {
  .drivers { grid-template-columns: repeat(2, 1fr); }
  form { display: grid; grid-template-columns: 1fr 1fr; column-gap: 16px; }
  form button { grid-column: span 2; }
}
</style>
</head>
<body>
<header><h1>{{TITLE}}</h1><p>{{TAG}}</p></header>
<main>
<h2>{{DRIVERS}}</h2>
<ul class="drivers">{{CARDS}}</ul>
<form method="post" action="/book">
<div><label for="name">{{NAME}}</label>
<input id="name" name="name" autocomplete="name"></div>
<div><label for="phone">{{PHONE}}</label>
<input id="phone" name="phone" type="tel" autocomplete="tel"></div>
```

Kode 10.2 (Lanjutan) *responsive.html: tulisan ulang mobile-first dengan baris viewport, tata letak cair, ruas berlabel, dan bilah tindakan dalam jangkauan ibu jari.*

```
<div><label for="from">{{FROM}}</label>
<input id="from" name="from" autocomplete="street-address"></div>
<div><label for="to">{{TO}}</label><input id="to" name="to"></div>
<button type="submit">{{BOOK}}</button>
</form>
</main>
<nav><a href="tel:+620000000000">{{CALL}}</a>
<a href="https://wa.me/620000000000">{{WA}}</a></nav>
</body>
</html>
```

Kata-katanya disimpan dalam tabel CSV, **Kode 10.3**, sehingga satu templat melayani kedua bahasa:

Kode 10.3 `strings.csv`: kata-kata halaman dalam bahasa Inggris dan Indonesia.

```
key,en,id
TITLE,Ojek Pak Darto,Ojek Pak Darto
TAG,Book a ride from the Jalan Kenari base,Pesan ojek dari pangkalan Jalan Kenari
DRIVERS,Drivers on duty,Pengemudi yang bertugas
NAME,Your name,Nama Anda
PHONE,Phone number,Nomor ponsel
FROM,Pick me up at,Jemput saya di
TO,Take me to,Antar saya ke
BOOK,Book the ride,Pesan ojek
PLATE,Plate,Pelat nomor
CALL,Call,Telepon
WA,WhatsApp,WhatsApp
STATUS,status,status
READY,ready,siap
TRIP,on a trip,sedang mengantar
```

10.5.2 Mengisi dan melayani halaman

`pages.py` (**Kode 10.4**) membaca teks dan daftar pengemudi, membangun baris tabel untuk versi pertama dan kartu untuk versi kedua, meloloskan setiap nama, dan mengganti setiap `{{KEY}}` di templat. `app.py` (**Kode 10.5**) melayani `/naive` dan `/responsive` dengan Flask pada porta 4100, dengan bahasa yang dipilih melalui `?lang=id`.

Kode 10.4 `pages.py`: mengisi templat mana pun dalam bahasa mana pun.

```
"""Isi kedua templat halaman (naive, responsive) dalam bhs. Inggris/Indonesia."""
import csv
from html import escape
from pathlib import Path

DATA = Path(__file__).resolve().parent.parent / "data"


def rows(name):
    with open(DATA / name, encoding="utf-8") as f:
        return list(csv.DictReader(f))


def render(page, lang):
    text = {r["key"]: r[lang] for r in rows("strings.csv")}
    status = {"ready": text["READY"], "trip": text["TRIP"]}
    table, cards = "", ""
    for d in rows("drivers.csv"):
        name, plate, st = escape(d["name"]), escape(d["plate"]), d["status"]
        table += f"<tr><td>{name}</td><td>{plate}</td><td>{status[st]}</td></tr>\n"
        cards += (f'<li>{name} &middot; {plate} &middot; '
                  f'<span class="{st}">{status[st]}</span></li>')
    html = (DATA / f"{page}.html").read_text(encoding="utf-8")
    for key, val in {**text, "LANG": lang, "ROWS": table,
                     "CARDS": cards}.items():
        html = html.replace("{{" + key + "}}", val)
    return html
```

Kode 10.5 `app.py`: melayani kedua halaman dengan Flask pada porta 4100.

```
"""Layani halaman pemesanan naive dan responsive dengan Flask pada porta 4100."""
from flask import Flask, abort, request
from pages import render

app = Flask(__name__)


@app.get("/<page>")
def show(page):
    if page not in ("naive", "responsive"):
        abort(404)
    lang = "id" if request.args.get("lang") == "id" else "en"
    return render(page, lang)


@app.post("/book")
def book():
    return "<p>OK: " + str(len(request.form)) + " fields received.</p>"


if __name__ == "__main__":
    app.run(port=4100)
```

Versi Java membagi pekerjaan yang sama antara `Pages.java` (**Kode 10.6**) dan `Server.java` (**Kode 10.7**), yang memakai peladen HTTP bawaan JDK pada porta 4101. Untuk halaman dan bahasa yang sama, kedua peladen mengirim bita yang identik.

Kode 10.6 `Pages.java`: pengisian templat 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.*;

/** Isi templat halaman (naive, responsive) dalam bhs. Inggris/Indonesia. */
public class Pages {
    static final Path DATA = Path.of("..", "data");

    static List<Map<String, String>> rows(String name) throws IOException {
        List<String> lines = Files.readAllLines(DATA.resolve(name),
                                                StandardCharsets.UTF_8);
        String[] head = lines.get(0).split(",");
        List<Map<String, String>> out = new ArrayList<>();
        for (String line : lines.subList(1, lines.size())) {
            String[] f = line.split(",", -1);
            Map<String, String> r = new HashMap<>();
            for (int i = 0; i < head.length; i++) r.put(head[i], f[i]);
            out.add(r);
        }
        return out;
    }

    static String esc(String s) {
        return s.replace("&", "&amp;").replace("<", "&lt;").replace(">", "&gt;");
    }

    static String render(String page, String lang) throws IOException {
        Map<String, String> v = new HashMap<>();
        for (var r : rows("strings.csv")) v.put(r.get("key"), r.get(lang));
        Map<String, String> status = Map.of("ready", v.get("READY"),
                                            "trip", v.get("TRIP"));
        StringBuilder table = new StringBuilder(), cards = new StringBuilder();
        for (var d : rows("drivers.csv")) {
            String name = esc(d.get("name")), plate = esc(d.get("plate"));
            String st = d.get("status");
            table.append("<tr><td>" + name + "</td><td>" + plate + "</td><td>"
                         + status.get(st) + "</td></tr>\n");
            cards.append("<li>" + name + " &middot; " + plate + " &middot; "
                         + "<span class=\"" + st + "\">" + status.get(st)
                         + "</span></li>");
        }
        v.put("LANG", lang);
        v.put("ROWS", table.toString());
        v.put("CARDS", cards.toString());
```

Kode 10.6 (Lanjutan) *Pages.java: pengisian templat yang sama dalam Java.*

```
        String html = Files.readString(DATA.resolve(page + ".html"));
        for (var e : v.entrySet())
            html = html.replace("{{" + e.getKey() + "}}", e.getValue());
        return html;
    }
}
```

Kode 10.7 `Server.java`: melayani kedua halaman dengan peladen HTTP JDK pada porta 4101.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;

/** Layani halaman pemesanan naive dan responsive pada porta 4101. */
public class Server {
    static void send(HttpExchange ex, int code, String html) throws IOException {
        byte[] b = html.getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", "text/html; charset=utf-8");
        ex.sendResponseHeaders(code, b.length);
        ex.getResponseBody().write(b);
        ex.close();
    }

    static void handle(HttpExchange ex) throws IOException {
        String path = ex.getRequestURI().getPath();
        String query = ex.getRequestURI().getQuery();
        String lang = "lang=id".equals(query) ? "id" : "en";
        if (ex.getRequestMethod().equals("POST") && path.equals("/book")) {
            String body = new String(ex.getRequestBody().readAllBytes(),
                                     StandardCharsets.UTF_8);
            send(ex, 200, "<p>OK: " + body.split("&").length
                          + " fields received.</p>");
        } else if (path.equals("/naive") || path.equals("/responsive")) {
            send(ex, 200, Pages.render(path.substring(1), lang));
        } else {
            send(ex, 404, "<p>Not found</p>");
        }
    }

    public static void main(String[] args) throws IOException {
        HttpServer server = HttpServer.create(new InetSocketAddress(4101), 0);
        server.createContext("/", Server::handle);
        server.start();
        System.out.println("Pages on http://127.0.0.1:4101/responsive");
    }
}
```

10.5.3 Peramban yang berpura-pura menjadi ponsel

`check_responsive.py` (**Kode 10.8**) memakai Playwright untuk membuka setiap halaman pada 320, 360, 768, dan 1280 piksel CSS. Di bawah 700 piksel ia membuat konteks peramban *seluler* (`is_mobile=True`, dengan sentuhan), sehingga Chromium berperilaku seperti peramban ponsel, termasuk *viewport* bawaan 980 piksel untuk halaman tanpa baris *viewport*. Untuk setiap lebar, ia meminta halaman memberikan lebar tata letaknya (`window.innerWidth`), lebar totalnya (`scrollWidth`, untuk mendeteksi gulir ke samping), ukuran fon terkecil dari elemen mana pun yang memuat teks, sisi terpendek dari tautan, tombol, atau ruas mana pun, dan apakah ruas telepon berjenis `tel`. Ia mengalikan ukuran-ukuran itu dengan faktor skala Persamaan (10.2), sehingga yang dilaporkannya adalah apa yang benar-benar dijumpai mata dan ibu jari, seperti pada Persamaan (10.3). Ia juga dapat menyimpan tangkapan layar pada 360 piksel.

Kode 10.8 `check_responsive.py`: mengukur lebar tata letak, luapan, ukuran teks dan target efektif, serta ruas telepon pada empat lebar.

```
"""Buka halaman pada beberapa lebar ponsel dan desktop, laporkan yang rusak."""
import sys
from playwright.sync_api import sync_playwright

WIDTHS = [320, 360, 768, 1280]
JS = """() => {
  const vw = window.innerWidth;
  const doc = document.documentElement;
  let text = 99, target = 999;
  for (const e of document.querySelectorAll('body *')) {
    const own = [...e.childNodes]
      .some(n => n.nodeType === 3 && n.textContent.trim());
    if (own) text = Math.min(text, parseFloat(getComputedStyle(e).fontSize));
  }
  for (const e of document.querySelectorAll('a, button, input')) {
    const r = e.getBoundingClientRect();
    target = Math.min(target, r.width, r.height);
  }
  const badInputs = [...document.querySelectorAll('input[name=phone]')]
    .filter(e => e.type !== 'tel').length;
  return [vw, doc.scrollWidth, text, target, badInputs];
}"""


def check(base, page, shot_dir=None, lang="en"):
    print(f"{page}: width  layout  overflow  text px  target px  phone input")
    with sync_playwright() as p:
        browser = p.chromium.launch()
        for w in WIDTHS:
            mobile = w < 700   # ponsel merender seperti peramban seluler
            ctx = browser.new_context(viewport={"width": w, "height": 800},
                                      is_mobile=mobile, has_touch=mobile,
                                      device_scale_factor=2)
            pg = ctx.new_page()
            pg.goto(f"{base}/{page}?lang={lang}")
            vw, sw, text, target, bad = pg.evaluate(JS)
            scale = w / vw            # < 1 bila peramban memperkecil halaman
            over = "yes" if sw > vw else "no"
            print(f"{'':8}{w:>5}  {vw:>6}  {over:>8}  {text * scale:>7.1f}"
                  f"  {target * scale:>9.1f}  {'type=text' if bad else 'type=tel'}")
            if shot_dir and w == 360:
                pg.screenshot(path=f"{shot_dir}/{page}_{lang}_360.png")
            ctx.close()
        browser.close()
```

Kode 10.8 (Lanjutan) *check_responsive.py: mengukur lebar tata letak, luapan, ukuran teks dan target efektif, serta ruas telepon pada empat lebar.*

```
if __name__ == "__main__":
    base = sys.argv[1] if len(sys.argv) > 1 else "http://127.0.0.1:4100"
    for page in ("naive", "responsive"):
        check(base, page, *sys.argv[2:3])
```

10.5.4 Perkiraan waktu muat

`budget.py` (**Kode 10.11**) dan `Budget.java` (**Kode 10.12**) menerapkan Persamaan (10.4) dengan $k = 4$ perjalanan pulang-pergi pada daftar sumber daya dan daftar jaringan, lalu membandingkan hasilnya dengan ambang LCP 2,5 detik. Sumber daya versi pertama, **Kode 10.9**, adalah yang akan dibawa halaman sang keponakan setelah selesai — foto besar pangkalan, fon web, kerangka JavaScript — dan versi kedua hanya menyimpan apa yang diperlukan halaman. Jaringannya,

Kode 10.10, adalah koneksi seluler median Indonesia [6], profil uji Lighthouse [12], dan contoh sinyal lemah di tepi jangkauan.

Kode 10.9 `resources_v1.csv`: sumber daya desain pertama yang sudah jadi (desain ramping ada di `resources_v2.csv`).

```
resource,kB
page HTML,32
hero photo of the base (JPEG),1800
web font (2 files),180
JavaScript framework bundle,420
stylesheet,60
```

Kode 10.10 `networks.csv`: tiga profil jaringan yang dipakai untuk perkiraan.

```
network,mbps,rtt_ms
Indonesia median mobile (Ookla Feb 2026),59.18,21
Lighthouse mobile test profile,1.6384,150
Weak signal (example: 0.4 Mbit/s),0.4,300
```

Kode 10.11 `budget.py`: memperkirakan waktu tiba halaman pada setiap jaringan dan membandingkannya dengan anggaran LCP.

```python
"""Perkirakan berapa lama halaman tiba pada berbagai jaringan."""
import csv
import sys
from pathlib import Path

DATA = Path(__file__).resolve().parent.parent / "data"
ROUND_TRIPS = 4      # DNS, TCP, TLS, dan permintaan, sebelum bita pertama
LCP_BUDGET = 2.5     # detik: LCP yang "baik"


def load(name):
    with open(DATA / name, encoding="utf-8") as f:
        return list(csv.DictReader(f))


def seconds(kb, mbps, rtt_ms):
    return ROUND_TRIPS * rtt_ms / 1000 + kb * 8 / (mbps * 1000)


def report(resources):
    kb = sum(float(r["kB"]) for r in load(resources))
    print(f"{resources}: {kb:.0f} kB")
    for n in load("networks.csv"):
        t = seconds(kb, float(n["mbps"]), float(n["rtt_ms"]))
        verdict = "within" if t <= LCP_BUDGET else "OVER"
        print(f"  {n['network']:<42}{t:>6.2f} s  {verdict} {LCP_BUDGET} s")


if __name__ == "__main__":
    for name in sys.argv[1:] or ["resources_v1.csv", "resources_v2.csv"]:
        report(name)
```

Kode 10.12 `Budget.java`: perkiraan yang sama dalam Java.

```java
import java.io.IOException;
import java.util.*;

/** Perkirakan berapa lama halaman tiba pada berbagai jaringan. */
public class Budget {
    static final int ROUND_TRIPS = 4;     // DNS, TCP, TLS, dan permintaannya
    static final double LCP_BUDGET = 2.5; // detik: LCP yang "baik"

    static double seconds(double kb, double mbps, double rttMs) {
        return ROUND_TRIPS * rttMs / 1000 + kb * 8 / (mbps * 1000);
    }

    static void report(String resources) throws IOException {
        double kb = 0;
        for (var r : Pages.rows(resources)) kb += Double.parseDouble(r.get("kB"));
        System.out.printf(Locale.US, "%s: %.0f kB%n", resources, kb);
        for (var n : Pages.rows("networks.csv")) {
            double t = seconds(kb, Double.parseDouble(n.get("mbps")),
                               Double.parseDouble(n.get("rtt_ms")));
            String verdict = t <= LCP_BUDGET ? "within" : "OVER";
            System.out.printf(Locale.US, "  %-42s%6.2f s  %s %s s%n",
                              n.get("network"), t, verdict, LCP_BUDGET);
        }
    }

    public static void main(String[] args) throws IOException {
        String[] names = args.length > 0 ? args
            : new String[] {"resources_v1.csv", "resources_v2.csv"};
        for (String name : names) report(name);
    }
}
```

10.6 Menjalankannya

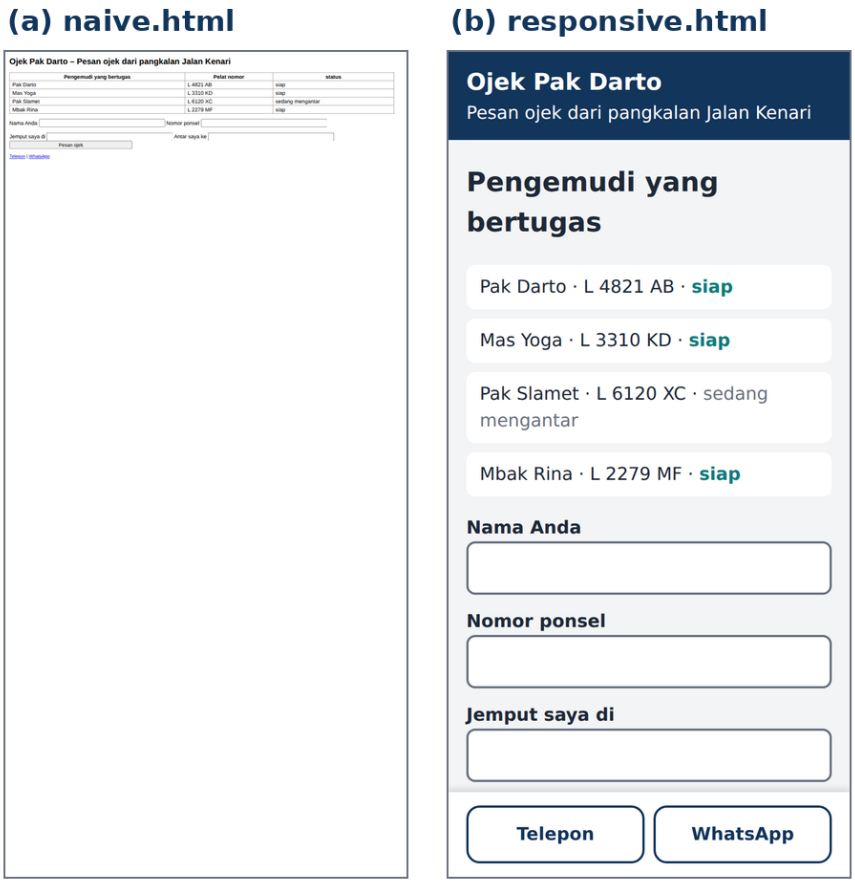
Jalankan salah satu peladen (`python app.py` atau `java -cp out Server`) dan jalankan pemeriksaan di terminal kedua. Terhadap Flask maupun peladen Java, keluarannya sama:

```
naive: width  layout  overflow  text px  target px  phone input
          320     980        no      3.6        3.9  type=text
          360     980        no      4.0        4.4  type=text
          768     768       yes     11.0       12.0  type=text
         1280    1280        no     11.0       12.0  type=text
responsive: width  layout  overflow  text px  target px  phone input
          320     320        no     16.0       48.0  type=tel
          360     360        no     16.0       48.0  type=tel
          768     768        no     16.0       48.0  type=tel
         1280    1280        no     16.0       48.0  type=tel
```

Bacalah blok pertama, versi pertama, baris demi baris. Pada kedua ponsel, lebar tata letaknya 980, bukan 320 atau 360: ketiadaan baris *viewport* membuat peramban menata halaman seperti untuk desktop lalu menciutkannya. Tidak ada gulir ke samping — peramban sudah memperkecil cukup jauh — tetapi ukuran teks efektifnya 3,6 dan 4,0 piksel dan target terkecilnya 3,9 dan 4,4 piksel, persis nilai-nilai Contoh di Subbab 10.3.2. Pada 768 piksel, tablet atau jendela laptop kecil, lebar tata letaknya jujur tetapi badan halaman selebar tetap 960 piksel lebih lebar daripada jendela, sehingga halaman bergulir ke samping (luapan “yes”): bibit kegagalan WCAG 1.4.10. Hanya pada 1280 piksel halaman bekerja seperti yang dilihat perancangnya. Ruas telepon berupa teks biasa pada setiap lebar.

Blok kedua, versi responsif, tanpa kejadian dalam arti terbaiknya: lebar tata letak sama dengan lebar layar pada setiap ukuran, tidak ada yang meluap, tidak ada teks di bawah 16 piksel, dan

tidak ada target di bawah 48. **Gambar 10.1** memperlihatkan kedua halaman sebagaimana ditampilkan ponsel 360 piksel.

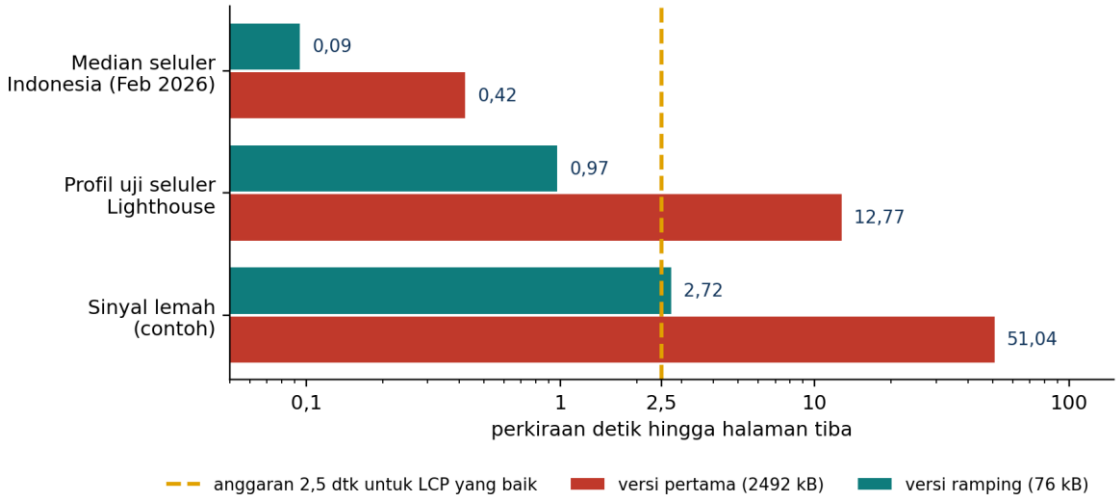


Gambar 10.1 Kedua halaman pemesanan di ponsel 360 piksel: (a) halaman desktop, ditata pada 980 piksel dan diciutkan agar muat, dan (b) halaman *mobile-first* dengan teks yang terbaca, ruas berlabel setinggi 48 piksel, serta bilah Telepon dan WhatsApp dalam jangkauan ibu jari.

Sekarang perkiraan waktu muat, `python budget.py` (versi Java mencetak hal yang sama):

```
resources_v1.csv: 2492 kB
  Indonesia median mobile (Ookla Feb 2026)    0.42 s  within 2.5 s
  Lighthouse mobile test profile             12.77 s  OVER 2.5 s
  Weak signal (example: 0.4 Mbit/s)          51.04 s  OVER 2.5 s
resources_v2.csv: 76 kB
  Indonesia median mobile (Ookla Feb 2026)    0.09 s  within 2.5 s
  Lighthouse mobile test profile              0.97 s  within 2.5 s
  Weak signal (example: 0.4 Mbit/s)           2.72 s  OVER 2.5 s
```

Periksa satu baris dengan tangan memakai Persamaan (10.4). Versi pertama berbobot $S = 32 + 1800 + 180 + 420 + 60 = 2492$ kB. Pada profil Lighthouse, $k \cdot RTT = 4 \times 0.150 = 0.6$ detik dan $8S/B = 8 \times 2492/1638.4 = 12.17$ detik, sehingga $t \approx 12.77$ detik — lima kali anggaran. Pada koneksi median, halaman yang sama tiba dalam 0,42 detik, dan karena itulah halaman ini tampak baik-baik saja dalam pengujian sang keponakan di kota. Versi ramping, 76 kB, tiba dalam anggaran baik pada median maupun pada profil uji, dan hanya sedikit melampauinya pada contoh sinyal lemah (2,72 detik). **Gambar 10.2** menggambar keenam perkiraan pada skala logaritmis.



Gambar 10.2 Perkiraan waktu tiba halaman pemesanan versi pertama (2.492 kB) dan versi ramping (76 kB) pada tiga jaringan, dari Persamaan (10.4) dengan empat perjalanan pulang-pergi; garis putus-putus adalah anggaran 2,5 detik untuk LCP yang baik. Sumbu waktu berskala logaritmis.

> **PERINGATAN:** Menguji hanya di ponsel cepat, di Wi-Fi kantor, di kota, menyembunyikan hampir semua masalah yang diukur bab ini. Ujilah di ponsel paling lambat yang kemungkinan dimiliki pengguna Anda, dengan koneksi lemah yang disimulasikan atau yang nyata, di luar ruangan.

10.7 Mengujinya

Pengujian melindungi bagian-bagian yang diandalkan pemeriksaan peramban. **Kode 10.13** memeriksa rumus Persamaan (10.4) terhadap nilai yang dihitung dengan tangan (empat perjalanan pulang-pergi 150 ms ditambah 200 kB pada 1,6 Mbit/s adalah $0.6 + 1.0 = 1.6$ detik), bahwa kedua templat terender dalam kedua bahasa tanpa tempat isian yang tersisa, dan bahwa halaman responsif — dan hanya halaman itu — memiliki baris *viewport* dan ruas `tel`. **Kode 10.14** melakukan tujuh pemeriksaan yang sama dalam Java.

Kode 10.13 `test_ch10.py`: pengujian perkiraan dan perenderan halaman.

```
"""Pemeriksaan perender halaman dan perkiraan waktu (python -m pytest -q)."""
from budget import seconds
from pages import render


def test_estimate_formula():
    # 4 x 150 ms pulang-pergi + 200 kB pada 1,6 Mbit/s = 0,6 + 1,0 detik
    assert abs(seconds(200, 1.6, 150) - 1.6) < 1e-9


def test_both_languages_and_no_placeholders_left():
    for page in ("naive", "responsive"):
        for lang in ("en", "id"):
            html = render(page, lang)
            assert "{{" not in html and f'lang="{lang}"' in html
    assert "Pesan ojek" in render("responsive", "id")


def test_responsive_page_has_viewport_and_tel_input():
    html = render("responsive", "en")
    assert 'name="viewport"' in html and 'type="tel"' in html
    assert 'name="viewport"' not in render("naive", "en")
```

Kode 10.14 `Ch10Test.java`: pemeriksaan yang sama dalam Java biasa.

```java
/** Pemeriksaan perender halaman dan perkiraan waktu (java -cp out Ch10Test). */
public class Ch10Test {
    public static void main(String[] args) throws Exception {
        int failed = 0;
        // 4 x 150 ms pulang-pergi + 200 kB pada 1,6 Mbit/s = 0,6 + 1,0 detik
        if (Math.abs(Budget.seconds(200, 1.6, 150) - 1.6) > 1e-9) failed++;
        for (String page : new String[] {"naive", "responsive"})
            for (String lang : new String[] {"en", "id"}) {
                String html = Pages.render(page, lang);
                if (html.contains("{{")
                    || !html.contains("lang=\"" + lang + "\"")) failed++;
            }
        String r = Pages.render("responsive", "en");
        if (!r.contains("name=\"viewport\"") || !r.contains("type=\"tel\""))
            failed++;
        if (Pages.render("naive", "en").contains("name=\"viewport\"")) failed++;
        System.out.println(failed == 0 ? "all 7 checks passed"
                                       : failed + " failed");
    }
}
```

`python -m pytest -q` mencetak

```
3 passed in 0.01s
```

dan `java -cp out Ch10Test` mencetak

```
all 7 checks passed
```

10.8 Membuatnya Lebih Baik

Halaman responsif lulus setiap pemeriksaan otomatis, tetapi perkiraan waktu memperlihatkan satu kelemahan yang tersisa, dan pemeriksaan itu menyembunyikan beberapa kelemahan lain.

Sinyal lemah. Dengan 2,72 detik, halaman ramping melampaui anggaran pada contoh sinyal lemah, dan fotonya adalah sebagian besar bobotnya (70 dari 76 kB). Tiga perubahan, masing-masing murah, membawanya ke dalam anggaran: muat foto *setelah* formulir dengan `loading="lazy"` agar teks dan formulir, konten bermakna terbesar di halaman, tiba lebih dahulu; kirim foto yang lebih kecil ke layar sempit dengan `srcset`; atau hapus saja fotonya, karena pelanggan yang memesan ojek tidak memerlukan gambar bangku. Tanpa foto, $S = 6$ kB dan $t \approx 4 \times 0.3 + 8 \times 6/400 = 1.32$ detik.

Kunjungan kedua. Perjalanan pulang-pergi mendominasi halaman kecil pada jalur lambat: dari 2,72 detik, 1,2 detik adalah empat perjalanan pulang-pergi. Halaman yang tersimpan di *cache* ponsel melewatkan semuanya. *Header* HTTP untuk *caching* lembar gaya dan foto, dan — bagi halaman yang harus berfungsi di bawah pohon tanpa sinyal sekalipun — sebuah *service worker* kecil yang menyajikan salinan terakhir secara luring, adalah langkah berikutnya.

Yang tak terlihat oleh pemeriksaan. Pemeriksaan otomatis tidak menguji apakah urutan ruas formulir sesuai dengan cara orang memikirkan perjalanan, apakah “Jemput saya di” dipahami, atau apakah halaman dapat dipakai dalam posisi mendatar dengan ponsel di dudukan dasbor. Bawalah halaman itu ke pangkalan, serahkan ponselnya kepada Mas Yoga dan Bu Ani, lalu amati, sebagaimana diuraikan Bab 7.

Kuota dan hemat data. Banyak pengguna di Indonesia membeli paket data dengan kuota terbatas, dan setiap megabita yang tidak perlu adalah uang yang keluar dari saku mereka, bukan

dari saku pemilik situs. Halaman seberat 2,5 MB yang dibuka dua kali sehari menghabiskan sekitar 150 MB sebulan hanya untuk satu halaman pemesanan. Anggaran kinerja karena itu bukan sekadar urusan kecepatan, melainkan juga urusan keadilan: halaman yang ringan menghormati kuota pelanggan. Pertimbangkan juga mode hemat data yang tersedia di banyak peramban seluler; halaman yang tetap berfungsi ketika gambar tidak dimuat akan tetap berguna bagi pengguna yang menyalakan mode itu.

10.9 Memakainya Sungguhan

- **Mulailah dari ponsel.** Rancang halaman Anda pada 360 piksel lebih dahulu, lalu pada 320, dan baru setelah itu tambahkan titik henti untuk layar yang lebih besar ketika konten tampak meregang.
- **Selalu sertakan baris *viewport*.** Hanya satu baris, dan tanpanya setiap upaya seluler lain menciut menjadi sepertiganya.
- **Tetapkan anggaran dan tuliskan.** Misalnya: "pada profil seluler Lighthouse halaman pemesanan tiba dalam 2,5 detik", yang menurut Persamaan (10.4) berarti paling banyak sekitar 389 kB. Setiap foto, fon, atau skrip baru harus muat atau menggantikan sesuatu.
- **Ukurlah pada lebar yang dimiliki pengguna Anda.** Jalankan `check_responsive.py` terhadap halaman Anda sendiri; tambahkan lebar ponsel yang Anda jumpai dalam riset pengguna.
- **Pakai penolong dari platform.** Jenis masukan, token *autocomplete*, dan `inputmode` tidak memakan biaya apa pun dan menghemat detik bagi setiap pengguna pada setiap kunjungan.
- **Pikirkan saat pemakaian.** Pengemudi di bawah pohon, ibu di pintu puskesmas, pelanggan di dalam angkot yang melaju: tugas singkat, interupsi, sinar matahari. Biarkan tindakan utama tetap terlihat dan halaman tetap pendek.

10.10 IMK dalam Praktik: Profitina

Pengguna Profitina adalah pemilik merek dan pemasar yang memeriksa seberapa siap situs web mereka untuk dikutip oleh mesin jawaban AI, dan banyak di antara mereka melihat skornya di ponsel di sela-sela rapat. **Dasbor** dirancang sebagai kokpit justru agar keadaan yang penting muat di layar kecil: satu permintaan mengumpulkan notifikasi, kanal yang diikuti, peringkat, dan aktivitas, dan setiap butir hanya berjarak satu ketukan dari bagiannya. Di ponsel, ini menghindarkan rangkaian menu yang mungkin masih dapat ditoleransi tata letak desktop.

Komunitas disematkan sebagai *widget* di seluruh situs pemasaran profitina.com dan di dalam aplikasi, yang merupakan persoalan desain responsif dalam arti *container query* di Subbab 10.3.3: komponen yang sama harus berfungsi di bilah samping yang sempit pada artikel desktop dan selebar penuh di ponsel. Umpannya dikirim dengan **Server-Sent Events**, satu koneksi berumur panjang yang dipakai peladen untuk mendorong kiriman baru; di jaringan seluler ini menghindarkan perjalanan pulang-pergi berulang yang akan dibayar bila memakai *polling*, yaitu suku $k \cdot RTT$ dalam Persamaan (10.4) yang dibayar sekali alih-alih setiap beberapa detik. **Pengiriman optimistis** — kotak tulis langsung kosong, kiriman langsung muncul, dan teksnya dipulihkan jika pengiriman gagal — menjaga interaksi tetap tanggap dalam arti INP bahkan ketika jaringan lambat. Studi kasus yang dapat dijalankan di buku ini tetaplah laundry di Bab 16; produk itu hanya memperlihatkan di mana prinsip-prinsip yang sama bertemu pengguna nyata.

10.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Tanpa baris *viewport*.** Halaman ditata pada 980 piksel dan diciutkan menjadi sepertiganya. Tambahkan `<meta name="viewport" content="width=device-width, initial-scale=1">`.
- **Lebar tetap.** `width: 960px` pada badan halaman atau tabel merusak *reflow* pada setiap lebar yang lebih kecil. Pakai lebar cair dan `max-width`.
- **Titik henti menurut perangkat.** Mengejar lebar ponsel tahun lalu menghasilkan tata letak yang rusak di ponsel tahun ini. Pilih titik henti di tempat konten membutuhkannya.
- ***Placeholder* sebagai label.** Ia lenyap saat mengetik dan sering berkontras rendah. Pakai label sungguhan di atas ruas.
- **Papan ketik yang salah.** Ruas telepon berjenis `text` memaksa setiap pengguna berganti papan ketik. Pakai `type="tel"`, `type="email"`, dan `autocomplete`.
- **Menguji pada median.** Halaman yang hanya diuji di ponsel cepat dan koneksi cepat akan gagal bagi pengguna yang paling sedikit ruang geraknya. Ujilah terhadap profil lambat.
- **Interaksi yang hanya lewat *hover*.** Menu yang terbuka saat penunjuk melayang dan *tooltip* yang memerlukan tetikus tidak ada di layar sentuh. Buat semuanya berfungsi dengan ketukan.
- **Membaca statistik tanpa bertanya apa yang dihitung.** Tayangan halaman di sebagian situs bukanlah orang; median bukanlah ekor sebaran.

10.12 Ringkasan Bab

- Desain responsif menyesuaikan satu antarmuka dengan setiap layar melalui proporsi cair, media yang lentur, serta *media query* atau *container query*; desain *mobile-first* dimulai dari layar terkecil dan kondisi terlemah.
- Di Indonesia, 81,72% penduduk memakai internet (APJII 2026); pangsa tayangan halaman StatCounter (38,98% seluler, 60,59% desktop pada Agustus 2026; Android 77,07% dari seluler) dan kecepatan seluler median Ookla (59,18 Mbit/s, Februari 2026) menggambarkan hal yang berbeda dan harus dibaca dengan cermat.
- Piksel CSS adalah piksel perangkat dibagi rasio piksel (Persamaan (10.1)). Tanpa baris *viewport*, ponsel menata halaman pada 980 piksel dan menskalakannya dengan s (Persamaan (10.2)), menciutkan teks dan target menjadi $s \cdot h_{css}$ (Persamaan (10.3)).
- Kebanyakan orang memakai ponsel dengan satu tangan dan ibu jari; letakkan tindakan yang sering dipakai dalam jangkauan ibu jari dan jaga target tetap besar.
- Jenis masukan, token *autocomplete*, dan label yang terhubung membuat formulir ponsel cepat dan aksesibel (WCAG 1.3.5).
- Core Web Vitals menilai halaman baik bila LCP ≤ 2,5 detik, INP ≤ 200 ms, dan CLS ≤ 0,1 pada persentil ke-75. Waktu tiba kira-kira $k \cdot RTT + 8S/B$ (Persamaan (10.4)); anggaran kinerja menjaganya dalam sasaran pada profil uji yang lambat.
- Program bab ini mengukur dua halaman pada empat lebar (**Tabel 10.1**, **Gambar 10.1**) dan memperkirakan waktu muatnya (**Gambar 10.2**): halaman desktop menciut menjadi teks 4 piksel dan 2,5 MB, halaman *mobile-first* mempertahankan teks 16 piksel, target 48 piksel, dan 76 kB.

10.13 Soal Latihan

1. Berapa ukuran efektif teks isi 13 piksel halaman pertama di ponsel 320 piksel? Dan di ponsel 412 piksel? Pakai Persamaan (10.2) dan Persamaan (10.3).

2. Dengan Persamaan (10.4) dan $k = 4$, berapa halaman terbesar yang tiba dalam 2,5 detik pada profil Lighthouse? Pada contoh sinyal lemah? Apa yang dikatakan jawaban kedua tentang foto?
3. Seorang rekan membaca angka StatCounter untuk Agustus 2026 dan mengusulkan agar halaman pemesanan dirancang "desktop lebih dahulu, karena 60% lalu lintas Indonesia adalah desktop". Jawablah dalam satu paragraf pendek, dengan memakai apa yang dihitung setiap sumber.
4. Tambahkan baris `Good 4G (example),10,50` ke `networks.csv` dan ramalkan dua baris baru keluaran `budget.py` sebelum menjalankannya.
5. Berikan kepada setiap ruas dari keempat ruas formulir pemesanan `type`, `autocomplete`, dan, bila berguna, `inputmode` yang terbaik, dan jelaskan satu keadaan ketika *autocomplete* sebaiknya dimatikan.
6. Halaman responsif beralih ke dua kolom pada 700 piksel. Uraikan cara Anda menguji apakah 700 adalah titik henti yang baik untuk konten ini, dan sebutkan satu tanda bahwa nilainya terlalu kecil dan satu tanda bahwa nilainya terlalu besar.

10.14 Penyelesaian Soal Latihan

1. Di ponsel 320 piksel, $s = 320/980 = 0.327$, sehingga teks isi tampak sebesar $0.327 \times 13 = 4.24$ piksel. Di ponsel 412 piksel, $s = 412/980 = 0.420$ dan teksnya $0.420 \times 13 = 5.47$ piksel. Bahkan di ponsel besar, teks itu hanya sekitar sepertiga dari 16 piksel yang nyaman; perbaikannya adalah baris *viewport*, bukan ponsel yang lebih besar.

2. Selesaikan $2.5 = 4 \cdot RTT + 8S/B$ untuk S: $S = (2.5 - 4 \cdot RTT) \cdot B/8$. Untuk profil Lighthouse, $S = (2.5 - 0.6) \times 1638.4/8 = 389.1$ kB. Untuk contoh sinyal lemah, $S = (2.5 - 1.2) \times 400/8 = 65.0$ kB. Pada sinyal lemah, seluruh halaman — teks, gaya, dan gambar — harus lebih ringan daripada satu foto biasa; foto harus kecil, dimuat belakangan, atau ditiadakan.

3. StatCounter menghitung tayangan halaman pada situs web yang memasang pelacaknya; ia tidak menghitung orang, dan tidak menghitung waktu yang dihabiskan di aplikasi. Desktop kantor menayangkan banyak halaman web, dan banyak pengguna ponsel menghabiskan waktunya di aplikasi, sehingga *tayangan halaman* desktop dapat mendominasi padahal sebagian besar *orang* memakai ponsel, sebagaimana diisyaratkan survei APJII atas orang. Pengguna halaman pemesanan adalah pengemudi dan tetangga yang akan membukanya di ponsel, seperti diperlihatkan adegan pertama bab ini; riset pengguna kita sendiri, bukan pangsa tayangan halaman nasional, yang seharusnya menentukan. Merancang *mobile-first* tidak merugikan apa pun di desktop, karena tata letaknya melebar melalui *media query*.

4. Untuk versi pertama: $4 \times 0.050 + 8 \times 2492/10000 = 0.2 + 1.99 = 2.19$ detik, dalam anggaran. Untuk versi ramping: $0.2 + 8 \times 76/10000 = 0.2 + 0.06 = 0.26$ detik. Menjalankan `budget.py` mencetak kedua nilai ini, 2.19 s dan 0.26 s, keduanya "within 2.5 s". Halaman yang berat lulus pada koneksi yang baik, dan persis karena itulah pengujian yang hanya pada koneksi baik menyesatkan.

5. Nama: `type="text"`, `autocomplete="name"`. Telepon: `type="tel"`, `autocomplete="tel"` (papan angka muncul). Penjemputan: `type="text"`, `autocomplete="street-address"`, karena sering kali alamat pengguna sendiri. Tujuan: `type="text"` tanpa token *autocomplete*, karena tidak ada token baku yang berarti "tempat tujuan saya"; `enterkeyhint="go"` dapat memberi label pada tombol *enter* papan ketik. *Autocomplete* sebaiknya dimatikan (`autocomplete="off"`, atau lebih baik `one-time-code` untuk kode) bagi nilai yang tidak boleh diisi dari ingatan, seperti kode sekali pakai atau ruas yang berbahaya bila terisi nilai tersimpan yang sudah usang.

6. Buka halaman di peramban desktop dan lebarkan jendela perlahan-lahan dari 320 piksel sambil mengamati daftar pengemudi dan formulir, atau jalankan `check_responsive.py` dengan daftar lebar setiap 20 piksel dan periksa tangkapan layarnya. Jika tepat di bawah 700 kartu satu kolom sudah sangat lebar dengan konten pendek dan banyak ruang kosong, atau baris teks melebihi sekitar 75 karakter, titik hentinya terlalu besar. Jika tepat di atas 700 kartu dua kolom begitu sempit sehingga nomor pelat terlipat ke baris kedua, atau formulir dua kolom menjepit labelnya, titik hentinya terlalu kecil. Titik henti yang tepat adalah tempat tata letak satu kolom mulai tampak meregang sementara tata letak dua kolom masih tampak lapang.

Daftar Pustaka

[1] E. Marcotte, "Responsive Web Design," *A List Apart*, no. 306, 25 Mei 2010. [Daring]. Tersedia: https://alistapart.com/article/responsive-web-design/

[2] L. Wroblewski, *Mobile First*. New York, NY, USA: A Book Apart, 2011.

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

[4] StatCounter Global Stats, "Desktop vs Mobile vs Tablet Market Share Indonesia," data Agustus 2026, diakses Sep. 2026. [Daring]. Tersedia: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/indonesia

[5] StatCounter Global Stats, "Mobile Operating System Market Share Indonesia," data Agustus 2026, diakses Sep. 2026. [Daring]. Tersedia: https://gs.statcounter.com/os-market-share/mobile/indonesia

[6] Bloomberg Technoz, "Ranking Kecepatan Internet RI Versi Speedtest Global Index Baru," melaporkan data Ookla Speedtest Global Index Februari 2026. [Daring]. Tersedia: https://www.bloombergtechnoz.com/detail-news/104703/ranking-kecepatan-internet-ri-versi-speedtest-global-index-baru

[7] MDN Web Docs, "Viewport meta tag" dan "CSS container queries," Mozilla, diakses Sep. 2026. [Daring]. Tersedia: https://developer.mozilla.org/en-US/docs/Web/HTML/Guides/Viewport_meta_element

[8] S. Hoober, "How Do Users Really Hold Mobile Devices?," *UXmatters*, 18 Feb. 2013. [Daring]. Tersedia: https://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php

[9] MDN Web Docs, "HTML attribute: autocomplete" dan "inputmode," Mozilla, diakses Sep. 2026. [Daring]. Tersedia: https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Attributes/autocomplete

[10] W3C, "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, 12 Des. 2024; disahkan sebagai ISO/IEC 40500:2025, Okt. 2025. [Daring]. Tersedia: https://www.w3.org/TR/WCAG22/

[11] B. McQuade, "Defining the Core Web Vitals metrics thresholds," *web.dev*, Google, diakses Sep. 2026. [Daring]. Tersedia: https://web.dev/articles/defining-core-web-vitals-thresholds

[12] Google Chrome, "Lighthouse: Network Throttling and CPU Throttling," dokumentasi Lighthouse, diakses Sep. 2026. [Daring]. Tersedia: https://github.com/GoogleChrome/lighthouse/blob/main/docs/throttling.md

Bab 11

Aksesibilitas dan Desain Inklusif (WCAG 2.2)

Sebuah antarmuka aksesibel bila penyandang disabilitas dapat menangkapnya, mengoperasikannya, memahaminya, dan mengandalkannya — dan dalam praktik antarmuka yang aksesibel lebih baik bagi semua orang: bagi pelanggan lansia yang matanya sudah lelah, bagi ayah muda yang menggendong bayi di satu lengan, bagi siapa pun yang berada di bawah terik matahari. Bab ini mengikuti kios antrean sebuah puskesmas yang halaman web pertamanya menyingkirkan beberapa pasiennya, lalu memperbaikinya dengan Web Content Accessibility Guidelines (WCAG) 2.2. Di akhir bab Anda akan mampu menjelaskan disabilitas sebagai ketidakcocokan antara seseorang dan sebuah desain, dan desain inklusif sebagai praktik memperbaiki desainnya; menyebut apa yang diwajibkan hukum Indonesia dan European Accessibility Act; menjelajahi WCAG 2.2 — empat prinsipnya, tiga tingkatnya, dan sembilan kriteria keberhasilan barunya — serta menyebut kedudukan WCAG 3.0 pada September 2026; menghitung rasio kontras dengan tangan dan dengan kode; menjelaskan cara pembaca layar dan teknologi bantu lain membaca halaman, dan mengapa HTML bawaan lebih baik daripada ARIA; menafsirkan survei WebAIM Million 2026; serta membangun, dalam Java dan Python, pemeriksa kontras yang menyarankan warna yang lulus dan program audit yang menemukan enam kegagalan WCAG paling umum dan beberapa lainnya, lalu memeriksa halaman yang sudah diperbaiki dengan papan ketik.

11.1 Suatu Pagi di Puskesmas Sukamaju

Pukul tujuh pagi, ruang tunggu Puskesmas Sukamaju sudah penuh. Pasien mengambil nomor antrean di sebuah kios sentuh kecil di dekat pintu, memilih layanan — poli umum, poli gigi, kesehatan ibu dan anak — dan, bila mau, mengetik nomor kartu BPJS Kesehatan mereka. Kios itu menjalankan halaman web di sebuah tablet, dan halaman yang sama diterbitkan di internet agar pasien dapat mengambil nomor dari rumah.

Amatilah empat pasien memakainya. Pak Harto, tujuh puluh satu tahun, yang penglihatannya melemah karena usia, membungkuk dekat ke layar; teks petunjuk kelabu di bawah judul, pucat di atas putih, tidak terlihat olehnya, dan ia tidak tahu kotak mana yang harus diisi. Mbak Laras, seorang tunanetra, membuka halaman itu di rumah dengan TalkBack, pembaca layar bawaan Android; TalkBack hanya mengumumkan “gambar, gambar, tombol, tautan” dan tidak lebih, karena kotak-kotak layanan bukanlah tombol, sedangkan tombol cetak dan tautan bantuan hanyalah gambar tanpa teks. Mas Dimas, yang lengan kanannya patah pekan lalu, memakai halaman itu dengan tangan kiri dan papan ketik laptopnya; tombol Tab melompati ketiga kotak layanan seolah-olah kotak itu tidak ada. Dan Bu Wulan, yang menggendong bayinya di satu lengan, mengetuk ikon bantuan yang kecil dengan ibu jarinya dan meleset.

Keempatnya bukan orang yang luar biasa. Halaman itu dibuat oleh seorang relawan yang bermaksud baik dan mengujinya dengan matanya sendiri, tetikusnya sendiri, dan kedua tangannya sendiri. Kegagalannya persis kegagalan yang paling umum di web, dan masing-masing mempunyai perbaikan yang sederhana. Puskesmas juga merupakan layanan publik, dan di

Indonesia layanan publik berkewajiban hukum untuk aksesibel. Bab ini memperlihatkan cara menemukan kegagalan-kegagalan itu, memperbaikinya, dan memeriksa perbaikannya.

11.2 Gagasannya dalam Bahasa Sehari-hari

Lama sekali disabilitas digambarkan hanya sebagai sifat seseorang: seseorang "tidak dapat melihat", "tidak dapat berjalan". **Model sosial disabilitas** menambahkan separuh gambaran lainnya: orang dengan penglihatan rendah *dibuat tak berdaya oleh* halaman berteks pucat, dan pengguna kursi roda *dibuat tak berdaya oleh* tangga, sementara orang-orang yang sama tidak terhalang oleh teks besar atau jalur landai. Disabilitas muncul dari ketidakcocokan antara seseorang dan sebuah desain. Pandangan itu memberdayakan perancang, karena desain dapat diubah.

Aksesibilitas adalah sifat sebuah desain yang dapat dipakai secara efektif oleh penyandang disabilitas. **Desain inklusif** adalah praktik merancang sejak awal untuk seluruh keragaman manusia. Salah satu alat yang berguna dari karya desain inklusif Microsoft adalah *spektrum persona*: setiap keterbatasan ada dalam bentuk permanen, sementara, dan situasional [1]. Satu lengan: orang dengan perbedaan anggota tubuh (permanen), Mas Dimas dengan lengannya yang patah (sementara), Bu Wulan dengan bayinya (situasional). Kebutaan (permanen), pupil yang dilebarkan setelah pemeriksaan mata (sementara), pengemudi yang tidak boleh melihat layar (situasional). Karena itu, desain yang berfungsi bagi kasus permanen melayani kelompok yang jauh lebih besar daripada kasus permanen saja.

> **DEFINISI.** *Aksesibilitas* adalah sejauh mana penyandang disabilitas dapat menangkap, memahami, menjelajahi, dan berinteraksi dengan sebuah produk. *Teknologi bantu* (*assistive technology*) adalah perangkat keras atau lunak yang dipakai seseorang untuk mengatasi ketidakcocokan, seperti pembaca layar, pembesar layar, sakelar, kendali suara, atau peraga braille. *Desain inklusif* adalah desain yang sejak awal mempertimbangkan seluruh keragaman manusia dalam hal kemampuan, bahasa, budaya, usia, dan bentuk perbedaan manusia lainnya.

11.3 Teori dan Buktinya

11.3.1 Siapa yang terdampak

Organisasi Kesehatan Dunia (WHO) memperkirakan sekitar 1,3 miliar orang — 16% penduduk dunia, satu dari enam orang — mengalami disabilitas yang berarti, dan jumlah itu terus bertambah seiring menuanya penduduk dan makin umumnya penyakit kronis [2]. Tambahkan keterbatasan sementara dan situasional dari spektrum persona, maka bagian orang yang, pada suatu saat dalam harinya, terlayani dengan buruk oleh desain yang tidak aksesibel mendekati semua orang. Usia adalah faktor tunggal terbesar bagi banyak keterbatasan — penglihatan, pendengaran, kendali motorik halus, dan kecepatan mempelajari antarmuka baru semuanya berubah seiring usia — dan karena itulah pasien lansia sebuah puskesmas adalah yang pertama menyadari kios yang tidak aksesibel.

11.3.2 Apa yang diwajibkan hukum

Di Indonesia, **Undang-Undang Nomor 8 Tahun 2016 tentang Penyandang Disabilitas** (UU 8/2016) mengakui hak penyandang disabilitas atas aksesibilitas serta atas informasi dan komunikasi, termasuk dalam pelayanan publik [3]. **Peraturan Pemerintah Nomor 42 Tahun 2020** (PP 42/2020) melaksanakannya untuk aksesibilitas terhadap permukiman, pelayanan

publik, dan pelindungan dari bencana, dan mewajibkan penyelenggara pelayanan publik menyediakan fasilitas dan layanan yang aksesibel [4]. Puskesmas adalah layanan publik; kios antrean dan situs webnya adalah bagian dari layanan itu.

Di Uni Eropa, **European Accessibility Act** (Directive (EU) 2019/882) berlaku sejak **28 Juni 2025** bagi daftar tertentu produk dan layanan, di antaranya perdagangan elektronik, layanan perbankan konsumen, buku elektronik, komunikasi elektronik, dan sebagian angkutan penumpang, termasuk situs web dan aplikasi selulernya [5]. Ketentuan ini penting bagi pengembang Indonesia yang membangun untuk pelanggan Eropa atau untuk toko aplikasi yang berjualan di sana. Standar teknis harmonisasinya, EN 301 549, memuat kriteria keberhasilan WCAG untuk konten web [6].

Rujukan teknis di balik hampir semua peraturan semacam itu adalah WCAG. **WCAG 2.2** menjadi W3C Recommendation pada Oktober 2023 (dengan edisi yang diperbarui pada 12 Desember 2024) [7], dan pada Oktober 2025 disahkan sebagai standar internasional **ISO/IEC 40500:2025** [8], menggantikan versi ISO dari WCAG 2.0. **WCAG 3.0** sedang dikembangkan: *Working Draft* terbarunya bertanggal 10 September 2026, dan W3C menyebutnya sebagai draf yang masih memerlukan beberapa tahun kerja; ia dibangun dari *outcome*, *requirement*, dan *assertion*, bukan dari kriteria keberhasilan seperti WCAG 2, dan belum dapat dijadikan acuan kesesuaian [9]. Pada September 2026, standar yang dipakai untuk membangun dan menguji adalah WCAG 2.2 tingkat AA.

11.3.3 WCAG 2.2 dalam satu halaman

WCAG menata persyaratannya di bawah empat **prinsip**, yang diingat sebagai **POUR**: konten harus **Perceivable** atau dapat ditangkap (teks alternatif, takarir, struktur yang dapat disesuaikan, kontras yang cukup), **Operable** atau dapat dioperasikan (akses papan ketik, waktu yang cukup, tanpa risiko kejang, dapat dijelajahi, modalitas masukan selain papan ketik), **Understandable** atau dapat dipahami (terbaca, dapat diduga, bantuan masukan), dan **Robust** atau kokoh (cocok dengan teknologi bantu). Di bawah prinsip terdapat pedoman, dan di bawah pedoman terdapat **kriteria keberhasilan** (*success criteria*) yang dapat diuji, masing-masing pada salah satu dari tiga tingkat: **A** (minimum), **AA** (tingkat yang diwajibkan sebagian besar peraturan dan kebijakan), dan **AAA** (tertinggi, tidak diwajibkan secara utuh bagi seluruh situs). WCAG 2.2 memiliki 86 kriteria keberhasilan: ia menambahkan sembilan pada WCAG 2.1 dan menghapus satu, 4.1.1 *Parsing*, yang sudah usang berkat peramban modern [7]. **Tabel 11.1** mendaftar kesembilan kriteria baru itu, yang sebagian besar menolong orang dengan disabilitas kognitif, motorik, atau penglihatan rendah.

Tabel 11.1 Sembilan kriteria keberhasilan yang ditambahkan pada WCAG 2.2.

Kriteria	Tingkat	Dalam satu kalimat
2.4.11 Focus Not Obscured (Minimum)	AA	Elemen yang terfokus tidak tertutup seluruhnya oleh *header*, *footer* lengket, atau jendela sembul.
2.4.12 Focus Not Obscured (Enhanced)	AAA	Tidak ada bagian elemen terfokus yang tertutup.
2.4.13 Focus Appearance	AAA	Penanda fokus cukup besar dan cukup kontras.
2.5.7 Dragging Movements	AA	Apa pun yang dilakukan dengan menyeret juga dapat dilakukan dengan ketukan atau klik tunggal.
2.5.8 Target Size (Minimum)	AA	Target penunjuk sekurang-kurangnya 24 × 24 px CSS, atau berjarak cukup.
3.2.6 Consistent Help	A	Mekanisme bantuan muncul di tempat relatif yang sama di setiap halaman.
3.3.7 Redundant Entry	A	Informasi yang sudah dimasukkan tidak diminta lagi dalam proses yang sama.
3.3.8 Accessible Authentication (Minimum)	AA	Masuk log tidak mensyaratkan uji fungsi kognitif, seperti mengingat atau menyalin, tanpa alternatif.
3.3.9 Accessible Authentication (Enhanced)	AAA	Seperti 3.3.8, tanpa pengecualian untuk pengenalan objek dan konten pribadi.

Bab 9 dan 10 sudah memakai beberapa kriteria: 2.5.8 ukuran target, 2.5.1 dan 2.5.7 gestur dan penyeretan, 1.4.10 *reflow*, 1.3.5 tujuan masukan. Bab ini menambahkan kriteria di balik kegagalan yang paling umum: 1.1.1 *Non-text Content* (teks alternatif), 1.3.1 *Info and Relationships* (struktur seperti label dan judul), 1.4.3 *Contrast (Minimum)* dan 1.4.11 *Non-text Contrast*, 2.1.1 *Keyboard*, 2.4.4 *Link Purpose*, 2.4.7 *Focus Visible*, 3.1.1 *Language of Page*, dan 4.1.2 *Name, Role, Value*.

11.3.4 Kontras, dihitung

Teks harus menonjol dari latarnya, dan WCAG mengukurnya dengan **rasio kontras** dua warna, yang dihitung dari **luminans relatif** keduanya — kecerahan warna sebagaimana ditangkap mata, dari 0 untuk hitam sampai 1 untuk putih [7]. Perhitungannya terdiri atas tiga langkah.

Pertama, setiap kanal merah, hijau, dan biru, yang diberikan sebagai bilangan 0 sampai 255, dibagi 255 menjadi nilai c antara 0 dan 1 lalu *dilinearkan*, membatalkan kurva gamma yang dipakai layar untuk menyandikan kecerahan:

$$c_{lin} = \begin{cases} c/12.92 & \text{jika } c \le 0.04045 \\ \left(\frac{c+0.055}{1.055}\right)^{2.4} & \text{jika tidak} \end{cases} \tag{11.1}$$

Kedua, ketiga kanal linear diberi bobot menurut seberapa kuat mata menanggapi masing-masing — hijau paling kuat, biru paling lemah:

$$L = 0.2126\, R_{lin} + 0.7152\, G_{lin} + 0.0722\, B_{lin} \tag{11.2}$$

Ketiga, dengan L_1 luminans warna yang lebih terang dan L_2 luminans warna yang lebih gelap, rasio kontrasnya adalah

$$CR = \frac{L_1 + 0.05}{L_2 + 0.05} \tag{11.3}$$

Angka 0,05 memodelkan silau cahaya sekitar pada layar, dan menjaga rasio tetap terhingga untuk hitam. CR berkisar dari 1 (warna identik) sampai 21 (hitam di atas putih). Kriteria keberhasilan

1.4.3 pada tingkat AA mensyaratkan sekurang-kurangnya **4,5:1** untuk teks biasa dan **3:1** untuk teks besar (sekurang-kurangnya 18 poin, atau 14 poin tebal — sekitar 24 dan 18,66 piksel CSS); 1.4.6 pada tingkat AAA mensyaratkan 7:1 dan 4,5:1; dan 1.4.11 mensyaratkan **3:1** untuk bagian visual kontrol, seperti bingkai ruas masukan, dan untuk grafik yang bermakna.

> **CONTOH.** Teks petunjuk halaman kios pertama berwarna #9CA3AF di atas putih. Kanal merahnya 156, sehingga $c = 156/255 = 0.6118$, dan menurut Persamaan (11.1) $R_{lin} = ((0.6118 + 0.055)/1.055)^{2.4} = 0.3325$. Dengan cara yang sama $G_{lin} = 0.3663$ (dari 163) dan $B_{lin} = 0.4287$ (dari 175). Menurut Persamaan (11.2), $L = 0.2126 \times 0.3325 + 0.7152 \times 0.3663 + 0.0722 \times 0.4287 = 0.3636$. Putih memiliki $L = 1$. Menurut Persamaan (11.3), $CR = (1 + 0.05)/(0.3636 + 0.05) = 1.05/0.4136 = 2.54$. Petunjuk itu memerlukan 4,5:1 dan hanya memiliki sedikit lebih dari separuhnya — itulah sebabnya Pak Harto tidak dapat melihatnya.

Satu peringatan kecil tentang rumus ini. Ambang pelinearan ditulis 0,03928 dalam teks WCAG yang lebih lama; WCAG 2.2 memakai nilai yang benar, 0,04045. Perbedaannya tidak mengubah hasil praktis apa pun, tetapi kode yang disalin dari tutorial lama mungkin masih membawa angka lama.

11.3.5 Cara teknologi bantu membaca halaman

Pembaca layar (*screen reader*) — TalkBack di Android, VoiceOver di iOS dan macOS, NVDA dan JAWS di Windows — tidak melihat piksel. Ia membaca **pohon aksesibilitas** (*accessibility tree*), struktur yang dibangun peramban dari HTML halaman: untuk setiap elemen ada *peran* (tombol, tautan, judul, ruas teks), *nama* (teks yang didengar pengguna, seperti “Cetak nomor saya”), dan *keadaan* (ditekan, dicentang, terbuka). Pembaca layar mengucapkan atau menampilkan pohon ini dalam braille dan memungkinkan pengguna bergerak di dalamnya: dengan Tab dari satu kontrol ke kontrol berikutnya, dengan usapan di ponsel, atau dengan melompat dari judul ke judul, cara banyak pengguna pembaca layar memindai sebuah halaman.

Karena itu segalanya bergantung pada HTML yang membawa peran dan nama yang benar. Elemen `<button>` memiliki peran “tombol”, dapat dicapai dengan Tab, dan diaktifkan dengan Enter atau Spasi; `<div onclick>` yang sekadar tampak seperti tombol tidak memiliki peran, dilewati Tab, dan tidak dapat diaktifkan dari papan ketik (kegagalan 2.1.1). `<img>` memerlukan teks `alt` sebagai namanya, atau `alt=""` bila hanya hiasan; tautan atau tombol yang hanya berisi gambar tanpa `alt` sama sekali tidak bernama dan diumumkan sebagai “tautan” atau “tombol” (kegagalan 2.4.4 dan 4.1.2). Nama sebuah masukan berasal dari `<label>`-nya; *placeholder* bukan nama yang andal dan lenyap saat mengetik (1.3.1). Judul (`h1` sampai `h6`) membentuk kerangka yang dijelajahi pengguna; melompat dari `h1` ke `h4` merusaknya (1.3.1). Dan `<html lang="…">` memberi tahu pembaca layar lafal bahasa mana yang dipakai (3.1.1): halaman berbahasa Indonesia yang dibaca dengan aturan bahasa Inggris nyaris tak dapat dipahami.

ARIA (Accessible Rich Internet Applications) adalah perangkat atribut seperti `role`, `aria-label`, `aria-describedby`, dan `aria-live` yang menambah atau mengubah peran, nama, dan keadaan bila HTML saja tidak dapat menyatakannya. Aturan pertamanya, menurut panduan W3C sendiri, adalah tidak memakainya bila ada elemen HTML bawaan dengan semantik yang diperlukan [10]: `<button>` sungguhan lebih baik daripada `<div role="button" tabindex="0">` yang penangan papan ketiknya ditambahkan dengan tangan. ARIA yang dipakai secara keliru justru membuat halaman *kurang* aksesibel, seperti diperlihatkan survei di subbab berikutnya.

Teknologi bantu lain bergantung pada fondasi yang sama. **Pembesar layar** dan perbesaran peramban bergantung pada *reflow* (1.4.10) dan teks yang dapat diperbesar (1.4.4). **Akses sakelar** dan **kendali suara** bergerak melalui kontrol-kontrol yang sama dengan papan ketik; pengguna

kendali suara mengucapkan label yang terlihat ("ketuk Cetak nomor saya"), dan karena itulah 2.5.3 *Label in Name* mensyaratkan nama aksesibel memuat teks yang terlihat.

11.3.6 Wajah web pada 2026

Setiap tahun WebAIM menganalisis halaman beranda sejuta situs web teratas dengan pengujian otomatis. Laporannya pada Februari 2026 menemukan kegagalan WCAG yang terdeteksi pada **95,9%** halaman beranda, naik dari 94,8% pada 2025, dengan rata-rata **56,1** galat terdeteksi per halaman [11]. Enam jenis kegagalan menyumbang 96% dari semua galat yang terdeteksi: teks berkontras rendah (pada 83,9% halaman), gambar tanpa teks alternatif (53,1%), ruas masukan formulir tanpa label (51,0%), tautan kosong (46,3%), tombol kosong (30,6%), dan bahasa dokumen yang tidak dinyatakan (13,5%). Laporan itu juga mencatat bahwa halaman yang memakai ARIA rata-rata memiliki *lebih banyak* galat terdeteksi daripada halaman tanpa ARIA (59,1 berbanding 42), pengingat akan aturan pertama ARIA.

Dua pelajaran mengikutinya. Kegagalan yang paling umum itu sederhana, dan memperbaiki enam jenis masalah akan menghapus hampir semua galat yang terdeteksi. Dan pengujian otomatis hanya mendeteksi sebagian masalah: ia dapat melihat bahwa sebuah gambar tidak memiliki `alt`, tetapi tidak dapat menilai apakah teks `alt` bermakna, apakah urutan fokus masuk akal, atau apakah halaman dapat dipahami. Karena itu metode di subbab berikutnya memadukan alat dengan manusia.

11.4 Metode Praktis: Empat Lapis Pengujian

Pengujian aksesibilitas dilakukan dalam empat lapis, dari yang termurah sampai yang paling mengungkap. Lakukan keempatnya sebelum sebuah layanan publik ditayangkan, dan dua lapis pertama pada setiap perubahan.

1. **Pemeriksaan otomatis.** Jalankan pemeriksa pada setiap halaman: program-program bab ini, atau alat peramban seperti panel aksesibilitas Chrome DevTools, axe, atau WAVE. Alat-alat ini menangkap kegagalan mekanis pada **Tabel 11.2** dalam hitungan detik.
2. **Penelusuran dengan papan ketik.** Singkirkan tetikus dan pakai hanya Tab, Shift+Tab, Enter, Spasi, dan tombol panah. Dapatkah setiap kontrol dicapai, dalam urutan yang masuk akal? Apakah fokus selalu terlihat (2.4.7) dan tidak pernah tertutup bilah lengket (2.4.11)? Dapatkah setiap tindakan diselesaikan?
3. **Pembaca layar dan perbesaran.** Nyalakan TalkBack atau VoiceOver di ponsel, NVDA di Windows, dan dengarkan halamannya; lalu perbesar peramban ke 200% dan 400%. Apakah setiap kontrol diumumkan dengan nama dan peran yang jelas? Apakah halaman tertata ulang?
4. **Penyandang disabilitas.** Mintalah pengguna dengan disabilitas yang relevan mengerjakan tugas nyata, dengan metode Bab 7. Tidak ada cara lain untuk mengungkap apakah halaman itu *dapat dipakai*, bukan sekadar sesuai secara teknis.

Tabel 11.2 Enam kegagalan paling umum dalam WebAIM Million 2026, kriteria WCAG 2.2 di baliknya, dan pemeriksaan oleh program-program bab ini.

Kegagalan umum (WebAIM 2026)	Kriteria WCAG 2.2	Pemeriksaan otomatis di bab ini
Teks berkontras rendah	1.4.3 (dan 1.4.11 untuk bagian UI)	`contrast.py` / `Contrast.java`
Gambar tanpa teks alternatif	1.1.1	`a11y_audit`: img tanpa `alt`
Ruas formulir tanpa label	1.3.1, 4.1.2	`a11y_audit`: input tanpa label
Tautan kosong	2.4.4, 4.1.2	`a11y_audit`: tautan tanpa teks terbaca
Tombol kosong	4.1.2	`a11y_audit`: tombol tanpa teks terbaca
Bahasa halaman tidak dinyatakan	3.1.1	`a11y_audit`: `html` tanpa `lang`
(juga) Elemen non-kontrol yang dapat diklik, lompatan tingkat judul, id ganda	2.1.1, 1.3.1, 4.1.2	`a11y_audit`

11.5 Membangunnya Langkah demi Langkah

11.5.1 Halaman kios pertama

Kode 11.1 adalah halaman pertama buatan sang relawan. Halaman itu pendek, dan setiap baris yang penting mengandung kegagalan: tidak ada `lang`; logo tanpa `alt`; judul yang melompat dari `h1` ke `h4`; tiga "tombol" layanan yang terbuat dari elemen `div` yang dapat diklik; petunjuk pucat alih-alih label; tombol cetak dan tautan bantuan yang hanya berisi gambar; `id` yang ganda; dan nomor yang sedang dilayani berwarna amber di atas putih.

Kode 11.1 `queue_v1.html`: halaman kios pertama, dengan kegagalan aksesibilitas yang paling umum.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Queue</title>
<style>
body { font-family: sans-serif; margin: 16px; }
.hint { color: #9CA3AF; }
.svc { display: inline-block; width: 140px; height: 90px; }
</style>
</head>
<body>
<img src="logo.png">
<h1>Puskesmas Sukamaju</h1>
<h4>Choose a service</h4>
<div class="svc" onclick="take('A')">General clinic</div>
<div class="svc" onclick="take('B')">Dental clinic</div>
<div class="svc" onclick="take('C')">Mother and child</div>
<p class="hint">BPJS card number (optional)</p>
<input id="card" placeholder="0001234567890">
<button onclick="take('A')"><img src="print.png"></button>
<a href="/help"><img src="help.png"></a>
<p id="card">Now serving: <b style="color:#E0A100">A-019</b></p>
</body>
</html>
```

11.5.2 Pemeriksa kontras

Pasangan-pasangan warna kios didaftar dalam berkas CSV, **Kode 11.2**, masing-masing dengan jenis konten yang diwarnainya: `text` biasa (4,5:1), teks `large` (3:1), atau bagian `ui` seperti bingkai ruas (3:1).

Kode 11.2 `colors.csv`: pasangan-pasangan warna kios dan jenis konten yang diwarnai masing-masing.

```
use,fg,bg,kind
body text,#1F2937,#FFFFFF,text
hint text (first version),#9CA3AF,#FFFFFF,text
button label,#FFFFFF,#0F7C7C,text
amber number on white,#E0A100,#FFFFFF,large
field border,#D1D5DB,#FFFFFF,ui
field border (second version),#6B7280,#FFFFFF,ui
now serving on navy,#FFFFFF,#12355B,large
```

`contrast.py` (**Kode 11.3**) menerapkan Persamaan (11.1), Persamaan (11.2), dan Persamaan (11.3) dalam tiga fungsi kecil, lalu melakukan sesuatu yang lebih berguna daripada sekadar vonis: untuk pasangan yang gagal ia *menyarankan* warna yang lulus, dengan mencampur warna depan dengan hitam selangkah 5% demi 5% sampai rasionya mencapai yang diperlukan. Saran itu mempertahankan rona warna, sehingga teal sebuah merek tetap teal, hanya lebih gelap. `Contrast.java` (**Kode 11.4**) adalah program yang sama dalam Java.

Kode 11.3 `contrast.py`: rasio kontras WCAG dan warna lebih gelap yang lulus.

```
"""Rasio kontras WCAG 2.2 pasangan warna, beserta warna lebih gelap yang lulus."""
import csv
import sys
from pathlib import Path

DATA = Path(__file__).resolve().parent.parent / "data"
NEEDED = {"text": 4.5, "large": 3.0, "ui": 3.0}   # AA: 1.4.3 and 1.4.11


def channel(c):
    c = c / 255
    return c / 12.92 if c <= 0.04045 else ((c + 0.055) / 1.055) ** 2.4


def luminance(hexcolor):
    r, g, b = (int(hexcolor[i:i + 2], 16) for i in (1, 3, 5))
    return 0.2126 * channel(r) + 0.7152 * channel(g) + 0.0722 * channel(b)


def ratio(fg, bg):
    hi, lo = sorted((luminance(fg), luminance(bg)), reverse=True)
    return (hi + 0.05) / (lo + 0.05)


def darker(hexcolor, step):
    """Campur warna dengan hitam: langkah 0,05 mengurangi 5% tiap kanal."""
    return "#" + "".join(f"{round(int(hexcolor[i:i + 2], 16) * (1 - step)):02X}"
                         for i in (1, 3, 5))


def fix(fg, bg, needed):
    step = 0.0
    while ratio(darker(fg, step), bg) < needed and step < 1:
        step = round(step + 0.05, 2)
    return darker(fg, step)
```

Kode 11.3 (Lanjutan) *contrast.py: rasio kontras WCAG dan warna lebih gelap yang lulus.*

```
def main(path):
    with open(path, encoding="utf-8") as f:
        rows = list(csv.DictReader(f))
    print(f"{'use':<30}{'fg':>8}{'bg':>8}{'ratio':>7}  need  result")
    for r in rows:
        need = NEEDED[r["kind"]]
        k = ratio(r["fg"], r["bg"])
        result = "pass" if k >= need else "FAIL -> try " + fix(r["fg"], r["bg"],
                                                               need)
        print(f"{r['use']:<30}{r['fg']:>8}{r['bg']:>8}{k:>7.2f}  {need:>4}  "
              + result)


if __name__ == "__main__":
    main(sys.argv[1] if len(sys.argv) > 1 else DATA / "colors.csv")
```

Kode 11.4 `Contrast.java`: pemeriksa kontras 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.*;

/** Rasio kontras WCAG 2.2 pasangan warna, dengan warna lebih gelap yang lulus. */
public class Contrast {
    static final Map<String, Double> NEEDED = Map.of("text", 4.5, "large", 3.0,
                                                     "ui", 3.0); // 1.4.3, 1.4.11

    static double channel(int c8) {
        double c = c8 / 255.0;
        return c <= 0.04045 ? c / 12.92 : Math.pow((c + 0.055) / 1.055, 2.4);
    }

    static int part(String hex, int i) {
        return Integer.parseInt(hex.substring(i, i + 2), 16);
    }

    static double luminance(String hex) {
        return 0.2126 * channel(part(hex, 1)) + 0.7152 * channel(part(hex, 3))
             + 0.0722 * channel(part(hex, 5));
    }

    static double ratio(String fg, String bg) {
        double a = luminance(fg), b = luminance(bg);
        return (Math.max(a, b) + 0.05) / (Math.min(a, b) + 0.05);
    }

    /** Campur warna dengan hitam: langkah 0,05 mengurangi 5% tiap kanal. */
    static String darker(String hex, double step) {
        StringBuilder s = new StringBuilder("#");
        for (int i : new int[] {1, 3, 5})
            s.append(String.format("%02X", Math.round(part(hex, i) * (1 - step))));
        return s.toString();
    }

    static String fix(String fg, String bg, double needed) {
        double step = 0;
        while (ratio(darker(fg, step), bg) < needed && step < 1)
            step = Math.round((step + 0.05) * 100) / 100.0;
        return darker(fg, step);
    }
```

Kode 11.4 (Lanjutan) *Contrast.java: pemeriksa kontras yang sama dalam Java.*

```
    public static void main(String[] args) throws IOException {
        Path p = Path.of(args.length > 0 ? args[0] : "../data/colors.csv");
        List<String> lines = Files.readAllLines(p, StandardCharsets.UTF_8);
        System.out.printf("%-30s%8s%8s%7s  need  result%n", "use", "fg", "bg",
                          "ratio");
        for (String line : lines.subList(1, lines.size())) {
            String[] f = line.split(",");
            double need = NEEDED.get(f[3]), k = ratio(f[1], f[2]);
            String result = k >= need ? "pass"
                                      : "FAIL -> try " + fix(f[1], f[2], need);
            System.out.printf(Locale.US, "%-30s%8s%8s%7.2f  %4s  %s%n", f[0], f[1],
                              f[2], k, need, result);
        }
    }
}
```

11.5.3 Audit

`a11y_audit.py` (**Kode 11.5**) membaca berkas HTML dan melaporkan, beserta nomor barisnya, kegagalan-kegagalan pada **Tabel 11.2**. Ia memakai dua ekspresi reguler alih-alih pengurai HTML lengkap, agar versi Java, yang pustaka standarnya tidak memiliki pengurai HTML, dapat identik: `TAG` menemukan setiap tag pembuka dan penutup beserta atributnya, dan `PAIR` menemukan setiap elemen `a`, `button`, dan `label` beserta isinya. Aturan-aturannya adalah aturan di Subbab 11.3.5. Sebuah masukan dianggap berlabel jika ada `<label for>` yang menunjuk `id`-nya, jika ia berada di dalam `<label>`, atau jika ia memiliki `aria-label` atau `aria-labelledby`. Tautan atau tombol dianggap bernama jika isinya memuat teks, atau gambar dengan `alt` yang tidak kosong, atau jika ia memiliki `aria-label`. Tag ber-`onclick` yang bukan tautan atau tombol dilaporkan sebagai tak terjangkau papan ketik. `A11yAudit.java` (**Kode 11.6**) menerapkan kedua pola yang sama dengan `java.util.regex` dan mencetak laporan yang sama. Audit berbasis ekspresi reguler memiliki batas — ia tidak melihat gaya, skrip, atau halaman yang sudah dirender — dan Subbab 11.8 menjelaskan apa yang harus dilakukan terhadapnya.

Kode 11.5 `a11y_audit.py`: menemukan bahasa, teks alternatif, label, dan nama yang hilang, elemen yang dapat diklik tetapi tak terjangkau papan ketik, lompatan judul, dan id ganda.

```
"""Audit aksesibilitas kecil berkas HTML: kegagalan WCAG yang paling umum."""
import re
import sys

TAG = re.compile(r"<(/?)([a-zA-Z][a-zA-Z0-9]*)([^>]*)>")
ATTR = re.compile(r'([a-zA-Z-]+)\s*=\s*"([^"]*)"')
PAIR = re.compile(r"<(a|button|label)\b([^>]*)>(.*?)</\1>", re.S | re.I)


def attrs(text):
    return {k.lower(): v for k, v in ATTR.findall(text)}


def visible_text(inner):
    alts = " ".join(attrs(m.group(3)).get("alt", "")
                    for m in TAG.finditer(inner) if m.group(2).lower() == "img")
    return (re.sub(r"<[^>]+>", " ", inner) + " " + alts).strip()


def audit(html):
    found = []

    def report(pos, check, msg):
        found.append((html.count("\n", 0, pos) + 1, check, msg))

    labelled = {attrs(m.group(3)).get("for") for m in TAG.finditer(html)
                if m.group(2).lower() == "label"}
    wrapped = [m.span() for m in PAIR.finditer(html)
               if m.group(1).lower() == "label"]
    ids, last_h = {}, 0
    for m in TAG.finditer(html):
        close, name, a = m.group(1), m.group(2).lower(), attrs(m.group(3))
        if close:
            continue
        if name == "html" and not a.get("lang"):
            report(m.start(), "3.1.1", "page has no lang attribute")
        if name == "img" and "alt" not in a:
            report(m.start(), "1.1.1", f"img {a.get('src', '')} has no alt")
        if name == "input" and a.get("type") not in ("hidden", "submit", "button"):
            inside = any(s <= m.start() < e for s, e in wrapped)
            if not (a.get("id") in labelled or inside or "aria-label" in a
                    or "aria-labelledby" in a):
                report(m.start(), "1.3.1", f"input {a.get('id', '')} has no label")
        if re.fullmatch(r"h[1-6]", name):
            level = int(name[1])
            if last_h and level > last_h + 1:
```

Kode 11.5 (Lanjutan) *a11y_audit.py: menemukan bahasa, teks alternatif, label, dan nama yang hilang, elemen yang dapat diklik tetapi tak terjangkau papan ketik, lompatan judul, dan id ganda.*

```
                report(m.start(), "1.3.1",
                       f"heading jumps from h{last_h} to {name}")
            last_h = level
        if "onclick" in a and name not in ("a", "button"):
            report(m.start(), "2.1.1",
                   f"clickable <{name}> is not reachable by keyboard")
        if "id" in a:
            if a["id"] in ids:
                report(m.start(), "4.1.2", f"id \"{a['id']}\" is used twice")
            ids[a["id"]] = True
    for m in PAIR.finditer(html):
        name = m.group(1).lower()
        if name in ("a", "button") and not (visible_text(m.group(3))
                                            or "aria-label" in attrs(m.group(2))):
            report(m.start(), "2.4.4" if name == "a" else "4.1.2",
                   f"{name} has no text a screen reader can read")
    return sorted(found)


if __name__ == "__main__":
    for path in sys.argv[1:]:
        problems = audit(open(path, encoding="utf-8").read())
        print(f"{path}: {len(problems)} problem(s)")
        for line, check, msg in problems:
            print(f"  line {line:>2}  SC {check:<6} {msg}")
```

Kode 11.6 `A11yAudit.java`: audit yang sama dalam Java dengan `java.util.regex`.

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.*;
import java.util.regex.*;

/** Audit aksesibilitas kecil berkas HTML: kegagalan WCAG yang paling umum. */
public class A11yAudit {
    static final Pattern TAG =
        Pattern.compile("<(/?)([a-zA-Z][a-zA-Z0-9]*)([^>]*)>");
    static final Pattern ATTR =
        Pattern.compile("([a-zA-Z-]+)\\s*=\\s*\"([^\"]*)\"");
    static final Pattern PAIR = Pattern.compile(
        "<(a|button|label)\\b([^>]*)>(.*?)</\\1>",
        Pattern.DOTALL | Pattern.CASE_INSENSITIVE);

    record Problem(int line, String check, String msg) {}

    static Map<String, String> attrs(String text) {
        Map<String, String> a = new HashMap<>();
        Matcher m = ATTR.matcher(text);
        while (m.find()) a.put(m.group(1).toLowerCase(), m.group(2));
        return a;
    }

    static String visibleText(String inner) {
        StringBuilder alts = new StringBuilder();
        Matcher m = TAG.matcher(inner);
        while (m.find())
            if (m.group(2).equalsIgnoreCase("img"))
                alts.append(" ").append(attrs(m.group(3)).getOrDefault("alt", ""));
        return (inner.replaceAll("<[^>]+>", " ") + " " + alts).strip();
    }

    static List<Problem> audit(String html) {
        List<Problem> found = new ArrayList<>();
        java.util.function.BiConsumer<Integer, String[]> report = (pos, cm) ->
            found.add(new Problem((int) html.substring(0, pos).chars()
                                      .filter(c -> c == '\n').count() + 1,
                                  cm[0], cm[1]));
        Set<String> labelled = new HashSet<>();
        List<int[]> wrapped = new ArrayList<>();
        Matcher m = TAG.matcher(html);
        while (m.find())
            if (m.group(2).equalsIgnoreCase("label"))
                labelled.add(attrs(m.group(3)).get("for"));
```

Kode 11.6 (Lanjutan) *A11yAudit.java: audit yang sama dalam Java dengan java.util.regex.*

```
        m = PAIR.matcher(html);
        while (m.find())
            if (m.group(1).equalsIgnoreCase("label"))
                wrapped.add(new int[] {m.start(), m.end()});
        Set<String> ids = new HashSet<>();
        int lastH = 0;
        m = TAG.matcher(html);
        while (m.find()) {
            String name = m.group(2).toLowerCase();
            var a = attrs(m.group(3));
            int pos = m.start();
            if (!m.group(1).isEmpty()) continue;
            if (name.equals("html") && a.getOrDefault("lang", "").isEmpty())
                report.accept(pos, new String[] {"3.1.1",
                    "page has no lang attribute"});
            if (name.equals("img") && !a.containsKey("alt"))
                report.accept(pos, new String[] {"1.1.1",
                    "img " + a.getOrDefault("src", "") + " has no alt"});
            if (name.equals("input") && !List.of("hidden", "submit", "button")
                    .contains(a.getOrDefault("type", ""))) {
                boolean inside = wrapped.stream()
                    .anyMatch(s -> s[0] <= pos && pos < s[1]);
                if (!(labelled.contains(a.get("id")) || inside
                      || a.containsKey("aria-label")
                      || a.containsKey("aria-labelledby")))
                    report.accept(pos, new String[] {"1.3.1",
                        "input " + a.getOrDefault("id", "") + " has no label"});
            }
            if (name.matches("h[1-6]")) {
                int level = name.charAt(1) - '0';
                if (lastH > 0 && level > lastH + 1)
                    report.accept(pos, new String[] {"1.3.1",
                        "heading jumps from h" + lastH + " to " + name});
                lastH = level;
            }
            if (a.containsKey("onclick") && !name.equals("a")
                    && !name.equals("button"))
                report.accept(pos, new String[] {"2.1.1",
                    "clickable <" + name + "> is not reachable by keyboard"});
            if (a.containsKey("id") && !ids.add(a.get("id")))
                report.accept(pos, new String[] {"4.1.2",
                    "id \"" + a.get("id") + "\" is used twice"});
        }
        m = PAIR.matcher(html);
        while (m.find()) {
            String name = m.group(1).toLowerCase();
```

Kode 11.6 (Lanjutan) *A11yAudit.java: audit yang sama dalam Java dengan java.util.regex.*

```
            if ((name.equals("a") || name.equals("button"))
                    && visibleText(m.group(3)).isEmpty()
                    && !attrs(m.group(2)).containsKey("aria-label"))
                report.accept(m.start(), new String[] {
                    name.equals("a") ? "2.4.4" : "4.1.2",
                    name + " has no text a screen reader can read"});
        }
        found.sort(Comparator.comparingInt(Problem::line)
                .thenComparing(Problem::check).thenComparing(Problem::msg));
        return found;
    }

    public static void main(String[] args) throws IOException {
        for (String path : args) {
            var problems = audit(Files.readString(Path.of(path)));
            System.out.println(path + ": " + problems.size() + " problem(s)");
            for (var p : problems)
                System.out.printf("  line %2d  SC %-6s %s%n", p.line(), p.check(),
                                  p.msg());
        }
    }
}
```

11.5.4 Halaman yang diperbaiki

Kode 11.7 memperbaiki setiap kegagalan dengan HTML biasa. Halaman memiliki `lang="en"` (salinan berbahasa Indonesia memiliki `lang="id"`). Logo hanyalah hiasan di samping nama yang terlihat, sehingga diberi `alt=""` dan dilewati pembaca layar. Judul-judulnya berurutan `h1`, `h2`. Layanan-layanannya adalah elemen `<button>` sungguhan setinggi 64 piksel dengan huruf antrean di dalam teksnya yang terlihat. Ruas kartu memiliki label sungguhan, petunjuk yang dihubungkan dengan `aria-describedby` sehingga dibacakan sesudah label, `inputmode="numeric"` untuk papan angka, dan bingkai 3:1. Tombol cetak dan tautan bantuan memiliki teks yang terlihat. Nomor yang sedang dilayani berada di wilayah ber-`aria-live="polite"`, sehingga pembaca layar mengumumkan “Sedang dilayani: A-020” ketika nomornya berubah. Dan cincin fokus amber yang tebal, yang hanya digambar untuk fokus papan ketik dengan `:focus-visible`, memperlihatkan posisi papan ketik.

Kode 11.7 `queue_v2.html`: halaman kios yang diperbaiki, memakai elemen HTML bawaan, label, teks alternatif, wilayah *live*, dan cincin fokus yang terlihat.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Take a queue number - Puskesmas Sukamaju</title>
<style>
body { font: 1.125rem/1.5 "DejaVu Sans", sans-serif; margin: 0; color: #1F2937; }
header { background: #12355B; color: #fff; padding: 12px 16px; }
header img { vertical-align: middle; }
h1 { display: inline; font-size: 1.4rem; margin-left: 8px; }
main { padding: 8px 16px 16px; }
.now { font-size: 1.5rem; }
.svc { display: grid; gap: 10px; }
.svc button, .print { width: 100%; min-height: 64px; font: inherit;
  text-align: left; padding: 8px 16px; border: 2px solid #0F7C7C;
  border-radius: 12px; background: #fff; color: #1F2937; }
.print { background: #0F7C7C; color: #fff; text-align: center; }
label { display: block; margin-top: 16px; font-weight: bold; }
.hint { color: #4B5563; font-size: 1rem; }
input { width: 100%; min-height: 48px; font: inherit; padding: 8px;
  border: 2px solid #6B7280; border-radius: 8px; box-sizing: border-box; }
button:focus-visible, input:focus-visible, a:focus-visible {
  outline: 4px solid #E0A100; outline-offset: 2px; }
a { display: inline-block; min-height: 48px; padding: 12px 0; color: #12355B; }
</style>
</head>
<body>
<header><img src="logo.png" alt="" width="32" height="32">
<h1>Puskesmas Sukamaju</h1></header>
<main>
<p class="now" aria-live="polite">Now serving: <strong>A-019</strong></p>
<h2>Choose a service</h2>
<div class="svc">
<button type="button">A &middot; General clinic</button>
<button type="button">B &middot; Dental clinic</button>
<button type="button">C &middot; Mother and child</button>
</div>
<label for="card">BPJS card number (optional)</label>
<p class="hint" id="card-hint">13 digits, on the front of the card</p>
<input id="card" inputmode="numeric" autocomplete="off"
       aria-describedby="card-hint">
<p><button class="print" type="button">Print my number</button></p>
<p><a href="/help">Help and other languages</a></p>
</main>
</body>
```

Kode 11.7 (Lanjutan) *queue_v2.html: halaman kios yang diperbaiki, memakai elemen HTML bawaan, label, teks alternatif, wilayah live, dan cincin fokus yang terlihat.*

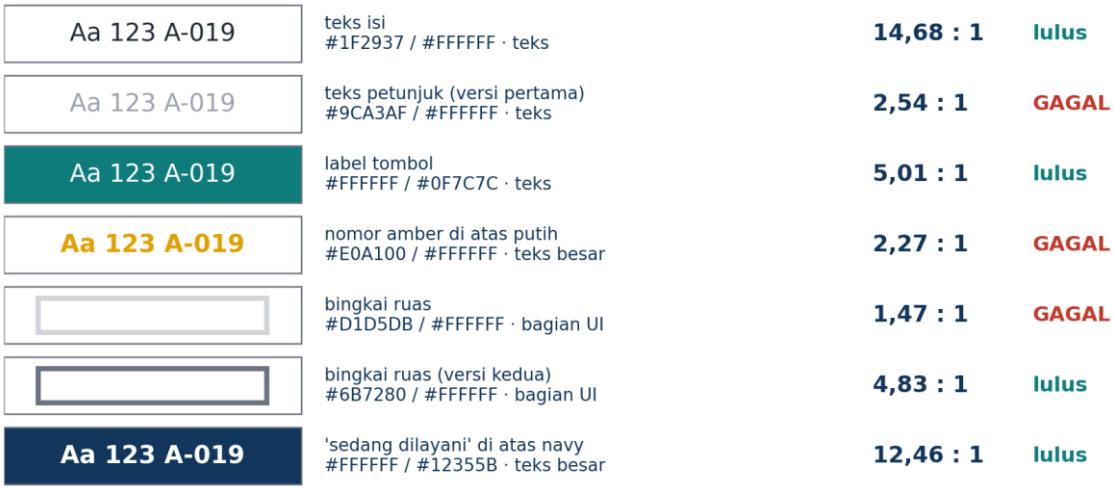
```
</html>
```

11.6 Menjalankannya

Jalankan pemeriksa kontras, `python contrast.py` atau `java -cp out Contrast`:

```
use                              fg       bg   ratio  need  result
body text                     #1F2937 #FFFFFF  14.68   4.5  pass
hint text (first version)     #9CA3AF #FFFFFF   2.54   4.5  FAIL -> try #6D727A
button label                  #FFFFFF #0F7C7C   5.01   4.5  pass
amber number on white         #E0A100 #FFFFFF   2.27   3.0  FAIL -> try #BE8900
field border                  #D1D5DB #FFFFFF   1.47   3.0  FAIL -> try #929599
field border (second version) #6B7280 #FFFFFF   4.83   3.0  pass
now serving on navy           #FFFFFF #12355B  12.46   3.0  pass
```

Tiga pasangan gagal. Teks petunjuk memiliki 2,54:1, sama dengan hitungan tangan pada Contoh di Subbab 11.3.4; pemeriksa menyarankan #6D727A, yang mencapai 4,84:1 — halaman yang diperbaiki memakai abu-abu serupa, #4B5563, untuk petunjuknya. Nomor amber di atas putih memiliki 2,27:1, bahkan di bawah 3:1 untuk teks besar; menggelapkan amber menjadi #BE8900 akan pas-pasan lulus dengan 3,10:1, tetapi halaman yang diperbaiki menyelesaikannya secara lain, dengan menampilkan nomor dalam teks gelap di atas putih dan menyimpan amber untuk cincin fokus, tempat amber berada di atas putih berdampingan dengan bingkai gelap. Bingkai ruas yang terang, #D1D5DB, memiliki 1,47:1 dan gagal pada 1.4.11; halaman yang diperbaiki memakai #6B7280 dengan 4,83:1. **Gambar 11.1** memperlihatkan ketujuh pasangan sebagaimana ditangkap mata.



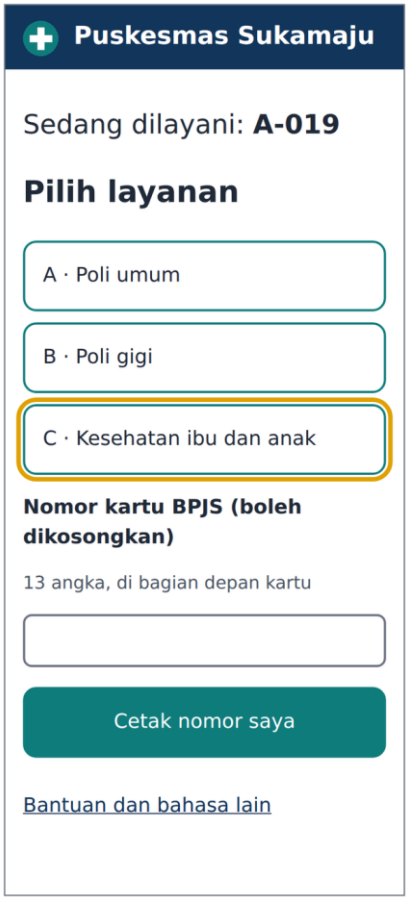
Gambar 11.1 Tujuh pasangan warna kios beserta rasio kontrasnya menurut Persamaan (11.3): tiga gagal memenuhi ambang WCAG 2.2 AA (4,5:1 untuk teks, 3:1 untuk teks besar dan bagian kontrol).

Sekarang audit kedua halaman, `python a11y_audit.py ../data/queue_v1.html ../data/queue_v2.html` (versi Java mencetak hal yang sama):

```
../data/queue_v1.html: 12 problem(s)
  line  2  SC 3.1.1  page has no lang attribute
  line 13  SC 1.1.1  img logo.png has no alt
  line 15  SC 1.3.1  heading jumps from h1 to h4
  line 16  SC 2.1.1  clickable <div> is not reachable by keyboard
  line 17  SC 2.1.1  clickable <div> is not reachable by keyboard
  line 18  SC 2.1.1  clickable <div> is not reachable by keyboard
  line 20  SC 1.3.1  input card has no label
  line 21  SC 1.1.1  img print.png has no alt
  line 21  SC 4.1.2  button has no text a screen reader can read
  line 22  SC 1.1.1  img help.png has no alt
  line 22  SC 2.4.4  a has no text a screen reader can read
  line 23  SC 4.1.2  id "card" is used twice
../data/queue_v2.html: 0 problem(s)
```

Kedua belas masalah halaman pertama berpadanan dengan keempat pasien di Subbab 11.1. Pak Harto menderita akibat label yang hilang (baris 20) sama besarnya dengan akibat petunjuk yang pucat: dengan label sungguhan, teksnya lebih besar, lebih gelap, dan mengetuknya memfokuskan ruas. Mbak Laras mendengar “gambar” untuk logo (baris 13), tidak dapat mengoperasikan ketiga `div` layanan (baris 16–18), hanya mendengar “tombol” dan “tautan” untuk tombol cetak dan tautan bantuan (baris 21–22), dan ponselnya membaca halaman berbahasa Inggris dengan lafal tebakan apa pun (baris 2). Mas Dimas tidak dapat mencapai layanan dengan Tab (baris 16–18). `id` “card” yang ganda (baris 23) berarti label yang menunjuk `card` dapat mendarat pada paragraf, bukan pada ruas. Halaman yang diperbaiki tidak memiliki masalah yang dapat ditemukan audit.

Gambar 11.2 memperlihatkannya di ponsel setelah Tab ditekan tiga kali: cincin fokus berada pada tombol layanan ketiga.



Gambar 11.2 Halaman kios yang diperbaiki pada 360 piksel CSS setelah tiga kali menekan Tab: tombol sungguhan dengan teks yang terlihat, ruas kartu berlabel dengan petunjuk dan bingkai 3:1, serta cincin fokus amber 4 piksel pada tombol yang terfokus.

11.7 Mengujinya

Kode 11.8 memeriksa rumus kontras pada kedua ujungnya (21:1 untuk hitam di atas putih, 1:1 untuk warna di atas dirinya sendiri) dan pada nilai yang sudah terkenal — #767676 di atas putih, abu-abu paling terang yang lulus 4,5:1 untuk teks biasa, menghasilkan 4,54 — dan memeriksa bahwa perbaikan yang disarankan benar-benar lulus, bahwa audit menemukan tepat dua belas masalah pada halaman pertama dan tidak satu pun pada halaman kedua, serta bahwa masukan yang dibungkus labelnya dianggap berlabel. **Kode 11.9** melakukan keenam pemeriksaan yang sama dalam Java.

Kode 11.8 `test_ch11.py`: pengujian pemeriksa kontras dan audit.

```python
"""Pemeriksaan rumus kontras dan audit (python -m pytest -q)."""
from pathlib import Path
from a11y_audit import audit
from contrast import fix, ratio

DATA = Path(__file__).resolve().parent.parent / "data"


def test_contrast_extremes_and_a_known_pair():
    assert round(ratio("#000000", "#FFFFFF"), 2) == 21.0
    assert ratio("#777777", "#777777") == 1.0
    assert round(ratio("#767676", "#FFFFFF"), 2) == 4.54   # abu-abu klasik


def test_fix_reaches_the_target():
    assert ratio(fix("#9CA3AF", "#FFFFFF", 4.5), "#FFFFFF") >= 4.5


def test_audit_counts():
    v1 = (DATA / "queue_v1.html").read_text(encoding="utf-8")
    v2 = (DATA / "queue_v2.html").read_text(encoding="utf-8")
    assert len(audit(v1)) == 12 and audit(v2) == []


def test_wrapped_label_counts_as_label():
    html = '<html lang="id"><label>Nama <input name="n"></label></html>'
    assert audit(html) == []
```

Kode 11.9 `Ch11Test.java`: pemeriksaan yang sama dalam Java biasa.

```java
import java.nio.file.Files;
import java.nio.file.Path;

/** Pemeriksaan rumus kontras dan audit (java -cp out Ch11Test). */
public class Ch11Test {
    static int failed = 0;

    static void check(boolean ok) { if (!ok) failed++; }

    public static void main(String[] args) throws Exception {
        check(Math.round(Contrast.ratio("#000000", "#FFFFFF") * 100) == 2100);
        check(Contrast.ratio("#777777", "#777777") == 1.0);
        check(Math.round(Contrast.ratio("#767676", "#FFFFFF") * 100) == 454);
        String fixed = Contrast.fix("#9CA3AF", "#FFFFFF", 4.5);
        check(Contrast.ratio(fixed, "#FFFFFF") >= 4.5);
        String v1 = Files.readString(Path.of("../data/queue_v1.html"));
        String v2 = Files.readString(Path.of("../data/queue_v2.html"));
        check(A11yAudit.audit(v1).size() == 12 && A11yAudit.audit(v2).isEmpty());
        String html = "<html lang=\"id\"><label>Nama <input name=\"n\"></label>";
        check(A11yAudit.audit(html + "</html>").isEmpty());
        System.out.println(failed == 0 ? "all 6 checks passed"
                                       : failed + " failed");
    }
}
```

`python -m pytest -q` mencetak

```
4 passed in 0.01s
```

dan `java -cp out Ch11Test` mencetak

```
all 6 checks passed
```

11.8 Membuatnya Lebih Baik

Audit otomatis yang bersih adalah awal aksesibilitas, bukan akhirnya. Telusurilah halaman yang diperbaiki dengan dua lapis berikutnya dari Subbab 11.4.

Papan ketik. Buka `queue_v2.html` dan tekan Tab berulang kali. Fokus mengunjungi, secara berurutan, ketiga tombol layanan, ruas kartu, tombol cetak, dan tautan bantuan — enam perhentian, dalam urutan yang sama dengan urutan baca pengguna yang melihat — dan cincin amber selalu terlihat. *Header* yang tetap tidak menutupi apa pun, sehingga 2.4.11 terpenuhi. Satu perbaikan masih tersisa: setelah layanan dipilih, fokus hendaknya berpindah ke hasilnya (atau pesan *live* mengumumkannya), "Nomor Anda B-024", agar pengguna pembaca layar tahu bahwa tindakannya berhasil.

Pembaca layar. Dengan TalkBack, halaman berbahasa Indonesia dibacakan sebagai "Puskesmas Sukamaju, judul tingkat 1; Sedang dilayani: A-019; Pilih layanan, judul tingkat 2; A, Poli umum, tombol; …; Nomor kartu BPJS (boleh dikosongkan), kotak isian, 13 angka, di bagian depan kartu; Cetak nomor saya, tombol; Bantuan dan bahasa lain, tautan". Titik tengah di antara huruf antrean dan nama layanan dibaca sebagai jeda. Wilayah *live* mengumumkan perubahan nomor yang sedang dilayani tanpa memindahkan fokus.

Yang tersisa bagi manusia. Apakah "Kesehatan ibu dan anak" adalah nama yang dipakai pasien, atau mereka menyebut "KIA"? Dapatkah Pak Harto, dengan matanya, menemukan tombol yang benar tanpa bantuan? Apakah karcis yang tercetak terbaca? Pertanyaan-pertanyaan ini memerlukan para pasien sendiri, dan metode Bab 7. Puskesmas juga dapat menawarkan sesuatu yang tidak dapat diberikan halaman mana pun: petugas yang mengambilkan nomor bagi pasien yang meminta. Desain inklusif mencakup manusia di sekitar antarmuka.

11.9 Memakainya Sungguhan

- **Masukkan aksesibilitas ke dalam persyaratan.** Ketika puskesmas, sekolah, atau kantor desa memesan situs web atau kios, pesanannya hendaknya menyebut "memenuhi WCAG 2.2 tingkat AA", sebagaimana tersirat dari UU 8/2016 dan PP 42/2020 untuk layanan publik. Dengan begitu keempat lapis pengujian menjadi bagian dari serah terima.
- **Otomatiskan bagian yang murah.** Jalankan `a11y_audit.py` dan `contrast.py` (atau alat arus utama) pada setiap halaman di setiap rilis, sebagai pengujian yang wajib lulus, sama seperti pengujian unit.
- **Perbaiki dengan HTML lebih dahulu.** Sebagian besar kegagalan lenyap bila elemen bawaan yang tepat dipakai: `button`, `a`, `label`, `h1`–`h6`, `alt`, `lang`. Raihlah ARIA hanya bila tidak ada elemen yang cocok.
- **Pilih warna sekali, dengan baik.** Uji palet sebuah *design system* sebelum dipakai di mana pun; warna merek yang gagal hanya menuntut satu keputusan bila diperbaiki di awal, dan seribu perbaikan bila di belakang.
- **Terbitkan pernyataan aksesibilitas.** Sebutkan standar yang dituju layanan, apa yang diketahui belum berfungsi, dan bagaimana orang dapat melaporkan masalah atau memperoleh bantuan dengan cara lain.
- **Ingat spektrum persona.** Merancang untuk pengguna tunanetra, pengguna lansia, atau pengguna yang hanya punya satu tangan bebas memperbaiki layanan bagi setiap orang yang lelah, sibuk, atau berada di luar ruangan.

11.10 IMK dalam Praktik: Profitina

Produk Profitina adalah sebuah skor — seberapa siap situs web sebuah merek untuk dikutip oleh mesin jawaban AI — dan skor hanya berguna bila setiap pengguna dapat membacanya. Hal ini menjadikan kriteria keberhasilan 1.4.1 *Use of Color* sentral: tingkat kesiapan tidak boleh disampaikan hanya dengan titik merah, amber, atau hijau, karena banyak orang, sebagian besar laki-laki, memiliki suatu bentuk defisiensi penglihatan warna, dan pembaca layar tidak mengumumkan warna. Angkanya, kata untuk tingkatnya, dan penjelasan setiap sinyal harus berupa teks. Hal yang sama berlaku bagi bagan tren mingguan di dasbor historis, yang memerlukan alternatif berupa teks atau tabel (1.1.1) serta ekspor CSV bagi pengguna yang lebih suka memakai alatnya sendiri.

Kepatuhan swalayan platform itu terhadap hukum pelindungan data pribadi Indonesia — persetujuan eksplisit, ekspor data, penggantian surel dan kata sandi, penghapusan akun, dan halaman hukum yang diterbitkan sebagai HTML biasa — juga merupakan persoalan aksesibilitas. Persetujuan yang tidak dapat ditangkap pengguna pembaca layar, atau tombol ekspor data yang tidak dapat dicapai dengan papan ketik, bukanlah persetujuan yang bermakna atau hak yang dapat dipakai. Menerbitkan halaman hukum sebagai HTML biasa, bukan sebagai gambar atau pindaian PDF, memberinya judul, teks sungguhan yang dapat diperbesar, dan atribut bahasa. Dan penyaringan Link Safety di komunitas, yang memperingatkan tentang tautan yang mencurigakan, harus menyajikan peringatannya sebagai teks yang diumumkan kepada teknologi bantu, bukan hanya sebagai ikon berwarna. Studi kasus yang dapat dijalankan di buku ini tetaplah laundry di Bab 16.

11.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **`div` dan `span` yang dapat diklik.** Tampak seperti tombol tetapi tidak memiliki peran dan akses papan ketik. Pakai `<button>` atau `<a href>`.
- **Gambar tanpa `alt` — atau dengan `alt` yang buruk.** Gambar informatif memerlukan uraian singkat tujuannya; gambar hiasan memerlukan `alt=""`. Nama berkas (“print.png”) dan “gambar dari” bukanlah uraian.
- ***Placeholder* alih-alih label.** Pakai `<label>` yang dihubungkan dengan `for`/`id`; taruh petunjuk di elemen terpisah yang dihubungkan dengan `aria-describedby`.
- **Teks dan bingkai abu-abu pucat.** Periksa setiap pasangan terhadap 4,5:1 (teks) dan 3:1 (teks besar, bagian kontrol). Gaya tembus pandang yang mengilap memerlukan pemeriksaan yang sama.
- **Menghapus garis fokus.** `outline: none` tanpa pengganti membuat papan ketik tak terlihat. Beri gaya pada `:focus-visible` sebagai gantinya.
- **Warna sebagai satu-satunya sinyal.** Tambahkan teks, atau ikon dengan teks alternatif, pada setiap status berwarna.
- **ARIA sebagai hiasan.** Peran dan label yang salah lebih buruk daripada tidak ada sama sekali. Ikuti aturan pertama ARIA.
- **Mempercayai alat begitu saja.** Laporan otomatis yang bersih itu perlu, tetapi tidak cukup. Lakukan lapis papan ketik, pembaca layar, dan pengguna.
- **Menganggap aksesibilitas sebagai langkah terakhir.** Memperbaiki belakangan jauh lebih mahal daripada merancangnya sejak awal; masukkan ke dalam persyaratan dan *design system*.

11.12 Ringkasan Bab

- Disabilitas muncul dari ketidakcocokan antara seseorang dan sebuah desain; desain inklusif memperbaiki desainnya, dan spektrum persona (permanen, sementara, situasional) memperlihatkan bahwa ia melayani jauh lebih banyak orang daripada penyandang disabilitas permanen saja. Sekitar 1,3 miliar orang, satu dari enam, mengalami disabilitas yang berarti (WHO).
- UU 8/2016 dan PP 42/2020 mewajibkan layanan publik yang aksesibel di Indonesia; European Accessibility Act berlaku sejak 28 Juni 2025; WCAG 2.2 adalah ISO/IEC 40500:2025; WCAG 3.0 masih berupa *Working Draft* (10 September 2026) dengan kerja bertahun-tahun di depannya.
- WCAG 2.2 memiliki empat prinsip (POUR), tiga tingkat, dan 86 kriteria keberhasilan, sembilan di antaranya baru (**Tabel 11.1**); AA adalah sasaran hukum yang lazim.
- Kontras dihitung dari kanal yang dilinearkan (Persamaan (11.1)), luminans relatif (Persamaan (11.2)), dan rasionya (Persamaan (11.3)); AA memerlukan 4,5:1 untuk teks dan 3:1 untuk teks besar dan bagian kontrol (**Gambar 11.1**).
- Pembaca layar membaca pohon aksesibilitas berisi peran, nama, dan keadaan; HTML bawaan menyediakannya, dan ARIA hendaknya dipakai hanya bila HTML tidak mampu.
- WebAIM Million 2026 menemukan kegagalan pada 95,9% halaman beranda; enam jenis kegagalan membentuk 96% galat yang terdeteksi (**Tabel 11.2**).
- Ujilah dalam empat lapis: otomatis, papan ketik, pembaca layar dan perbesaran, serta penyandang disabilitas. Program-program bab ini mengotomatiskan lapis pertama, dan menemukan dua belas masalah pada halaman kios pertama dan tidak satu pun pada halaman yang diperbaiki (**Gambar 11.2**).

11.13 Soal Latihan

1. Hitung dengan tangan, memakai Persamaan (11.1), Persamaan (11.2), dan Persamaan (11.3), rasio kontras #767676 di atas putih. Apakah lulus untuk teks biasa pada tingkat AA? Pada AAA?
2. Amber #E0A100 gagal di atas putih. Hitung (atau jalankan) rasionya di atas navy #12355B, serta rasio putih di atas amber. Kombinasi mana dari ketiganya yang akan Anda pakai untuk nomor “sedang dilayani”, dan mengapa?
3. Untuk setiap masalah dari kedua belas masalah dalam audit halaman pertama, sebutkan pasien mana dari keempat pasien di Subbab 11.1 yang paling terdampak, dan perbaikannya dalam satu baris.
4. Perluas `a11y_audit` agar juga melaporkan gambar yang teks `alt`-nya hanya berupa nama berkas, seperti `alt="print.png"`. Tunjukkan aturan barunya dan hasilnya pada `<html lang="en"><img src="a.png" alt="a.png"></html>`.
5. Daftarlah urutan fokus papan ketik `queue_v2.html`. Apa yang berubah jika tombol cetak diletakkan sebelum tombol-tombol layanan dalam HTML tetapi dipindahkan ke bawahnya dengan CSS?
6. Halaman masuk log sebuah kantor desa meminta pengguna mengetik huruf-huruf terdistorsi dari sebuah gambar (CAPTCHA) lalu kata sandi. Kriteria WCAG 2.2 mana yang tersentuh, alternatif apa yang memenuhinya, dan mengapa *passkey* merupakan jawaban yang baik?

11.14 Penyelesaian Soal Latihan

1. Setiap kanal bernilai 118, sehingga $c = 118/255 = 0.4627$, yang di atas 0,04045, sehingga menurut Persamaan (11.1) $c_{lin} = ((0.4627 + 0.055)/1.055)^{2.4} = 0.1812$. Ketiga kanal sama, sehingga menurut Persamaan (11.2) $L = (0.2126 + 0.7152 + 0.0722) \times 0.1812 = 0.1812$. Menurut Persamaan (11.3), $CR = (1 + 0.05)/(0.1812 + 0.05) = 1.05/0.2312 = 4.54$. Lulus AA untuk teks biasa (4,5:1) — nyaris — dan gagal AAA (7:1). Pengujian dalam **Kode 11.8** memeriksa nilai yang sama.

2. Amber di atas navy menghasilkan 5,50:1 dan putih di atas amber menghasilkan 2,27:1 (sama dengan amber di atas putih, karena rasionya simetris). Amber di atas navy lulus AA untuk teks biasa dan dengan lapang untuk teks besar, sehingga nomor amber besar di atas pita navy akan berfungsi; putih di atas amber gagal. Di halaman yang putih, pilihan paling sederhana adalah teks gelap di atas putih (14,68:1), dan karena itulah halaman yang diperbaiki memakainya serta menyimpan amber untuk cincin fokus. Pilihan apa pun juga harus menghindari pemakaian amber *saja* untuk menandai makna (1.4.1).

3. Baris 2, tanpa `lang`: Mbak Laras (lafal salah); tambahkan `lang="en"` atau `lang="id"`. Baris 13, logo tanpa `alt`: Mbak Laras mendengar "gambar"; tambahkan `alt=""`, karena namanya sudah ada di judul. Baris 15, `h1` ke `h4`: navigasi judul Mbak Laras; pakai `h2`. Baris 16–18, `div` yang dapat diklik: Mas Dimas (papan ketik) dan Mbak Laras (tanpa peran); pakai `<button>`. Baris 20, masukan tanpa label: Pak Harto (tidak dapat melihat petunjuk pucat) dan Mbak Laras (ruas tak bernama); tambahkan `<label for="card">`. Baris 21, gambar tanpa `alt` di dalam tombol, dan tombol tanpa nama: Mbak Laras; beri tombol teks yang terlihat. Baris 22, hal yang sama untuk tautan bantuan: Mbak Laras, dan Bu Wulan, yang baginya tautan teks juga dapat dibuat lebih besar; beri teks yang terlihat dan bidang 48 piksel. Baris 23, `id` ganda: semua orang yang bergantung pada label; buat id unik.

4. Tambahkan aturan sebelum pemeriksaan `img` yang sudah ada:

```
        if name == "img" and re.search(r"\.(png|jpe?g|gif|svg|webp)$",
                                       a.get("alt", ""), re.I):
            report(m.start(), "1.1.1", f"img alt is a file name: {a['alt']}")
```

Pada potongan yang diberikan, audit kini melaporkan `[(1, '1.1.1', 'img alt is a file name: a.png')]`, dan halaman kios pertama tetap memiliki dua belas masalah, karena tidak satu pun gambarnya memiliki `alt`. Di Java, uji yang sama adalah `a.getOrDefault("alt", "").matches("(?i).*\\.(png|jpe?g|gif|svg|webp)")`.

5. Urutannya: Poli umum, Poli gigi, Kesehatan ibu dan anak, ruas kartu, Cetak nomor saya, Bantuan dan bahasa lain — enam perhentian, mengikuti sumber HTML. Jika tombol cetak berada paling awal dalam HTML tetapi dipindahkan ke bawah dengan CSS (misalnya dengan `order` pada tata letak *flex*), pengguna papan ketik dan pembaca layar akan menjumpai "Cetak nomor saya" lebih dahulu, sebelum memilih layanan, sedangkan pengguna tetikus yang melihat akan menjumpainya paling akhir: urutan fokus tidak lagi sesuai dengan urutan visual, sehingga gagal pada 2.4.3 *Focus Order*. Pertahankan urutan sumber dan urutan visual tetap sama.

6. Yang tersentuh adalah 3.3.8 *Accessible Authentication (Minimum)*, tingkat AA: langkah masuk log tidak boleh bergantung pada uji fungsi kognitif — seperti menyalin karakter terdistorsi atau mengingat kata sandi — kecuali disediakan metode alternatif, mekanisme bantuan (seperti mengizinkan pengelola kata sandi dan tempel), atau uji pengenalan objek. CAPTCHA teks gagal bagi banyak pengguna dengan disleksia, penglihatan rendah, atau disabilitas kognitif.

Alternatifnya antara lain tautan atau kode masuk yang dikirim ke ponsel atau surel pengguna, mengizinkan tempel dan pengelola kata sandi, serta **passkey**, yang menggantikan kata sandi dengan pembuka kunci perangkat itu sendiri (sidik jari, wajah, atau PIN). *Passkey* tidak memerlukan ingatan maupun penyalinan, dan kini sudah dikenal luas: laporan FIDO Alliance 2026 mendapati bahwa 90% konsumen yang disurvei mengetahui *passkey* dan 75% telah mengaktifkannya pada sekurang-kurangnya satu akun [12].

Daftar Pustaka

[1] Microsoft, *Inclusive Design Toolkit*, Microsoft Design, 2016. [Daring]. Tersedia: https://inclusive.microsoft.design/

[2] World Health Organization, "Disability," lembar fakta, 7 Mar. 2023. [Daring]. Tersedia: https://www.who.int/news-room/fact-sheets/detail/disability-and-health

[3] Republik Indonesia, *Undang-Undang Nomor 8 Tahun 2016 tentang Penyandang Disabilitas*, 2016. [Daring]. Tersedia: https://peraturan.bpk.go.id/Details/37251/uu-no-8-tahun-2016

[4] Republik Indonesia, *Peraturan Pemerintah Nomor 42 Tahun 2020 tentang Aksesibilitas terhadap Permukiman, Pelayanan Publik, dan Pelindungan dari Bencana bagi Penyandang Disabilitas*, 2020. [Daring]. Tersedia: https://peraturan.bpk.go.id/Details/142301/pp-no-42-tahun-2020

[5] European Parliament and Council, "Directive (EU) 2019/882 on the accessibility requirements for products and services (European Accessibility Act)," *Official Journal of the European Union*, L 151, 7 Jun. 2019; berlaku sejak 28 Jun. 2025. [Daring]. Tersedia: https://eur-lex.europa.eu/eli/dir/2019/882/oj

[6] ETSI, "EN 301 549 V3.2.1: Accessibility requirements for ICT products and services," Mar. 2021. [Daring]. Tersedia: https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf

[7] W3C, "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, 12 Des. 2024. [Daring]. Tersedia: https://www.w3.org/TR/WCAG22/

[8] W3C, "Web Content Accessibility Guidelines (WCAG) 2.2 Approved as ISO/IEC International Standard," W3C News, 21 Okt. 2025. [Daring]. Tersedia: https://www.w3.org/news/2025/web-content-accessibility-guidelines-wcag-2-2-approved-as-iso-iec-international-standard

[9] W3C, "W3C Accessibility Guidelines (WCAG) 3.0," W3C Working Draft, 10 Sep. 2026. [Daring]. Tersedia: https://www.w3.org/TR/wcag-3.0/

[10] W3C, "Using ARIA," W3C Working Group Note, diakses Sep. 2026. [Daring]. Tersedia: https://www.w3.org/TR/using-aria/

[11] WebAIM, "The WebAIM Million: The 2026 report on the accessibility of the top 1,000,000 home pages," Feb. 2026. [Daring]. Tersedia: https://webaim.org/projects/million/

[12] FIDO Alliance, "The State of Passkeys: Global Consumer and Workforce Report," Mei 2026. [Daring]. Tersedia: https://fidoalliance.org/wp-content/uploads/2026/05/The-State-of-Passkeys-Global-Consumer-and-Workforce-Report-1.pdf

Bab 12

Desain Emosional, Kepercayaan, dan UX yang Etis

Orang tidak hanya *memakai* antarmuka; mereka merasakan sesuatu terhadapnya, dan perasaan itulah yang menentukan apakah mereka kembali, merekomendasikan layanan kepada tetangga, atau tidak pernah memercayainya lagi. Bab ini membahas perancangan untuk perasaan-perasaan itu secara jujur. Bab ini mengikuti sebuah laundry kecil yang memindahkan pemesanannya dari obrolan WhatsApp ke aplikasi web dengan keanggotaan berbayar, meniru trik-trik persuasi aplikasi besar, lalu belajar mengapa trik lebih mahal ongkosnya daripada hasilnya. Di akhir bab Anda akan mampu menjelaskan tiga tingkat desain emosional Norman dan efek estetika–kegunaan; menguraikan bagaimana kepercayaan dibangun dan hilang dalam sebuah antarmuka; mengenali *dark pattern* yang umum — desain yang menipu atau memanipulasi — menurut namanya, dan menjelaskan bagaimana Digital Services Act Uni Eropa, Federal Trade Commission Amerika Serikat, dan hukum pelindungan data pribadi Indonesia (UU PDP 27/2022, beserta peraturan pelaksanaannya PP 33/2026) memperlakukannya; merancang layar persetujuan yang dapat dipahami dan ditolak orang; serta membangun, dalam Java dan Python, sebuah *linter* yang menemukan *dark pattern* dalam uraian alur pendaftaran dan sebuah buku besar yang mengubah peristiwa persetujuan menjadi bukti tentang apa yang telah dan belum diizinkan pengguna.

12.1 Keanggotaan Bu Sri

Bu Sri mengelola Laundry Wangi, laundry kiloan di kawasan indekos dekat kampus. Bertahun-tahun pelanggannya memesan lewat WhatsApp: sebuah pesan berisi foto kantong cucian, balasan berisi harga per kilogram, dan pesan lain ketika pakaian sudah siap. Cara itu berjalan baik, dan pelanggan menyukainya: ia cepat membalas, ia jujur memberi tahu bahwa musim hujan akan memperlambat pengeringan, dan ia pernah mengembalikan dompet yang tertinggal lengkap dengan uang yang sudah dihitung.

Tahun ini keponakannya membuatkan aplikasi web dengan pelacakan pesanan dan keanggotaan, *Member Hemat*: Rp25.000 sebulan untuk harga per kilogram yang lebih murah. Untuk “menaikkan konversi”, ia meniru apa yang dilihatnya di aplikasi-aplikasi besar. Spanduk merah berteriak “Diskon laundry 50%!”, hitung mundur berbunyi “Promo berakhir dalam 09:59” — dan mulai lagi dari 09:59 pada setiap kunjungan. “Tinggal 3 kuota hari ini!” muncul setiap hari. Satu kotak yang sudah tercentang menyetujui syarat, kebijakan privasi, promosi WhatsApp, dan pembagian nomor pelanggan ke “toko mitra”, semuanya sekaligus. Tombol besar bertuliskan “Saya setuju semuanya”; menolak berarti mencari tautan kelabu kecil bertuliskan “Tidak, saya suka bayar mahal”. Bergabung cukup dua ketukan; berhenti memerlukan pesan WhatsApp ke layanan pelanggan dan enam langkah.

Selama sebulan pendaftaran naik. Lalu keluhan berdatangan. Seorang mahasiswi menemukan nomornya dalam siaran promosi sebuah bengkel motor. Beberapa anggota yang ingin berhenti mendapati bahwa mereka tidak dapat melakukannya di aplikasi dan menceritakannya di grup obrolan indekos. Seorang pelanggan tetap selama lima tahun mengucapkan kalimat yang diingat

Bu Sri: "*Kok sekarang Ibu kayak aplikasi pinjol?*" Kepercayaan yang dibangunnya bertahun-tahun lewat balasan yang jujur telah habis dalam beberapa minggu.

Bab ini menjelaskan apa yang keliru, dalam bahasa desain emosional dan kepercayaan; menamai setiap trik; memperlihatkan apa kata hukum kini tentangnya; dan membangun ulang alur pendaftaran agar persuasif dengan satu-satunya cara yang bertahan lama — dengan jelas dan adil.

12.2 Gagasannya dalam Bahasa Sehari-hari

Setiap antarmuka membuat orang merasakan sesuatu: tenang atau terburu-buru, dihargai atau dikelabui, cakap atau bodoh. **Desain emosional** adalah perancangan perasaan-perasaan itu secara sengaja. Pada wujud terbaiknya ia menghasilkan *delight* (rasa senang): aplikasi laundry yang berkata "Pakaian Anda sedang disetrika — siap pukul 16.00" membuat pelanggan merasa diperhatikan dengan biaya yang nyaris nol. **Kepercayaan** adalah kesediaan pengguna untuk mengandalkan sebuah sistem ketika ia tidak dapat memeriksa semuanya sendiri: menyerahkan nomor teleponnya, uangnya, datanya. Kepercayaan tumbuh perlahan dari banyak pengalaman kecil akan kecakapan dan kejujuran, dan dapat dihancurkan oleh satu trik yang ketahuan.

Dark pattern — istilah yang diperkenalkan Harry Brignull pada 2010, kini sering disebut *deceptive pattern* [1] — adalah desain yang membuat pengguna melakukan sesuatu yang tidak dimaksudkannya, atau yang tidak akan dilakukannya bila ia memahaminya, demi keuntungan bisnis. Hitung mundur palsu, kotak yang sudah tercentang, tautan "tidak" yang mungil, teks penolakan yang mempermalukan, dan labirin pembatalan di Laundry Wangi semuanya contoh klasik. **UX yang etis** adalah desain yang membujuk hanya dengan informasi yang benar dan pilihan yang adil, serta memperlakukan perhatian, uang, waktu, dan data pengguna sebagai milik pengguna.

> **DEFINISI.** *Dark pattern* (pola menipu) adalah pilihan desain antarmuka yang menguntungkan sebuah layanan dengan cara menipu, memaksa, atau memanipulasi pengguna agar mengambil keputusan yang tidak akan mereka ambil sebaliknya. *Persetujuan* dalam pengertian hukum data pribadi adalah pernyataan kehendak seseorang yang diberikan secara bebas, spesifik, berdasarkan informasi, dan tidak ambigu untuk pemrosesan datanya bagi tujuan yang dinyatakan, dan dapat ditarik kembali.

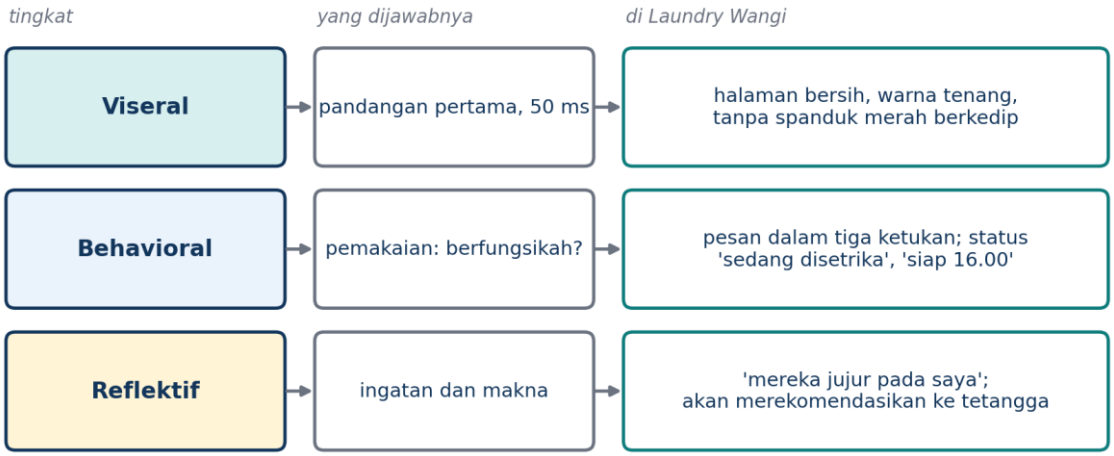
Bagi usaha kecil, taruhannya lebih besar daripada bagi aplikasi raksasa. Pelanggan sebuah laundry kiloan tinggal di lingkungan yang sama, saling mengenal, dan bertukar cerita di grup obrolan indekos, grup RT, atau grup alumni. Satu pengalaman buruk tidak berhenti pada satu orang; ia berpindah dalam hitungan menit ke puluhan orang yang belum pernah memakai aplikasinya. Sebaliknya, satu pengalaman baik — pakaian yang tertukar dan segera dikembalikan dengan permintaan maaf, atau pengingat jujur bahwa hujan akan menunda pengeringan — juga beredar dengan cara yang sama. Karena itu, bagi Bu Sri, desain yang etis bukan kemewahan moral, melainkan strategi usaha yang paling masuk akal.

12.3 Teori dan Buktinya

12.3.1 Tiga tingkat desain emosional

Norman membedakan tiga tingkat tempat sebuah produk memengaruhi orang [2]. Tingkat **viseral** adalah reaksi seketika yang belum disadari terhadap rupa: menarik atau jelek, menenangkan atau mengkhawatirkan. Reaksi ini cepat — orang membentuk kesan estetis

pertama atas sebuah halaman web dalam sekitar 50 milidetik [3]. Tingkat **behavioral** adalah pengalaman pemakaian: berfungsikah, mudahkah, apakah saya merasa memegang kendali? Tingkat **reflektif** adalah pikiran dan ingatan yang disadari: apa yang dikatakan produk ini tentang diri saya, apakah saya memercayai orang-orang di baliknya, akankah saya merekomendasikannya? **Gambar 12.1** menerapkan ketiga tingkat itu pada laundry.



Gambar 12.1 Tiga tingkat desain emosional Norman yang diterapkan pada Laundry Wangi: kesan pertama yang viseral, pengalaman behavioral saat memesan dan melacak, serta penilaian reflektif tentang kejujuran yang menentukan apakah pelanggan merekomendasikan laundry itu.

Ketiga tingkat saling memengaruhi. **Efek estetika–kegunaan** (*aesthetic–usability effect*), yang pertama kali dilaporkan Kurosu dan Kashimura pada 1995, adalah temuan bahwa orang menilai antarmuka yang menarik lebih mudah dipakai, meskipun sebenarnya tidak [4]: tingkat viseral mewarnai penilaian behavioral. Ini alasan untuk memedulikan rupa, sekaligus peringatan, karena antarmuka yang indah dapat menyembunyikan masalah kegunaan yang nyata dari pengujian yang hanya menanyakan pendapat. *Dark pattern* bekerja pada tingkat viseral dan behavioral — desakan waktu, tombol berwarna mencolok, hambatan — dan merusak tingkat reflektif, tempat kepercayaan tinggal. Pelanggan Bu Sri berbicara dari tingkat reflektif ketika membandingkan laundry itu dengan aplikasi pinjol.

12.3.2 Bagaimana kepercayaan dibangun dan hilang

Kepercayaan terhadap antarmuka bertumpu pada tiga sifat yang sama dengan kepercayaan terhadap orang: **kecakapan** (ia berfungsi, dengan andal), **itikad baik** (ia bertindak demi kepentingan saya, bukan hanya kepentingannya sendiri), dan **kejujuran** (ia mengatakan yang benar dan menepati janjinya). Antarmuka menandakan sifat-sifat ini dengan cara yang konkret: status yang akurat (“siap pukul 16.00” lalu benar-benar siap), harga yang jelas tanpa kejutan di akhir, jalan yang terlihat untuk membatalkan dan untuk keluar, mengakui kesalahan dengan terus terang, dan hanya meminta data yang diperlukan. Antarmuka kehilangan kepercayaan dengan cara yang sama konkretnya: kebohongan yang ketahuan (hitung mundur yang mulai lagi), biaya tersembunyi, data yang dipakai untuk tujuan tak terduga, dan jalan masuk yang jauh lebih mudah daripada jalan keluar.

Keamanan juga bagian dari kepercayaan, dan cara menyajikannya penting. Orang membatalkan pembelian dan masuk log ketika lupa kata sandi; survei FIDO Alliance 2026 melaporkan bahwa 47% konsumen cenderung membatalkan pembelian atau masuk log karena lupa kata sandi, sementara 90% kini mengetahui ***passkey*** dan 75% telah mengaktifkannya pada sekurang-kurangnya satu akun [5]. *Passkey* memungkinkan pengguna masuk log dengan pembuka kunci

ponselnya sendiri, yang lebih aman sekaligus tidak menegangkan — contoh desain yang memperbaiki tingkat behavioral dan reflektif sekaligus.

12.3.3 Katalog *dark pattern*

Para peneliti telah mengatalogkan *dark pattern* secara sistematis. Mathur dan rekan-rekannya menjelajahi sekitar 11.000 situs belanja pada 2019 dan menemukan 1.818 kemunculan *dark pattern* pada 1.254 situs di antaranya, sekitar 11%, termasuk penghitung waktu mundur palsu, pesan stok menipis palsu, dan *confirmshaming* [6]. Gray dan rekan-rekannya menyatukan banyak taksonomi yang saling tumpang tindih dari akademisi dan regulator dalam sebuah ontologi pengetahuan *dark pattern* yang dipresentasikan di CHI 2024, dengan strategi tingkat tinggi seperti *sneaking* (menyelinap), *obstruction* (menghalangi), *interface interference* (campur tangan antarmuka), *forced action* (tindakan paksa), dan *social engineering* (rekayasa sosial) [7]. **Tabel 12.1** mendaftar pola-pola yang muncul di Laundry Wangi, beserta alternatif etis untuk masing-masing.

Tabel 12.1 *Dark pattern* pada alur pertama Laundry Wangi, mengapa masing-masing tidak adil, dan alternatifnya yang adil.

Pola	Yang dilakukan Laundry Wangi	Mengapa tidak adil	Alternatif yang adil
Desakan waktu palsu	Hitung mundur yang mulai lagi dari 09:59	Klaim palsu yang menekan keputusan	Tampilkan tenggat sungguhan, atau tidak sama sekali
Kelangkaan palsu	"Tinggal 3 kuota hari ini!" setiap hari	Klaim kelangkaan yang palsu	Nyatakan batas yang nyata saja
Kotak sudah tercentang	Tujuan opsional sudah menyala	Diam dianggap setuju	Tujuan opsional mati secara bawaan
Persetujuan dibundel	Satu centang untuk syarat, privasi, promosi, dan pembagian	Menolak satu tujuan berarti menolak layanan	Satu pilihan per tujuan
Campur tangan antarmuka	Tombol "setuju" besar dan terisi, tautan "tidak" kecil kelabu	Penolakan sulit dilihat dan dikenai	Tombol setara untuk pilihan setara
Confirmshaming	"Tidak, saya suka bayar mahal"	Rasa bersalah alih-alih informasi	Kata-kata netral: "Nanti saja"
Sulit berhenti (*roach motel*)	Gabung 2 langkah, keluar 6 langkah lewat WhatsApp	Hambatan menjebak pengguna berbayar	Berhenti di tempat bergabung, sama mudahnya

Dua besaran membuat dua baris terakhir dapat diukur. **Asimetri hambatan** (*friction asymmetry*) sebuah langganan adalah banyaknya langkah untuk keluar dibagi banyaknya langkah untuk bergabung,

$$A = \frac{s_{out}}{s_{in}} \tag{12.1}$$

dan alur yang adil memiliki $A \leq 1$: keluar tidak pernah lebih sulit daripada bergabung. **Rasio penonjolan** (*prominence ratio*) pilihan dua tombol adalah luas tombol yang disukai bisnis dibagi luas tombol yang lain,

$$P = \frac{w_a h_a}{w_r h_r} \tag{12.2}$$

dengan w dan h lebar dan tinggi tombol setuju dan tombol tolak. Pilihan yang setara layak mendapat bobot yang setara; buku ini memakai $P \leq 2$ bersama gaya visual yang sama untuk kedua tombol sebagai patokan praktis (bukan ambang hukum). Untuk layar pertama Laundry Wangi, $A = 6/2 = 3$ menurut Persamaan (12.1), dan menurut Persamaan (12.2) $P = (320 \times 56)/(90 \times 14) = 17920/1260 = 14.2$: tombol "setuju" empat belas kali lebih besar daripada jalan untuk menolak.

12.3.4 Apa kata hukum

Dark pattern telah berpindah dari kritik desain ke dalam hukum.

Uni Eropa. Pasal 25 **Digital Services Act** (Regulation (EU) 2022/2065), yang berlaku bagi semua platform daring sejak 17 Februari 2024, melarang penyedia platform daring merancang, menata, atau mengoperasikan antarmukanya dengan cara yang menipu atau memanipulasi pengguna atau dengan cara lain secara material mendistorsi atau merusak kemampuan mereka untuk mengambil keputusan yang bebas dan berdasarkan informasi [8]. Sebuah studi CHI 2026 oleh Ahuja, Gunawan, Bielova, dan Santos memetakan 59 *dark pattern* terhadap tiga jenis pelanggaran otonomi yang disebut dalam DSA dan mengidentifikasi delapan faktor desain yang membantu menentukan kapan sebuah pola melewati batas [9] — tanda bahwa riset IMK dan regulasi kini menggarap pertanyaan yang sama. Komisi Eropa telah mengumumkan **Digital Fairness Act** untuk menangani *dark pattern*, desain yang membuat kecanduan, dan personalisasi yang tidak adil secara lebih luas; per September 2026 undang-undang itu diperkirakan diajukan pada kuartal keempat 2026 [10].

Amerika Serikat. Federal Trade Commission mendokumentasikan praktik-praktik ini dalam laporan stafnya pada 2022, *Bringing Dark Patterns to Light*, dan telah menindaknya berdasarkan kewenangan umumnya terhadap praktik yang tidak adil dan menipu. Aturan "*click-to-cancel*" tahun 2024, yang akan mewajibkan pembatalan semudah pendaftaran, dibatalkan atas dasar prosedural oleh Pengadilan Banding Sirkuit Kedelapan pada Juli 2025, beberapa hari sebelum mulai berlaku; pada 30 Januari 2026 FTC mengirimkan pemberitahuan awal rancangan aturan yang baru kepada Office of Management and Budget untuk memulai lagi prosesnya [11]. Dengan demikian, prinsip di balik Persamaan (12.1) tetap menjadi arah kebijakan, meskipun belum menjadi aturan, di Amerika Serikat.

Indonesia. Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi (UU PDP) menjadikan persetujuan yang sah secara eksplisit sebagai salah satu dasar pemrosesan data pribadi (Pasal 20); mensyaratkan persetujuan diberikan secara tertulis atau terekam, secara elektronik maupun nonelektronik, serta permintaan persetujuan dapat dibedakan secara jelas dari hal lain dan memakai bahasa yang jelas dan mudah dipahami (Pasal 22); memberi subjek data hak menarik kembali persetujuan (Pasal 9); mewajibkan pengendali mampu menunjukkan bukti persetujuan yang diandalkannya (Pasal 24); dan mensyaratkan persetujuan orang tua atau wali untuk data anak (Pasal 25) [12]. **Peraturan Pemerintah Nomor 33 Tahun 2026** (PP 33/2026), peraturan pelaksanaan UU PDP, ditetapkan pada 16 Juli 2026 dan mulai berlaku pada 16 Januari 2027; di antara banyak hal lain, peraturan itu menegaskan kembali bahwa persetujuan harus sah secara eksplisit dan diberikan untuk satu atau beberapa tujuan tertentu yang telah disampaikan pengendali [13]. Kotak Laundry Wangi yang sudah tercentang dan dibundel, serta penarikan persetujuan yang hanya lewat WhatsApp, tidak sejalan dengan setiap ketentuan itu.

> **PERINGATAN:** Persetujuan yang tidak diberikan pengguna secara aktif, tidak dapat dipahaminya, atau tidak dapat ditariknya semudah saat memberikannya adalah bukti yang

> lemah, paling-paling. Merancang layar persetujuan bukanlah hiasan di sekitar hukum; itulah *cara* persyaratan hukum sampai kepada pengguna.

12.3.5 Mengukur perasaan

Desain emosional dapat dievaluasi, bukan hanya diperdebatkan. Kuesioner seperti **User Experience Questionnaire** (UEQ), dengan skala daya tarik, kejelasan, efisiensi, keterandalan, stimulasi, dan kebaruan, serta bentuk pendeknya yang berisi delapan butir, UEQ-S, mengukur bagaimana sebuah antarmuka dialami melampaui keberhasilan tugas; SUS di Bab 7 mengukur kegunaan yang dirasakan. Sinyal perilaku lebih penting daripada perasaan yang dinyatakan bila persoalannya kepercayaan: laju pembatalan, penarikan persetujuan, keluhan, dan rekomendasi kepada orang lain. Dalam kasus Bu Sri, tingkat reflektif terukur dengan cara yang paling langsung — lewat satu kalimat di grup obrolan. Bab 14 mengubah sinyal-sinyal semacam itu menjadi metrik; aturan etis bab ini adalah bahwa desain yang menaikkan pendaftaran sambil menaikkan penarikan persetujuan dan keluhan bukanlah keberhasilan.

12.3.6 Persuasi yang menghormati pengguna

Tidak semua persuasi adalah manipulasi. Sebuah laundry wajar menginginkan anggota, dan sebuah antarmuka wajar mendorong suatu pilihan: dengan menjelaskan manfaat yang nyata, dengan menjadikan pilihan yang dianjurkan sebagai bawaan bila memang demi kepentingan pengguna (kata sandi yang lebih kuat, pengingat sebelum perpanjangan), atau dengan menampilkan bukti sosial yang benar (“dipilih 120 pelanggan di lingkungan ini”). Tiga pertanyaan memisahkan persuasi dari manipulasi. **Keterbukaan:** apakah desain itu tetap berhasil bila pengguna memahami sepenuhnya cara kerjanya? Angka penghematan yang benar tetap berhasil ketika dipahami; hitung mundur palsu berhenti berhasil begitu ketahuan. **Keterbalikan:** dapatkah pengguna dengan mudah membatalkan keputusan yang didorong desain itu? Keanggotaan yang dapat dihentikan dalam dua langkah adalah dorongan yang dapat dibalik; yang memerlukan enam langkah dan obrolan adalah jebakan. **Keselarasan:** apakah desain itu melayani tujuan pengguna sebaik tujuan bisnis? Pengingat bahwa pakaian sudah siap melayani keduanya; membagikan nomor pelanggan kepada toko mitra hanya melayani satu pihak. Desain yang lulus ketiganya boleh sepersuasif yang diinginkan pembuatnya.

12.4 Metode Praktis: Tinjauan Alur yang Adil

Sebelum alur pendaftaran, langganan, atau persetujuan dirilis, tinjaulah dengan **Tabel 12.2**. Setiap “tidak” adalah cacat yang harus diperbaiki sebelum rilis.

Tabel 12.2 Tinjauan alur yang adil: pertanyaan-pertanyaan yang harus dijawab "ya" sebelum merilis alur pendaftaran, langganan, atau persetujuan.

Pertanyaan	Periksa
Apakah setiap klaim benar?	Hitung mundur, jumlah stok, label "paling laris", harga
Apakah setiap tujuan opsional mati secara bawaan?	Promosi, pembagian data, pemrofilan, notifikasi
Adakah satu pilihan per tujuan, dalam bahasa yang lugas?	Setiap tujuan dinamai beserta artinya bagi pengguna
Apakah pilihan yang setara tampak setara?	Gaya sama untuk setuju dan tolak; rasio penonjolan $P \leq 2$ menurut Persamaan (12.2)
Apakah penolakan dirumuskan netral?	Tanpa rasa bersalah, tanpa lelucon atas beban pengguna
Apakah keluar semudah bergabung?	Asimetri hambatan $A \leq 1$ menurut Persamaan (12.1); saluran yang sama
Dapatkah persetujuan ditarik di tempat ia diberikan?	Halaman yang terlihat di aplikasi, bukan hanya obrolan dengan staf
Apakah persetujuan dicatat sebagai bukti?	Siapa, tujuan apa, kapan, versi teks mana, memberi atau menarik
Apakah total biaya jelas sebelum membayar?	Tanggal perpanjangan, harga, cara menghentikan perpanjangan otomatis, pengingat sebelum penagihan

Sebagian besar pertanyaan dapat diperiksa dengan membaca uraian alurnya; Subbab 12.5 mengubahnya menjadi program. Pertanyaan terakhir dan kebenaran klaim memerlukan orang yang mengenal bisnisnya.

12.5 Membangunnya Langkah demi Langkah

12.5.1 Menguraikan sebuah alur

Agar dapat diperiksa secara otomatis, sebuah alur harus diuraikan dalam bentuk yang dapat dibaca program. **Kode 12.1** menguraikan alur pertama Laundry Wangi dalam format baris yang sederhana: satu baris per tujuan (id, teks, wajib atau tidak, dan menyala atau tidak secara bawaan), apakah tujuan-tujuan dibundel dalam satu centang, kedua tombol beserta label, ukuran, dan gayanya, perilaku hitung mundur, banyaknya langkah untuk berlangganan dan untuk berhenti, serta di mana persetujuan dapat ditarik.

Kode 12.1 `flow_v1.txt`: alur pendaftaran dan keanggotaan pertama Laundry Wangi, diuraikan baris demi baris.

```
# Laundry Wangi sign-up and membership, first version
purpose|orders|Process my laundry orders|required|on
purpose|promo|Send me promotions by WhatsApp|optional|on
purpose|partners|Share my contact with partner shops|optional|on
bundled|yes
button|accept|I agree to everything|320|56|filled
button|reject|No thanks, I like paying more|90|14|text
countdown|resets
steps|subscribe|2
steps|cancel|6
withdraw|WhatsApp to customer service only
```

12.5.2 *Linter*-nya

`consent_lint.py` (**Kode 12.2**) membaca uraian semacam itu dan menerapkan satu aturan untuk setiap baris **Tabel 12.1**: tujuan opsional yang menyala secara bawaan adalah kotak yang sudah tercentang; `bundled|yes` adalah persetujuan yang dibundel; rasio penonjolan di atas 2 menurut

Persamaan (12.2), atau tombol tolak yang gayanya berbeda dari tombol setuju, adalah campur tangan antarmuka; label penolakan yang memuat frasa mempermalukan yang dikenal adalah *confirmshaming*; hitung mundur yang mulai lagi adalah desakan waktu palsu; asimetri hambatan di atas 1 menurut Persamaan (12.1) adalah alur yang sulit dihentikan; dan penarikan persetujuan yang "hanya" dapat dilakukan lewat saluran lain juga ditandai. Daftar frasa mempermalukan itu pendek dan memuat frasa bahasa Indonesia, karena di sanalah ia akan dipakai; memperluasnya adalah Soal Latihan 12.4. `ConsentLint.java` (**Kode 12.3**) adalah *linter* yang sama dalam Java.

Kode 12.2 `consent_lint.py`: memeriksa uraian alur dari persetujuan yang sudah tercentang dan dibundel, campur tangan antarmuka, *confirmshaming*, desakan waktu palsu, dan jalan keluar yang sulit.

```
"""Periksa alur persetujuan dan langganan dari dark pattern yang umum."""
import sys

SHAMING = ("i like paying more", "i don't want to save", "no thanks, i",
           "saya suka bayar mahal", "saya tidak mau hemat")
MAX_PROMINENCE = 2.0   # luas tombol setuju paling banyak 2x tombol tolak


def load(path):
    flow = {"purpose": [], "button": {}, "steps": {}}
    for line in open(path, encoding="utf-8"):
        line = line.strip()
        if not line or line.startswith("#"):
            continue
        f = line.split("|")
        if f[0] == "purpose":
            flow["purpose"].append(f[1:])
        elif f[0] == "button":
            flow["button"][f[1]] = (f[2], int(f[3]), int(f[4]), f[5])
        elif f[0] == "steps":
            flow["steps"][f[1]] = int(f[2])
        else:
            flow[f[0]] = f[1]
    return flow


def lint(flow):
    found = []
    for pid, text, need, default in flow["purpose"]:
        if not text.strip():
            found.append(("unclear purpose", f"purpose '{pid}' has no text"))
        if need == "optional" and default == "on":
            found.append(("pre-ticked box", f"optional purpose '{pid}' is on"))
    if flow.get("bundled") == "yes":
        found.append(("bundled consent", "one tick covers all purposes"))
    (_, aw, ah, _), (rlabel, rw, rh, rstyle) = (flow["button"]["accept"],
                                                flow["button"]["reject"])
    prominence = (aw * ah) / (rw * rh)
    if prominence > MAX_PROMINENCE or rstyle != "filled":
        found.append(("interface interference",
                      f"accept has {prominence:.1f}x the area of reject, "
                      f"reject is '{rstyle}'"))
    if any(p in rlabel.lower() for p in SHAMING):
        found.append(("confirmshaming", f"reject label: '{rlabel}'"))
    if flow.get("countdown") == "resets":
        found.append(("false urgency", "countdown restarts on every visit"))
```

Kode 12.2 (Lanjutan) *consent_lint.py: memeriksa uraian alur dari persetujuan yang sudah tercentang dan dibundel, campur tangan antarmuka, confirmshaming, desakan waktu palsu, dan jalan keluar yang sulit.*

```
    asym = flow["steps"]["cancel"] / flow["steps"]["subscribe"]
    if asym > 1:
        found.append(("hard to cancel",
                      f"cancel takes {asym:.1f}x the steps of subscribing"))
    if "only" in flow.get("withdraw", "").lower():
        found.append(("hard to withdraw", "withdraw: " + flow["withdraw"]))
    return found, prominence, asym


if __name__ == "__main__":
    for path in sys.argv[1:]:
        found, prominence, asym = lint(load(path))
        print(f"{path}: prominence {prominence:.1f}, cancel/subscribe "
              f"{asym:.1f}, {len(found)} finding(s)")
        for kind, detail in found:
            print(f"  - {kind:<23}{detail}")
```

Kode 12.3 `ConsentLint.java`: *linter* 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.*;

/** Periksa alur persetujuan dan langganan dari dark pattern yang umum. */
public class ConsentLint {
    static final List<String> SHAMING = List.of("i like paying more",
        "i don't want to save", "no thanks, i", "saya suka bayar mahal",
        "saya tidak mau hemat");
    static final double MAX_PROMINENCE = 2.0; // luas setuju paling banyak 2x tolak

    record Result(List<String[]> found, double prominence, double asym) {}

    static Map<String, Object> load(String path) throws IOException {
        Map<String, Object> flow = new HashMap<>();
        List<String[]> purposes = new ArrayList<>();
        Map<String, String[]> buttons = new HashMap<>();
        Map<String, Integer> steps = new HashMap<>();
        var lines = Files.readAllLines(Path.of(path), StandardCharsets.UTF_8);
        for (String raw : lines) {
            String line = raw.strip();
            if (line.isEmpty() || line.startsWith("#")) continue;
            String[] f = line.split("\\|");
            switch (f[0]) {
                case "purpose" -> purposes.add(Arrays.copyOfRange(f, 1, f.length));
                case "button" -> buttons.put(f[1], Arrays.copyOfRange(f, 2, 6));
                case "steps" -> steps.put(f[1], Integer.parseInt(f[2]));
                default -> flow.put(f[0], f[1]);
            }
        }
        flow.put("purpose", purposes);
        flow.put("button", buttons);
        flow.put("steps", steps);
        return flow;
    }
```

Kode 12.3 (Lanjutan) *ConsentLint.java: linter yang sama dalam Java.*

```
    @SuppressWarnings("unchecked")
    static Result lint(Map<String, Object> flow) {
        List<String[]> found = new ArrayList<>();
        for (String[] p : (List<String[]>) flow.get("purpose")) {
            if (p[1].isBlank())
                found.add(new String[] {"unclear purpose",
                    "purpose '" + p[0] + "' has no text"});
            if (p[2].equals("optional") && p[3].equals("on"))
                found.add(new String[] {"pre-ticked box",
                    "optional purpose '" + p[0] + "' is on"});
        }
        if ("yes".equals(flow.get("bundled")))
            found.add(new String[] {"bundled consent",
                                    "one tick covers all purposes"});
        var b = (Map<String, String[]>) flow.get("button");
        String[] acc = b.get("accept"), rej = b.get("reject");
        double prominence =
            (Double.parseDouble(acc[1]) * Double.parseDouble(acc[2]))
            / (Double.parseDouble(rej[1]) * Double.parseDouble(rej[2]));
        if (prominence > MAX_PROMINENCE || !rej[3].equals("filled"))
            found.add(new String[] {"interface interference", String.format(
                Locale.US, "accept has %.1fx the area of reject, reject is '%s'",
                prominence, rej[3])});
        String label = rej[0].toLowerCase();
        if (SHAMING.stream().anyMatch(label::contains))
            found.add(new String[] {"confirmshaming",
                                    "reject label: '" + rej[0] + "'"});
        if ("resets".equals(flow.get("countdown")))
            found.add(new String[] {"false urgency",
                                    "countdown restarts on every visit"});
        var s = (Map<String, Integer>) flow.get("steps");
        double asym = (double) s.get("cancel") / s.get("subscribe");
        if (asym > 1)
            found.add(new String[] {"hard to cancel", String.format(Locale.US,
                "cancel takes %.1fx the steps of subscribing", asym)});
        String w = (String) flow.getOrDefault("withdraw", "");
        if (w.toLowerCase().contains("only"))
            found.add(new String[] {"hard to withdraw", "withdraw: " + w});
        return new Result(found, prominence, asym);
    }
```

Kode 12.3 (Lanjutan) *ConsentLint.java: linter yang sama dalam Java.*

```
    public static void main(String[] args) throws IOException {
        for (String path : args) {
            Result r = lint(load(path));
            System.out.printf(Locale.US,
                "%s: prominence %.1f, cancel/subscribe %.1f, %d finding(s)%n",
                path, r.prominence(), r.asym(), r.found().size());
            for (String[] f : r.found())
                System.out.printf("  - %-23s%s%n", f[0], f[1]);
        }
    }
}
```

12.5.3 Alur kedua

Kode 12.4 adalah alur yang diperbaiki. Tujuan yang wajib menyebutkan mengapa ia wajib. Tujuan-tujuan opsional mati. Setiap tujuan memiliki centangnya sendiri. Kedua tombol berukuran sama dan bergaya terisi yang sama, dan labelnya menguraikan kedua hasilnya — “Simpan pilihan saya” dan “Hanya yang diperlukan” — tanpa rasa bersalah. Tidak ada hitung mundur. Berhenti

memerlukan dua langkah, sama seperti bergabung, dan persetujuan ditarik di halaman Akun, pada jenis layar yang sama dengan tempat ia diberikan.

Kode 12.4 `flow_v2.txt`: alur yang diperbaiki, dengan tujuan mati secara bawaan, tombol setara, tanpa hitung mundur, dan jalan keluar semudah jalan masuk.

```
# Laundry Wangi sign-up and membership, second version
purpose|orders|Process my laundry orders (needed to serve you)|required|on
purpose|promo|Send me promotions by WhatsApp|optional|off
purpose|partners|Share my contact with partner shops|optional|off
bundled|no
button|accept|Save my choices|150|48|filled
button|reject|Only what is needed|150|48|filled
countdown|none
steps|subscribe|2
steps|cancel|2
withdraw|Account page, same screen as sign-up
```

12.5.4 Buku besar persetujuan

Persetujuan bukanlah satu saat tunggal, melainkan sebuah riwayat: pelanggan memberikannya, menariknya, lalu memberikannya lagi untuk teks yang lebih baru. Kewajiban UU PDP agar pengendali mampu menunjukkan bukti persetujuan (Pasal 24) dipenuhi dengan menyimpan setiap peristiwa — siapa, tujuan apa, tindakan apa, kapan, dan versi teks persetujuan yang mana — serta menurunkan keadaan kini dari catatan itu alih-alih menimpa satu penanda tunggal. **Kode 12.5** adalah catatan kecil.

Kode 12.5 `consent_log.csv`: peristiwa persetujuan beserta waktu, pengguna, tujuan, tindakan, dan versi teks persetujuan yang ditampilkan.

```
time,user,purpose,action,text_version
2026-09-01T08:02,U17,orders,give,v2
2026-09-01T08:02,U17,promo,give,v2
2026-09-03T19:40,U17,promo,withdraw,v2
2026-09-04T10:15,U23,orders,give,v2
2026-09-04T10:15,U23,partners,give,v2
2026-09-20T07:30,U23,promo,give,v3
```

`consent_ledger.py` (**Kode 12.6**) memutar ulang peristiwa-peristiwa menurut urutan waktu, menyimpan tindakan terakhir untuk setiap pasangan pengguna dan tujuan, dan menjawab satu-satunya pertanyaan yang boleh diajukan bagian lain sistem: *bolehkah kita memakai data pengguna ini untuk tujuan ini sekarang?* Pasangan yang tidak pernah muncul dalam catatan tidak memiliki persetujuan. `ConsentLedger.java` (**Kode 12.7**) melakukan hal yang sama.

Kode 12.6 `consent_ledger.py`: memutar ulang peristiwa persetujuan menjadi keadaan kini dan menjawab apakah suatu tujuan diizinkan.

```
"""Putar ulang peristiwa persetujuan menjadi keadaan kini, sebagai bukti."""
import csv
import sys


def replay(path):
    state = {}   # (pengguna, tujuan) -> (tindakan, waktu, versi teks)
    with open(path, encoding="utf-8") as f:
        for e in sorted(csv.DictReader(f), key=lambda e: e["time"]):
            state[(e["user"], e["purpose"])] = (e["action"], e["time"],
                                                e["text_version"])
    return state


def may_use(state, user, purpose):
    return state.get((user, purpose), ("none",))[0] == "give"


if __name__ == "__main__":
    state = replay(sys.argv[1] if len(sys.argv) > 1 else "../data/consent_log.csv")
    for (user, purpose), (action, time, ver) in sorted(state.items()):
        now = "may use" if action == "give" else "must NOT use"
        print(f"{user} {purpose:<9}{action:<9}{time}  text {ver}  -> {now}")
    print("U17 promo allowed?", may_use(state, "U17", "promo"))
    print("U23 promo allowed?", may_use(state, "U23", "promo"))
```

Kode 12.7 `ConsentLedger.java`: buku besar 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.*;

/** Putar ulang peristiwa persetujuan menjadi keadaan kini, sebagai bukti. */
public class ConsentLedger {
    // (pengguna, tujuan) -> {tindakan, waktu, versi teks}
    static TreeMap<String, String[]> replay(String path) throws IOException {
        List<String> lines = Files.readAllLines(Path.of(path),
                                                StandardCharsets.UTF_8);
        List<String[]> events = new ArrayList<>();
        for (String line : lines.subList(1, lines.size()))
            events.add(line.split(","));
        events.sort(Comparator.comparing(e -> e[0]));  // menurut waktu, stabil
        TreeMap<String, String[]> state = new TreeMap<>();
        for (String[] e : events)
            state.put(e[1] + " " + e[2], new String[] {e[3], e[0], e[4]});
        return state;
    }

    static boolean mayUse(Map<String, String[]> state, String user,
                          String purpose) {
        String[] s = state.get(user + " " + purpose);
        return s != null && s[0].equals("give");
    }

    public static void main(String[] args) throws IOException {
        var state = replay(args.length > 0 ? args[0] : "../data/consent_log.csv");
        for (var e : state.entrySet()) {
            String[] k = e.getKey().split(" "), v = e.getValue();
            String now = v[0].equals("give") ? "may use" : "must NOT use";
            System.out.printf("%s %-9s%-9s%s  text %s  -> %s%n", k[0], k[1], v[0],
                              v[1], v[2], now);
        }
        System.out.println("U17 promo allowed? " + (mayUse(state, "U17", "promo")
                                                    ? "True" : "False"));
        System.out.println("U23 promo allowed? " + (mayUse(state, "U23", "promo")
                                                    ? "True" : "False"));
    }
}
```

12.6 Menjalankannya

Periksa kedua alur dengan `python consent_lint.py ../data/flow_v1.txt ../data/flow_v2.txt`; *linter* Java mencetak hal yang sama:

```
../data/flow_v1.txt: prominence 14.2, cancel/subscribe 3.0, 8 finding(s)
  - pre-ticked box         optional purpose 'promo' is on
  - pre-ticked box         optional purpose 'partners' is on
  - bundled consent        one tick covers all purposes
  - interface interference accept has 14.2x the area of reject, reject is 'text'
  - confirmshaming         reject label: 'No thanks, I like paying more'
  - false urgency          countdown restarts on every visit
  - hard to cancel         cancel takes 3.0x the steps of subscribing
  - hard to withdraw       withdraw: WhatsApp to customer service only
../data/flow_v2.txt: prominence 1.0, cancel/subscribe 1.0, 0 finding(s)
```

Alur pertama memiliki delapan temuan, satu untuk setiap masalah yang dialami pelanggan Bu Sri, serta kedua bilangan di Subbab 12.3.3: rasio penonjolan 14,2 dan asimetri hambatan 3,0. Alur kedua tidak memiliki temuan, dengan kedua rasio bernilai 1,0. **Gambar 12.2** memperlihatkan kedua layar persetujuan sebagaimana dilihat pelanggan di ponsel.



Gambar 12.2 Kedua layar persetujuan di ponsel: (a) versi pertama, dengan spanduk merah, hitung mundur yang mulai lagi, satu kotak tercentang untuk empat tujuan, tombol "setuju" yang besar, dan tautan kecil yang mempermalukan; (b) versi kedua, dengan satu pilihan per tujuan, tujuan opsional mati, penjelasan yang lugas, dan dua tombol setara.

Sekarang buku besarnya, `python consent_ledger.py`:

```
U17 orders   give     2026-09-01T08:02  text v2  -> may use
U17 promo    withdraw 2026-09-03T19:40  text v2  -> must NOT use
U23 orders   give     2026-09-04T10:15  text v2  -> may use
U23 partners give     2026-09-04T10:15  text v2  -> may use
U23 promo    give     2026-09-20T07:30  text v3  -> may use
U17 promo allowed? False
U23 promo allowed? True
```

Pelanggan U17 memberikan persetujuan untuk promosi pada 1 September dan menariknya dua hari kemudian; penarikan itu adalah peristiwa terakhir, sehingga jawaban untuk promosi adalah "must NOT use", sedangkan pesanan tetap boleh diproses. Pelanggan U23 memberikan persetujuan untuk promosi pada 20 September atas teks versi v3, dan catatan merekamnya. Jika kelak Laundry Wangi mengubah arti promosi — lebih banyak pesan, atau pesan dari mitra — versi teks memperlihatkan pelanggan mana yang menyetujui rumusan mana, dan mereka yang menyetujui versi lama dapat ditanya lagi alih-alih dianggap setuju.

12.7 Mengujinya

Kode 12.8 memeriksa bahwa alur pertama memiliki kedelapan temuan dengan rasio yang dihitung dengan tangan ($320 \times 56/(90 \times 14) = 14.2$ dan $6/2 = 3.0$), bahwa alur kedua bersih, dan bahwa dalam buku besar penarikan yang lebih akhir mengalahkan persetujuan yang lebih awal serta tujuan yang tidak pernah ditanyakan tidak diizinkan. **Kode 12.9** melakukan keenam pemeriksaan yang sama dalam Java.

Kode 12.8 `test_ch12.py`: pengujian *linter* dan buku besar.

```
"""Pemeriksaan linter dan buku besar persetujuan (python -m pytest -q)."""
from pathlib import Path
from consent_ledger import may_use, replay
from consent_lint import lint, load

DATA = Path(__file__).resolve().parent.parent / "data"


def test_first_flow_has_all_eight_findings():
    found, prominence, asym = lint(load(DATA / "flow_v1.txt"))
    assert len(found) == 8 and asym == 3.0
    assert round(prominence, 1) == 14.2          # 320*56 / (90*14)


def test_second_flow_is_clean():
    assert lint(load(DATA / "flow_v2.txt"))[0] == []


def test_withdrawal_wins_over_earlier_consent():
    state = replay(DATA / "consent_log.csv")
    assert not may_use(state, "U17", "promo")
    assert may_use(state, "U17", "orders")
    assert not may_use(state, "U99", "orders")   # tak pernah ditanya
```

Kode 12.9 `Ch12Test.java`: pemeriksaan yang sama dalam Java biasa.

```
/** Pemeriksaan linter dan buku besar persetujuan (java -cp out Ch12Test). */
public class Ch12Test {
    static int failed = 0;

    static void check(boolean ok) { if (!ok) failed++; }

    public static void main(String[] args) throws Exception {
        var r1 = ConsentLint.lint(ConsentLint.load("../data/flow_v1.txt"));
        check(r1.found().size() == 8 && r1.asym() == 3.0);
        check(Math.round(r1.prominence() * 10) == 142);   // 320*56 / (90*14)
        check(ConsentLint.lint(ConsentLint.load("../data/flow_v2.txt"))
              .found().isEmpty());
        var state = ConsentLedger.replay("../data/consent_log.csv");
        check(!ConsentLedger.mayUse(state, "U17", "promo"));
        check(ConsentLedger.mayUse(state, "U17", "orders"));
        check(!ConsentLedger.mayUse(state, "U99", "orders")); // tak pernah ditanya
        System.out.println(failed == 0 ? "all 6 checks passed"
                                       : failed + " failed");
    }
}
```

`python -m pytest -q` mencetak

```
3 passed in 0.01s
```

dan `java -cp out Ch12Test` mencetak

```
all 6 checks passed
```

12.8 Membuatnya Lebih Baik

Rasa senang yang benar. Setelah trik-trik disingkirkan, laundry memerlukan alasan yang jujur agar orang bergabung dan bertahan, dan desain emosional menyediakannya. Baris status untuk setiap pesanan (“diterima”, “dicuci”, “dikeringkan — hujan bisa menambah setengah hari”, “disetrika”, “siap pukul 16.00”) melayani tingkat behavioral dan, karena tetap benar bahkan ketika kabarnya buruk, tingkat reflektif pula. Ucapan terima kasih singkat beserta penghematan

anggota bulan ini ("Anda hemat Rp18.000 pada September") mengubah potongan harga yang abstrak menjadi manfaat yang terlihat. Keduanya tidak memerlukan hitung mundur.

Pengingat sebelum menagih. Perpanjangan otomatis itu adil bila terlihat: tampilkan tanggal perpanjangan berikutnya di halaman Akun, kirim pengingat sehari sebelum menagih, dan biarkan anggota menghentikan perpanjangan dalam satu langkah langsung dari pengingat itu.

Melampaui yang terlihat oleh *linter*. *Linter* memeriksa sebuah *uraian*; sebuah alur dapat lulus darinya tetapi tetap menyesatkan, misalnya bila teks sebuah tujuan secara teknis benar tetapi kabur ("untuk meningkatkan layanan kami"). Bacakan setiap tujuan kepada orang di luar bisnis dan tanyakan apa yang menurutnya diizinkan; jika jawabannya berbeda dari kenyataan, tulis ulang. *Linter* juga tidak dapat melihat kebenaran: apakah "siap pukul 16.00" biasanya tepat adalah pertanyaan bagi data laundry itu, dan Bab 14 memperlihatkan cara mengukurnya.

Bahasa yang menenangkan. Kata-kata kecil di antarmuka — *microcopy* — memikul banyak beban emosional. Bandingkan "Pembayaran gagal!" berhuruf merah dengan "Pembayaran belum masuk. Saldo Anda tidak terpotong; silakan coba lagi atau bayar di kasir". Kalimat kedua lebih panjang, tetapi menjawab tiga pertanyaan yang langsung muncul di benak pelanggan: apa yang terjadi, apakah uang saya aman, dan apa yang bisa saya lakukan sekarang. Bagi pelanggan yang pernah menjadi korban penipuan daring, kalimat semacam itu membedakan layanan yang dapat dipercaya dari layanan yang patut dicurigai. Aturan yang sama berlaku untuk pesan keterlambatan, penolakan, dan kesalahan: katakan yang sebenarnya, katakan apa akibatnya bagi pengguna, dan tawarkan langkah berikutnya.

Kebiasaan setempat. Banyak usaha kecil di Indonesia hidup dari percakapan WhatsApp, dan pelanggan membawa harapan dari percakapan itu ke aplikasi: jawaban yang cepat, nada yang ramah, dan manusia yang dapat dihubungi bila ada masalah. Aplikasi yang baik tidak memutus hubungan itu, tetapi melengkapinya. Tombol "Tanya Bu Sri" yang membuka WhatsApp dengan nomor pesanan yang sudah terisi mempertahankan kehangatan lama sambil memberi ketertiban baru. Yang tidak boleh adalah menjadikan WhatsApp *satu-satunya* jalan untuk hal yang merugikan bisnis, seperti berhenti berlangganan atau menarik persetujuan, sementara hal yang menguntungkan bisnis cukup satu ketukan di aplikasi.

12.9 Memakainya Sungguhan

- **Jalankan tinjauan alur yang adil (Tabel 12.2) pada setiap alur yang meminta uang, data, atau perhatian**, dan uraikan alurnya dalam format **Kode 12.1** agar *linter* dapat berjalan pada setiap rilis.
- **Simpan buku besar persetujuan, bukan penanda persetujuan.** Simpan peristiwa beserta versi teksnya; turunkan keadaan kininya; tampilkan kepada pengguna di halaman Akun dengan kendali penarikan untuk setiap tujuan.
- **Tulis tujuan dalam bahasa pengguna.** Bagi khalayak Indonesia, artinya bahasa Indonesia yang jelas lebih dahulu, dengan akibat yang konkret: "sekitar dua pesan sebulan", "hanya tiga toko yang tercantum".
- **Buat jalan keluar sejelas jalan masuk.** Tombol berhenti berlangganan tempatnya di halaman yang sama dengan status langganan.
- **Hitung ongkos trik.** Lacak penarikan persetujuan, pembatalan, keluhan, dan pengembalian dana di samping pendaftaran. Pola yang menaikkan yang satu sambil menaikkan yang lain sedang meminjam kepercayaan dengan bunga tinggi.

- **Bila ragu, tanyakan apakah Anda sanggup menjelaskan desain itu langsung di depan pengguna.** Bu Sri tidak akan pernah berkata "*Tinggal 3 kuota*" setiap hari di meja kasirnya; aplikasinya pun tidak semestinya berkata begitu.

12.10 IMK dalam Praktik: Profitina

Profitina, platform visibilitas AI asal Indonesia, memperlakukan persetujuan dan kejujuran sebagai fitur produk. Kepatuhan swalayannya terhadap UU PDP 27/2022 mencakup persetujuan eksplisit, *hashing* kata sandi dengan Argon2id, pembatasan laju masuk log, ekspor data, penggantian surel dan kata sandi, penghapusan akun sepenuhnya yang dimulai pengguna sendiri, serta halaman hukum yang diterbitkan sebagai HTML. Dalam istilah bab ini, jalan keluar (ekspor, penghapusan) tersedia sama baiknya dengan jalan masuk, dan bukti persetujuan disimpan, bukan diandaikan.

Beberapa keputusan produk mengikuti tinjauan alur yang adil. **Langganan Pro** (Rp50.000 untuk 30 hari, dibayar lewat QRIS atau dengan menukarkan PROFIT) dapat diperpanjang otomatis dari saldo pengguna, dan pengguna diingatkan dalam 24 jam sebelum perpanjangan — perpanjangan yang mengumumkan dirinya. Membuat **kanal pengguna** mensyaratkan saldo 50 PROFIT, yang diperiksa tetapi *tidak dipotong*, dan aturannya menyatakan demikian; batas-batasnya (paling banyak dua kanal baru sehari, lima yang aktif) dinyatakan, bukan ditemukan belakangan. **Mesin konten**, yang menyusun draf dengan model Qwen2.5 yang dijalankan secara lokal, hanya menyajikan draf AI bila benar-benar lebih baik daripada templat bawaan, dan bila tidak ada model yang tersedia aplikasi *mengatakannya* alih-alih diam-diam menyajikan templat: kejujuran tentang kecakapan membangun kepercayaan reflektif yang akan terkikis oleh cadangan yang disembunyikan. Dan prinsip yang tertulis dalam aturan produk itu — *aplikasi menyarankan; pengguna memutuskan* — adalah bentuk positif dari semua yang diperingatkan Subbab 12.3.3. Studi kasus yang dapat dijalankan di buku ini tetaplah laundry di Bab 16.

12.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Meniru trik "pertumbuhan" dari aplikasi besar.** Apa yang dapat ditanggung aplikasi besar tidak dapat ditanggung usaha kecil yang dibangun di atas kepercayaan pribadi. Tinjau setiap pola tiruan dengan **Tabel 12.1**.
- **Desakan waktu dan kelangkaan palsu.** Hitung mundur atau jumlah stok yang tidak benar adalah klaim palsu, apa pun maksud desainnya.
- **Kotak opsional yang sudah tercentang dan persetujuan yang dibundel.** Diam bukanlah setuju; satu centang untuk banyak tujuan bukanlah pilihan.
- **Tombol tak setara untuk pilihan setara.** Ukur rasio penonjolan Persamaan (12.2); beri penolakan gaya yang sama.
- **Kata-kata yang membuat merasa bersalah.** Ganti *confirmshaming* dengan label netral yang menguraikan hasilnya.
- **Jalan keluar lebih sulit daripada jalan masuk.** Jaga asimetri hambatan Persamaan (12.1) pada 1 atau kurang, di saluran yang sama.
- **Satu penanda persetujuan yang ditimpa pada setiap perubahan.** Ini menghancurkan bukti yang diminta hukum. Simpan peristiwanya.
- **Tujuan yang kabur.** "Untuk meningkatkan layanan kami" tidak menjelaskan apa pun. Sebutkan pemakaiannya dan akibatnya.

- **Mengira lonjakan pendaftaran sebagai keberhasilan.** Amati juga penarikan persetujuan, pembatalan, dan keluhan.

12.12 Ringkasan Bab

- Tiga tingkat Norman — viseral, behavioral, reflektif — menggambarkan bagaimana sebuah antarmuka dirasakan pada pandangan pertama, saat dipakai, dan dalam ingatan (**Gambar 12.1**); efek estetika–kegunaan membuat antarmuka yang menarik tampak lebih mudah, yang dapat menolong sekaligus menyesatkan.
- Kepercayaan bertumpu pada kecakapan, itikad baik, dan kejujuran; ia dibangun oleh status yang benar, biaya yang jelas, dan jalan keluar yang mudah, dan hilang oleh trik yang ketahuan. *Passkey* memperlihatkan bahwa keamanan juga dapat mengurangi ketegangan.
- *Dark pattern* menipu atau memanipulasi pengguna demi keuntungan layanan; yang umum antara lain desakan waktu dan kelangkaan palsu, persetujuan yang sudah tercentang dan dibundel, campur tangan antarmuka, *confirmshaming*, dan pembatalan yang sulit (**Tabel 12.1**). Riset menemukannya pada sekitar 11% dari 11.000 situs belanja.
- Asimetri hambatan $A = s_{out}/s_{in}$ (Persamaan (12.1)) dan rasio penonjolan P (Persamaan (12.2)) membuat dua di antaranya terukur; alur yang adil memiliki $A \leq 1$ dan tombol yang setara.
- DSA Uni Eropa (Pasal 25) melarang antarmuka yang menipu dan memanipulasi di platform daring; Digital Fairness Act diperkirakan hadir pada akhir 2026; aturan *click-to-cancel* FTC dibatalkan pada Juli 2025 dan penyusunan aturan baru dimulai pada 2026; UU PDP mewajibkan persetujuan yang eksplisit, jelas, dapat ditarik, dan berbukti, dan PP 33/2026 (berlaku mulai 16 Januari 2027) melaksanakannya.
- Tinjauan alur yang adil (**Tabel 12.2**), *linter*, dan buku besar persetujuan mengubah prinsip-prinsip ini menjadi pemeriksaan: delapan temuan pada alur pertama Laundry Wangi, tidak satu pun pada alur kedua (**Gambar 12.2**), dan buku besar yang menjawab "bolehkah kita memakai ini?" berdasarkan bukti.

12.13 Soal Latihan

1. Sebuah aplikasi pesan-antar makanan memungkinkan pengguna berlangganan dengan satu ketukan dari layar beranda, tetapi berhenti memerlukan membuka Pengaturan, lalu Akun, lalu Keanggotaan, lalu "Kelola", lalu menjawab survei, lalu mengonfirmasi, lalu mengetik alasan. Hitung A dengan Persamaan (12.1) dan usulkan alur dengan $A \leq 1$.
2. Sebuah spanduk *cookie* memiliki tombol "Terima semua" berukuran 300 × 48 piksel dan tautan "Pengaturan" berukuran 70 × 16 piksel, tanpa "Tolak semua". Hitung P dengan Persamaan (12.2), sebutkan polanya, dan buat sketsa spanduk yang adil.
3. Golongkan setiap hal berikut menurut tingkat Norman dan katakan apakah ia membangun atau mempertaruhkan kepercayaan: (a) animasi menyenangkan ketika pesanan dibuat; (b) status pesanan yang berbunyi "siap" padahal pakaian masih dikeringkan; (c) pesan bulanan yang menunjukkan penghematan anggota yang sebenarnya.
4. Tambahkan frasa "*tidak, terima kasih, saya mau rugi*" dan "no, i prefer to miss out" ke daftar frasa mempermalukan pada *linter*, lalu tulis berkas alur yang satu-satunya masalahnya adalah tombol tolak berlabel "No, I prefer to miss out". Apa yang dicetak *linter*?
5. Dalam catatan persetujuan, pelanggan U23 menyetujui promosi atas teks versi v3. Andaikan Laundry Wangi memperkenalkan versi v4, yang menambahkan penawaran mitra ke dalam promosi. Jelaskan, dengan merujuk UU PDP, mengapa buku besar tidak boleh memperlakukan

persetujuan U23 atas v3 sebagai persetujuan atas v4, dan ubah `may_use` agar menerima versi teks yang disyaratkan sebagai parameter.

6. Pasal 25 DSA berbicara tentang desain yang "secara material mendistorsi atau merusak" kemampuan pengguna untuk mengambil keputusan yang bebas dan berdasarkan informasi. Ambil pesan "Tinggal 3 kuota hari ini!" dan kemukakan argumen kedua sisi: kapan pesan kelangkaan dapat menjadi informasi yang sah, dan apa yang membuat versi Laundry Wangi menjadi *dark pattern*?

12.14 Penyelesaian Soal Latihan

1. Bergabung memerlukan $s_{in} = 1$ langkah; keluar memerlukan $s_{out} = 7$ (Pengaturan, Akun, Keanggotaan, Kelola, survei, konfirmasi, alasan), sehingga menurut Persamaan (12.1) $A = 7/1 = 7$. Desain yang adil mengubah kedua sisi. Berlangganan menjadi dua langkah — ketuk "Berlangganan", lalu konfirmasi layar yang menyatakan harga, tanggal perpanjangan, dan cara berhenti — yang melindungi pengguna dari tagihan satu ketukan yang tak disengaja. Berhenti juga menjadi dua langkah: tombol "Berhenti berlangganan" pada kartu keanggotaan yang sama, lalu satu konfirmasi yang menyatakan tanggal berakhirnya ("Keanggotaan Anda berakhir 31 Oktober; Anda tidak akan ditagih lagi"), dengan pertanyaan opsional tentang alasan yang boleh dilewati. Kini $s_{in} = s_{out} = 2$ dan $A = 1$.

2. $P = (300 \times 48)/(70 \times 16) = 14400/1120 = 12.9$. Polanya adalah campur tangan antarmuka (tombol besar melawan tautan kecil), serta penghalangan atau tindakan paksa karena tidak adanya "Tolak semua": menolak memerlukan lebih banyak langkah daripada menerima ($A > 1$). Spanduk yang adil menampilkan tiga tombol berukuran dan bergaya sama — "Terima semua", "Tolak semua", "Pilih" — dengan setiap kategori nonesensial mati secara bawaan pada tampilan "Pilih".

3. (a) Viseral dan behavioral: konfirmasi yang singkat dan menyenangkan meyakinkan pengguna bahwa ketukannya berhasil; ia membangun kepercayaan selama cepat dan tidak menghalangi. (b) Behavioral dan terutama reflektif: status palsu ketahuan begitu pelanggan tiba, dan ia merusak kejujuran laundry di mata pelanggan; ia sangat mempertaruhkan kepercayaan. Lebih baik: "dikeringkan — hujan membuat kami terlambat, perkiraan besok pukul 10.00". (c) Reflektif: manfaat pribadi yang benar mengingatkan anggota mengapa ia bergabung; ia membangun kepercayaan, asalkan angkanya akurat.

4. Tambahkan kedua frasa dalam huruf kecil ke `SHAMING` pada *linter* di kedua bahasa pemrograman (daftar itu dibandingkan dengan label yang sudah dijadikan huruf kecil). Berkas alur dengan tujuan wajib, tanpa tujuan opsional yang menyala, `bundled|no`, dua tombol terisi yang setara, tanpa hitung mundur, `steps|subscribe|2`, `steps|cancel|2`, baris penarikan tanpa kata "only", dan `button|reject|No, I prefer to miss out|150|48|filled` menghasilkan satu temuan:

```
    - confirmshaming         reject label: 'No, I prefer to miss out'
```

dengan prominence 1.0 dan cancel/subscribe 1.0 pada baris kepalanya.

5. Persetujuan menurut UU PDP diberikan untuk tujuan tertentu yang telah disampaikan kepada subjek data (Pasal 20 dan 22, ditegaskan kembali oleh PP 33/2026), dan pengendali harus mampu menunjukkan buktinya (Pasal 24). U23 menyetujui tujuan sebagaimana diuraikan dalam v3; v4 menguraikan tujuan yang berbeda (penawaran mitra), yang tidak pernah disetujui U23. Memperlakukan persetujuan lama sebagai mencakup teks baru berarti memproses data untuk

tujuan yang tidak pernah diberitahukan kepada orang itu. Perbaikannya adalah mensyaratkan versi:

```
def may_use(state, user, purpose, version=None):
    action, _, ver = state.get((user, purpose), ("none", "", ""))
    return action == "give" and (version is None or ver == version)
```

Maka `may_use(state, "U23", "promo", "v4")` bernilai `False`, dan aplikasi menampilkan teks baru kepada U23 dan bertanya lagi.

6. Pesan kelangkaan adalah informasi yang sah bila benar, spesifik, dan relevan: laundry yang berkapasitas tepat dua puluh anggota dengan harga khusus boleh berkata “20 kuota anggota bulan ini, tersisa 3”, bila hitungannya nyata dan berubah seiring kuota terisi. Pesan itu membantu pelanggan memutuskan. Pesan Laundry Wangi ditampilkan setiap hari tanpa peduli ada atau tidaknya batas yang nyata; pesan itu palsu, dan satu-satunya fungsinya adalah menciptakan tekanan yang memperpendek pertimbangan. Itulah “distorsi material” dalam rumusan DSA: desain mengubah keputusan bukan dengan memberi informasi, melainkan dengan memberi informasi yang salah. Ujinya sederhana — apakah pesan itu tetap muncul bila tidak benar? Bagi Laundry Wangi jawabannya ya, sehingga ia adalah *dark pattern*.

Daftar Pustaka

[1] H. Brignull, “Deceptive Patterns,” diakses Sep. 2026. [Daring]. Tersedia: https://www.deceptive.design/

[2] D. A. Norman, *Emotional Design: Why We Love (or Hate) Everyday Things*. New York, NY, USA: Basic Books, 2004.

[3] G. Lindgaard, G. Fernandes, C. Dudek, and J. Brown, “Attention web designers: You have 50 milliseconds to make a good first impression!,” *Behaviour & Information Technology*, vol. 25, no. 2, pp. 115–126, 2006.

[4] M. Kurosu and K. Kashimura, “Apparent usability vs. inherent usability: Experimental analysis on the determinants of the apparent usability,” in *Conf. Companion CHI ’95*, Denver, CO, USA, 1995, pp. 292–293.

[5] FIDO Alliance, “The State of Passkeys: Global Consumer and Workforce Report,” Mei 2026. [Daring]. Tersedia: https://fidoalliance.org/wp-content/uploads/2026/05/The-State-of-Passkeys-Global-Consumer-and-Workforce-Report-1.pdf

[6] A. Mathur, G. Acar, M. J. Friedman, E. Lucherini, J. Mayer, M. Chetty, and A. Narayanan, “Dark patterns at scale: Findings from a crawl of 11K shopping websites,” *Proc. ACM Hum.-Comput. Interact.*, vol. 3, no. CSCW, art. 81, 2019.

[7] C. M. Gray, C. T. Santos, N. Bielova, and T. Mildner, “An ontology of dark patterns knowledge: Foundations, definitions, and a pathway for shared knowledge-building,” in *Proc. CHI ’24*, Honolulu, HI, USA, 2024, doi: 10.1145/3613904.3642436.

[8] European Parliament and Council, “Regulation (EU) 2022/2065 on a Single Market for Digital Services (Digital Services Act),” Art. 25, *Official Journal of the European Union*, L 277, 27 Okt. 2022. [Daring]. Tersedia: https://eur-lex.europa.eu/eli/reg/2022/2065/oj

[9] S. Ahuja, J. Gunawan, N. Bielova, and C. T. Santos, “Dark patterns and the EU Digital Services Act: Mapping autonomy violations and design factors,” in *Proc. CHI ’26*, 2026, doi: 10.1145/3772318.3791479.

[10] European Parliament, “Digital Fairness Act,” Legislative Train Schedule, diakses Sep. 2026. [Daring]. Tersedia: https://www.europarl.europa.eu/legislative-train/theme-protecting-our-democracy-upholding-our-values/file-digital-fairness-act

[11] Crowell & Moring LLP, “Clicking All the Right Boxes: FTC Moves to Revive ‘Click-to-Cancel’ Rule Following Eighth Circuit Vacatur,” 11 Feb. 2026. [Daring]. Tersedia:

https://www.crowell.com/en/insights/client-alerts/clicking-all-the-right-boxes-ftc-moves-to-revive-click-to-cancel-rule-following-eighth-circuit-vacatur

[12] Republik Indonesia, *Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*, 2022. [Daring]. Tersedia: https://peraturan.bpk.go.id/Details/229798/uu-no-27-tahun-2022

[13] Republik Indonesia, *Peraturan Pemerintah Nomor 33 Tahun 2026 tentang Peraturan Pelaksanaan Undang-Undang Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*, ditetapkan 16 Jul. 2026, berlaku 16 Jan. 2027; diringkas dalam Veritask, "Pengaturan Teknis Pelindungan Data Pribadi … Akhirnya Terbit Lewat PP 33/2026," 2026. [Daring]. Tersedia: https://veritask.ai/id/artikel/pengaturan-teknis-pelindungan-data-pribadi-dan-kewajiban-pengendali-serta-prosesor-akhirnya-terbit-lewat-pp-33-2026

Bab 13

Merancang Antarmuka AI: Chat, Kopilot, dan Agen

Seorang pemilik laundry menginginkan asisten chat yang menerima pesanan tengah malam, saat ia tidur. Asisten itu akan membaca pesan yang ditulis tergesa-gesa, menebak maksud pelanggan, menyebutkan harga, dan memesan penjemputan, dan setiap langkah itu bisa keliru dengan cara yang merugikan uang atau kepercayaan. Bab ini membahas perancangan percakapan antara manusia dan sistem yang menebak. Di akhir bab Anda akan mampu membedakan antarmuka chat, kopilot, dan agen, serta menyebutkan kewajiban masing-masing kepada penggunanya; menerapkan pedoman interaksi manusia-AI yang telah diterbitkan; menempatkan setiap tindakan sistem AI pada tangga otonomi menurut risikonya; menampilkan ketidakpastian dengan kata-kata yang dipahami orang dan bertanya alih-alih menebak; memberi orang jalan untuk mengonfirmasi, membatalkan, dan menjangkau manusia; mengukur ketergantungan berlebih, ketergantungan kurang, dan kalibrasi dari log keputusan; membangun asisten pesanan kecil dalam Java dan Python yang mengikuti aturan-aturan itu; serta memenuhi kewajiban transparansi yang berlaku pada 2026 menurut EU AI Act dan hukum Indonesia.

13.1 Pesan Tengah Malam Bu Wati

Bu Wati mengelola Profitina Laundry, toko kecil yang menjadi contoh berjalan buku ini: tiga mesin cuci, dua pengering, satu meja setrika, dan nomor WhatsApp yang tercetak di setiap kantong plastik. Sebagian besar pelanggannya mahasiswa dari indekos di sekitar toko dan pekerja muda yang berangkat ke kantor sebelum pukul tujuh. Mereka mengirim pesanan ketika teringat, dan mereka teringat larut malam: setelah tugas selesai, setelah sif malam, setelah episode terakhir sebuah serial. Setiap pagi Bu Wati membuka ponselnya dan mendapati dua puluh sampai tiga puluh pesan yang masuk antara pukul sepuluh malam dan dua dini hari.

Pesan-pesan itu pendek dan ditulis terburu-buru. “Bisa cuci 4 kg dan setrika, jemput besok jam 9?” termasuk pesan yang baik. “Ada sekantong cucian, tolong jemput jam 16” tidak menyebut berat maupun layanan. “Cuci kilat 6 kilo, hari ini jam 13.30” jelas soal kecepatan, tetapi tidak soal setrika. Bu Wati menjawab satu per satu, menanyakan yang kurang, menyebutkan harga, lalu memesankan penjemputan kepada sepupunya yang mengendarai sepeda motor. Ketika ia selesai menjawab, sebagian pelanggan sudah pindah ke toko lain yang menjawab tengah malam.

Keponakannya, Dimas, mahasiswa informatika, mengusulkan asisten AI: program yang langsung menjawab pesan WhatsApp siang dan malam, memahami keinginan pelanggan, menyebutkan harga, dan memesan penjemputan. Bu Wati menyukai gagasan itu sekaligus mencemaskannya. Ia pernah melihat tangkapan layar chatbot yang menjanjikan pengembalian uang yang tak pernah disetujui siapa pun, yang mengarang jam buka, dan yang tak bisa ditinggalkan ketika pelanggan ingin bicara dengan manusia. Ia mengajukan tiga pertanyaan yang menjadi pokok bab ini: *Bagaimana pelanggan tahu bahwa mereka berbicara dengan mesin? Apa yang terjadi bila ia salah paham? Dan siapa yang bertanggung jawab bila ia memesan hal yang keliru?*

Pertanyaan-pertanyaan itu bukan tentang model bahasa di dalam asisten. Semuanya tentang antarmuka di sekelilingnya: apa yang ditampilkan asisten, apa yang ditanyakannya, apa yang dikerjakannya sendiri, dan apa yang diserahkannya kepada manusia. Dua asisten yang dibangun di atas model yang sama bisa menyenangkan dan tepercaya, atau menjengkelkan dan berbahaya, sepenuhnya bergantung pada keputusan desain itu. Pada 2025 dan 2026 persoalannya bergeser dari jendela chat ke *agen* yang bertindak: ChatGPT agent dari OpenAI, yang diluncurkan 17 Juli 2025, menjelajah situs web, mengisi formulir, dan dapat melakukan pembelian, serta "dilatih untuk secara eksplisit meminta izin Anda sebelum mengambil tindakan yang berakibat nyata" [1]. Masalah desain yang dihadapi Bu Wati dengan satu nomor WhatsApp sama dengan masalah yang dihadapi perusahaan terbesar dengan jutaan pengguna.

13.2 Gagasannya dengan Kata-Kata Sederhana

13.2.1 Chat, kopilot, dan agen

Tiga jenis antarmuka AI lazim dijumpai pada 2026, dan ada gunanya menamainya sebelum merancang salah satunya.

> **DEFINISI.** *Antarmuka chat* menjawab dengan kata-kata: orang bertanya, sistem membalas, dan orang memutuskan apa yang dilakukan dengan balasan itu. *Kopilot* (*copilot*) bekerja di dalam alat lain dan mengusulkan perubahan yang dapat diterima atau ditolak, misalnya usulan balasan di program surel atau usulan baris di penyunting kode. *Agen* (*agent*) mengambil tindakan di dunia atas nama orang, misalnya memesan, membayar, mengirim, atau menghapus, sering kali dalam beberapa langkah tanpa instruksi baru untuk tiap langkah.

Garis pemisah ketiganya adalah garis *akibat*. Balasan chat yang salah membuang satu menit. Usulan kopilot yang salah lalu diterima tanpa dibaca menjadi kesalahan orang itu sendiri. Tindakan agen yang salah sudah telanjur terjadi: sepeda motor sudah melaju ke indekos yang salah, atau uang sudah keluar dari rekening. Makin banyak yang dapat dilakukan sistem AI, makin besar kewajiban antarmukanya untuk membantu orang melihat apa yang hendak dilakukannya dan menghentikannya.

Asisten Bu Wati adalah agen kecil. Ia mengobrol (menanyakan yang kurang), ia mengusulkan (menampilkan pesanan yang dipahaminya), dan ia bertindak (memesan penjemputan). Setiap bagian memerlukan kehati-hatian yang berbeda.

13.2.2 Empat pertanyaan yang harus dijawab setiap antarmuka AI

Antarmuka klasik bersifat *deterministik*: tombol yang sama selalu melakukan hal yang sama, dan Bab 3 membangun banyak prinsip desain yang baik di atas konsistensi itu. Sistem AI menebak, dan tebakan dapat keliru dengan cara baru setiap kali. Karena itu penggunanya memerlukan jawaban atas empat pertanyaan yang tak pernah muncul pada sakelar lampu, yang dihimpun dalam **Tabel 13.1**.

Tabel 13.1 Empat pertanyaan pengguna tentang sistem AI mana pun, dan cara asisten pesanan menjawab masing-masing.

Pertanyaan pengguna	Yang harus ditampilkan antarmuka	Pada asisten Bu Wati
Apa yang bisa dilakukannya, dan seberapa baik?	Cakupan dan batasnya, sebelum pemakaian pertama	Baris "asisten otomatis"; menerima pesanan, tidak menangani keluhan
Seberapa yakin ia pada jawaban ini?	Ketidakpastian dalam bentuk yang dipahami orang	"sure", "likely – please check", "assumed – please check" di samping tiap bagian pesanan
Apa yang akan dilakukannya berikutnya?	Tindakan yang diusulkan, sebelum terjadi	"Book this pickup? (yes / no)" dengan pesanan lengkap dan harga
Bagaimana saya mengoreksi, menghentikan, atau membatalkannya?	Jalan kembali, dan jalan ke manusia	"no" untuk mengubah, UNDO untuk satu pesan, STAFF untuk menjangkau Mbak Dewi

Keempat pertanyaan ini merangkum banyak sekali hasil penelitian, yang disajikan bersama buktinya di Subbab 13.3. Keempatnya juga menjadi uji cepat bagi produk AI mana pun yang Anda temui: bila salah satunya tak dapat dijawab hanya dari layar, desainnya berlubang.

13.2.3 Contoh yang dikerjakan dengan tangan

Ambil pesan "I have a bag of laundry, pick up at 16 please" (ada sekantong cucian, tolong jemput jam 16). Bacalah seperti asisten harus membacanya.

1. *Layanan.* Tidak ada kata untuk mencuci, menyetrika, atau melipat. Layanan yang paling sering di toko adalah cuci lipat, jadi asisten boleh *mengasumsikannya*, tetapi harus mengatakan bahwa ia berasumsi.
2. *Berat.* Tidak ada angka dengan "kg" atau "kilo". "Sekantong" bisa dua kilogram, bisa delapan. Harga bergantung padanya, jadi menebak tidak dapat diterima; asisten harus bertanya. Pertanyaan yang baik juga memberi jalan keluar bagi orang yang tidak tahu: "kami menimbangnya saat penjemputan dan mengirim harga pastinya sebelum mencuci".
3. *Kecepatan.* Tidak ada kata seperti "express" atau "today". Asumsikan reguler, dan katakan demikian.
4. *Penjemputan.* "at 16" memberi jam tetapi bukan hari. Pada malam hari, besok adalah makna yang paling mungkin, tetapi hanya mungkin.

Pelanggan menjawab "about 2.5 kg". Kini asisten dapat mengusulkan: cuci lipat (diasumsikan), 2,5 kg (yakin), reguler (diasumsikan), besok 16.00 (mungkin). Toko menetapkan tarif minimum 3 kg, sehingga harganya 3 kg kali Rp8.000, yaitu Rp24.000, dan asisten harus menjelaskan *mengapa* ia menagih 3 kg padahal pelanggan menyebut 2,5. Barulah setelah pelanggan menjawab "yes" ia memesan.

Aturan harga, ditulis sekali agar setiap bagian program memakai rumus yang sama, adalah

$$P = r_s \cdot \max(w,\, w_{\min}) \cdot e \tag{13.1}$$

dengan r_s tarif per kilogram untuk layanan s, w berat dalam kilogram, $w_{\min} = 3$ tarif minimum, dan e bernilai 1,5 untuk layanan kilat dan 1 untuk lainnya. Untuk pesanan ini, Persamaan (13.1) memberi $P = 8000 \cdot \max(2.5,3) \cdot 1 = 24000$ rupiah.

13.3 Teori dan Buktinya

13.3.1 Pedoman interaksi manusia-AI

Pada 2019 sebuah tim di Microsoft menghimpun lebih dari 150 rekomendasi desain untuk produk AI dari industri dan riset, meringkasnya menjadi 18 pedoman, dan mengujinya bersama 49 desainer pada 20 produk AI nyata; hasilnya, yang diterbitkan di CHI 2019, masih menjadi daftar periksa yang paling banyak dikutip di bidang ini dan kini dipelihara sebagai HAX Toolkit [2]. Pedoman itu dikelompokkan menurut *kapan* berlaku: di awal, selama interaksi, ketika sistem keliru, dan seiring waktu. **Tabel 13.2** memuat yang paling penting bagi asisten kecil.

Tabel 13.2 Sebagian dari 18 pedoman interaksi manusia-AI dan cara masing-masing membentuk asisten pesanan.

Fase	Pedoman (diparafrasekan dari [2])	Maknanya bagi asisten pesanan
Di awal	G1 Jelaskan apa yang dapat dilakukan sistem	Katakan "asisten otomatis" dan apa yang ditanganinya
Di awal	G2 Jelaskan seberapa baik ia melakukannya	Tampilkan keyakinan per bagian pesanan
Selama interaksi	G4 Tampilkan informasi yang relevan dengan konteks	Tampilkan harga dan alasan tarif minimum bersama pesanan
Selama interaksi	G5 Ikuti norma sosial yang relevan	Sopan, singkat, sesuai ragam bahasa pelanggan
Ketika keliru	G8 Dukung penolakan yang efisien	"no" langsung membuang usulan
Ketika keliru	G9 Dukung koreksi yang efisien	Ubah satu bagian tanpa mengetik ulang seluruh pesanan
Ketika keliru	G10 Batasi layanan saat ragu	Tanyakan berat alih-alih menebaknya
Ketika keliru	G11 Jelaskan mengapa sistem berbuat begitu	Katakan "assumed" bila nilai bawaan dipakai
Seiring waktu	G13 Belajar dari perilaku pengguna	Ingat apa yang sudah dikatakan pelanggan dalam chat ini
Seiring waktu	G17 Sediakan kendali global	STAFF untuk menjangkau manusia kapan saja

People + AI Guidebook dari kelompok People + AI Research (PAIR) Google membahas wilayah yang sama dari sudut pandang tim produk, dengan bab tentang kebutuhan pengguna, model mental, keterjelasan dan kepercayaan, umpan balik dan kendali, serta "galat dan kegagalan yang anggun" [3]. Nasihat utamanya tentang kepercayaan: penjelasan harus membantu orang menakar seberapa jauh mereka bersandar pada sistem, bukan membujuk mereka agar makin bersandar. Kedua dokumen sepakat pada hal yang paling penting bagi Bu Wati: fitur AI harus dirancang di sekitar apa yang terjadi ketika ia keliru, sebab ia pasti akan keliru.

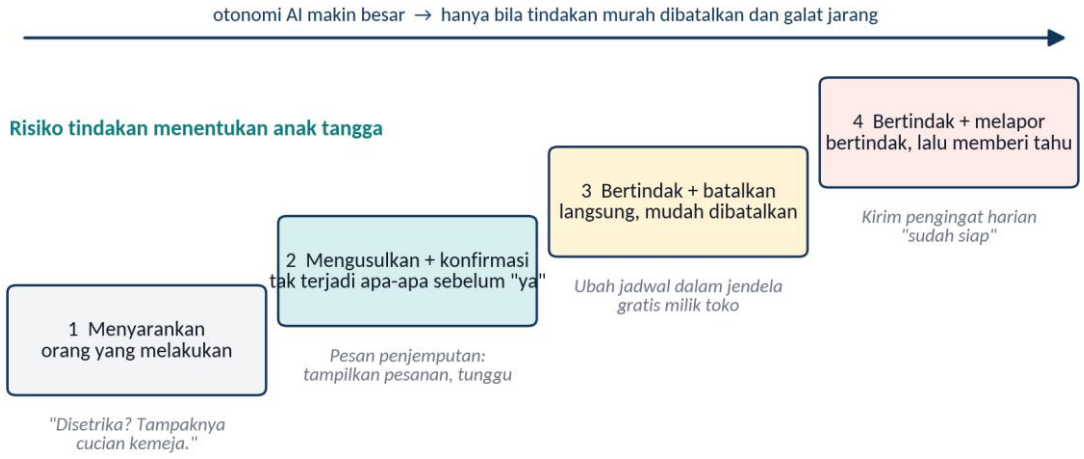
13.3.2 Tingkat otomatisasi dan tangga otonomi

Jauh sebelum model bahasa, para insinyur pesawat dan pabrik bertanya seberapa banyak mesin boleh memutuskan sendiri. Parasuraman, Sheridan, dan Wickens mengusulkan pada 2000 agar otomatisasi dinilai terpisah untuk empat fungsi (memperoleh informasi, menganalisisnya, memutuskan, dan bertindak) dan bahwa tiap fungsi dapat diotomatisasi pada tingkat yang berbeda, dari "komputer tidak membantu sama sekali" sampai "komputer bertindak dan mengabaikan manusia" [4]. Pesan kuncinya: tingkat yang tepat dipilih dari akibatnya; otomatisasi *tindakan* yang tinggi hanya dapat diterima bila galat murah atau dapat dibalik.

Prinsip antarmuka inisiatif campuran (*mixed-initiative*) dari Horvitz, yang ditulis pada 1999 untuk asisten generasi awal, menambahkan argumen ekonomis: asisten sebaiknya bertindak

sendiri hanya bila manfaat harapan dari bertindak melebihi biaya harapan dari keliru, dan ia sebaiknya bertanya ketika ragu, karena bertanya lebih murah daripada memperbaiki [5].

Gambar 13.1 mengubah gagasan-gagasan ini menjadi tangga empat anak tangga untuk desain sehari-hari. Di anak tangga terbawah AI hanya menyarankan dan orang yang bertindak. Di anak tangga kedua AI mengusulkan tindakan lengkap dan tidak terjadi apa-apa sampai orang mengonfirmasi. Di anak tangga ketiga ia langsung bertindak, tetapi tindakannya mudah dibatalkan. Di anak tangga teratas ia bertindak lalu melapor. Anak tangga dipilih *per tindakan*, bukan per produk: asisten Bu Wati menyarankan layanan, mengusulkan lalu menunggu untuk pemesanan penjemputan, dan boleh mengirim sendiri pengingat harian "cucian Anda sudah siap".



Gambar 13.1 Tangga otonomi: risiko dan keterbalikan tiap tindakan menentukan seberapa banyak AI boleh berbuat tanpa bertanya, dengan contoh dari laundry untuk setiap anak tangga.

> **WAWASAN KUNCI:** Otonomi bukan sifat AI; ia sifat setiap tindakan. Asisten yang sama semestinya menyarankan satu hal, bertanya sebelum hal lain, dan langsung mengerjakan hal ketiga. Pembayaran, penghapusan, atau pesan yang dikirim atas nama pelanggan berada di bagian bawah tangga, sebaik apa pun modelnya.

13.3.3 Kepercayaan, ketergantungan, dan kalibrasi

Orang tidak sekadar percaya atau tidak percaya pada sistem; mereka *bersandar* padanya atau tidak, satu keputusan demi satu keputusan. Tinjauan Lee dan See tentang kepercayaan pada otomatisasi merumuskan tujuannya sebagai *ketergantungan yang tepat* (*appropriate reliance*): bersandar pada otomatisasi ketika ia benar dan tidak bersandar ketika ia salah [6]. Dua galat mungkin terjadi. *Ketergantungan berlebih* (*over-reliance*, disebut juga bias otomatisasi) adalah menerima saran yang salah; *ketergantungan kurang* (*under-reliance*) adalah menolak saran yang benar. Dari log yang mencatat untuk tiap kasus apakah AI benar dan apakah orang menerimanya, kedua laju itu adalah

$$\text{ketergantungan berlebih} = \frac{\text{diterima dan AI salah}}{\text{AI salah}} \tag{13.2}$$

$$\text{ketergantungan kurang} = \frac{\text{ditolak dan AI benar}}{\text{AI benar}} \tag{13.3}$$

dan proporsi keputusan akhir yang benar, yang sesungguhnya dipedulikan toko, adalah

$$\text{tim benar} = \frac{\text{diterima dan AI benar} + \text{ditolak dan AI salah}}{\text{semua kasus}} \tag{13.4}$$

Tinjauan analitis CHI 2026 atas studi pengambilan keputusan manusia-AI oleh Raees, Khan, Lykourentzou, dan Papangelis menemukan bahwa, di seluruh literatur, "orang terlalu bersandar pada saran AI tanpa menelaahnya secara analitis", dan bahwa intervensi yang membuat orang menelaah, seperti meminta mereka memutuskan lebih dulu atau menunjukkan alasan saran AI, memperbaiki ketergantungan tetapi menuntut waktu dan tenaga [7]. Tidak ada makan siang gratis: setiap desain yang menurunkan ketergantungan berlebih menambah sedikit gesekan, dan perancanglah yang harus memutuskan di mana gesekan itu diletakkan.

Ketergantungan hanya bisa tepat bila keyakinan yang ditampilkan sistem jujur. Sistem disebut *terkalibrasi* bila, di antara semua kasus yang ia katakan 80% yakin, ia benar kira-kira 80% dari waktu. Jaringan saraf modern sering terlalu yakin [8]. Ukuran yang lazim adalah *galat kalibrasi harapan* (*expected calibration error*, ECE): kelompokkan kasus ke dalam B kelompok menurut keyakinan yang dinyatakan, lalu di tiap kelompok b bandingkan rerata keyakinan $\text{conf}(b)$ dengan proporsi yang benar $\text{acc}(b)$, dibobot dengan banyak kasus n_b dari N:

$$\text{ECE} = \sum_{b=1}^{B} \frac{n_b}{N} \left|\text{acc}(b) - \text{conf}(b)\right| \tag{13.5}$$

ECE sebesar 0 berarti keyakinan yang sepenuhnya jujur; ECE sebesar 0,10 berarti keyakinan yang dinyatakan rata-rata meleset sepuluh poin persentase. Bila ECE tinggi, menampilkan persentase menyesatkan, dan kata-kata seperti "yakin" dan "mohon periksa", yang dipilih dari ambang hasil pengukuran, lebih melayani orang.

13.3.4 Agen pada 2025 dan 2026

Dua perkembangan mengubah antarmuka AI antara 2025 dan 2026. Yang pertama adalah agen yang mengoperasikan perangkat lunak seperti manusia. ChatGPT agent, misalnya, menarasikan apa yang sedang dikerjakannya di layar, meminta izin sebelum tindakan yang berakibat seperti pembelian, mewajibkan pengawasan aktif untuk tugas seperti mengirim surel, dan membiarkan orang menyela, mengambil alih peramban, atau menghentikan tugas kapan saja [1]. Itulah tepatnya anak tangga kedua dan ketiga, ditambah "kendali global" pedoman G17.

Yang kedua adalah cara bersama bagi agen untuk menjangkau alat dan data. Model Context Protocol (MCP), yang diperkenalkan Anthropic pada akhir 2024, memungkinkan aplikasi AI terhubung ke "server" yang menyediakan alat seperti kalender, basis data, atau buku pesanan toko. Pada Desember 2025 MCP diserahkan kepada Agentic AI Foundation yang baru di bawah Linux Foundation, bersama proyek agen lainnya [9]. Bagi desain antarmuka hal ini penting, karena agen yang dapat menjangkau banyak alat memerlukan antarmuka yang menunjukkan alat *mana* yang akan dipakainya dan dengan data *apa*, serta membiarkan orang menyetujui pemakaian itu.

13.3.5 Aturan pada 2026: keterbukaan dan persetujuan

Merancang antarmuka AI pada 2026 juga berarti memenuhi kewajiban hukum yang belum ada beberapa tahun lalu.

Di Uni Eropa, Pasal 50 AI Act mewajibkan orang diberi tahu ketika berinteraksi dengan sistem AI seperti chatbot, kecuali bila hal itu sudah jelas, dan mewajibkan konten buatan atau olahan AI

ditandai atau diungkapkan [10]. Kewajiban transparansi ini berlaku sejak 2 Agustus 2026. Sebuah "omnibus digital" yang disepakati pada 2026 menunda kewajiban sistem berisiko tinggi dalam undang-undang itu ke Desember 2027 dan Agustus 2028, tetapi mempertahankan kewajiban Pasal 50, hanya dengan masa tenggang singkat sampai 2 Desember 2026 untuk penandaan teknis konten oleh sistem yang sudah beredar [11]. Toko Indonesia yang melayani pelanggan di Eropa, atau produk yang dijual di sana, harus mengikuti aturan ini.

Di Indonesia, Undang-Undang Pelindungan Data Pribadi (UU PDP 27/2022) mengatur setiap asisten yang menangani nama, alamat, dan nomor telepon. Peraturan pelaksananya, PP 33/2026, mensyaratkan persetujuan yang diberikan secara bebas, spesifik, dan tidak ambigu, tidak digabungkan dengan syarat dan ketentuan, didahului informasi yang singkat dan akurat, dapat ditarik kapan saja, serta melarang permintaan persetujuan yang menipu dan penurunan layanan hanya karena persetujuan tidak diberikan [12]. Dua peraturan presiden tentang kecerdasan artifisial, satu tentang peta jalan AI nasional 2026–2029 dan satu tentang etika AI yang memilah pemakaian menjadi terlarang, diawasi ketat, dan berisiko lebih rendah, masih menunggu tanda tangan Presiden per September 2026 [13]. Aturan praktis bagi perancang sederhana dan berlaku di yurisdiksi mana pun: katakan bahwa itu mesin, katakan data apa yang dipakainya, dan biarkan orang menjangkau manusia.

13.4 Metode: Kanvas Interaksi AI

Prinsip menjadi berguna ketika menjadi rutinitas. **Tabel 13.3** adalah kanvas satu halaman yang diisi untuk setiap fitur AI sebelum satu baris kode pun ditulis. Setiap barisnya memaksa satu keputusan yang dituntut oleh keempat pertanyaan dalam **Tabel 13.1**.

Tabel 13.3 Kanvas interaksi AI, diisi untuk asisten pesanan laundry.

Baris	Putuskan dan tuliskan	Asisten pesanan
1. Tugas	Apa yang dikerjakan AI dalam satu kalimat, dan apa yang tidak	Menerima pesanan jemput; bukan keluhan, bukan pengembalian uang
2. Keterbukaan	Bagaimana dan kapan orang tahu bahwa ini AI	Baris pertama setiap chat
3. Masukan dan data	Apa yang dibaca dan disimpan; persetujuan yang diperlukan	Teks pesan, alamat jemput; persetujuan pada pesanan pertama
4. Ketidakpastian	Cara menampilkan keyakinan; ambangnya	Kata: yakin ≥ 0.9, mungkin ≥ 0.6, selebihnya diasumsikan
5. Tanya atau asumsikan	Bagian mana yang ditanyakan, mana yang boleh diasumsikan	Menanyakan berat dan waktu jemput; boleh mengasumsikan layanan dan kecepatan
6. Tindakan dan anak tangga	Setiap tindakan beserta anak tangganya	Harga: anak tangga 1; pemesanan: anak tangga 2; pengingat: anak tangga 4
7. Batal dan henti	Cara membalik atau menghentikan tiap tindakan	"no" sebelum pemesanan; UNDO selama satu pesan sesudahnya
8. Alih ke manusia	Cara menjangkau manusia, dan apa yang dilihatnya	STAFF; seluruh chat diteruskan
9. Ukuran	Apa yang dicatat untuk memeriksa ketergantungan dan kalibrasi	Usulan, keyakinan, diterima atau diubah, hasil akhir
10. Kegagalan	Apa yang terjadi bila model tidak tersedia	Katakan demikian dan alihkan ke manusia; jangan pernah templat diam-diam

Isilah kanvas bersama orang-orang yang menanggung risikonya. Bagi Bu Wati, itu dirinya sendiri dan sepupunya yang mengendarai sepeda motor: dialah yang tahu bahwa penjemputan yang dipesan "jam 16" tanpa hari adalah penjemputan yang bermasalah. Kanvas yang hanya diisi

pemrogram biasanya membiarkan baris 7, 8, dan 10 kosong, karena pemrogram membayangkan sistemnya berjalan lancar.

Daftar periksa singkat melengkapi metode ini. Sebelum dirilis, seseorang yang tidak ikut membangun asisten harus dapat menjawab "ya" untuk setiap pertanyaan berikut hanya dari layar: *Apakah saya tahu ini mesin? Dapatkah saya melihat apa yang dipahaminya? Apakah saya tahu bagian mana yang ditebaknya? Dapatkah saya menolak? Dapatkah saya membatalkan? Dapatkah saya menjangkau manusia? Apakah ia bertanya sebelum membelanjakan uang saya atau bertindak atas nama saya?*

13.5 Membangun Asisten Pesanan

Asisten dalam bab ini tidak memakai model bahasa. Ia membaca pesan dengan beberapa aturan kata kunci, karena pokok bahasannya adalah antarmuka di sekitar tebakan, bukan tebakan itu sendiri. Aturan-aturan itu menghasilkan keluaran sejenis dengan keluaran model: nilai untuk setiap bagian pesanan, yang disebut *slot*, dan keyakinan untuk setiap slot. Mengganti aturan dengan model di kemudian hari hanya mengubah fungsi `parse`, dan semua yang dibangun bab ini di sekelilingnya tetap berlaku.

Program sengaja berbicara dalam bahasa Inggris karena toko melayani banyak mahasiswa asing; aturan kata kuncinya juga mengenali kata Indonesia seperti "setrika", "cuci", dan "kilat".

13.5.1 Program dalam Python

Kode 13.1 adalah seluruh asisten. Fungsi `parse` mengubah satu pesan menjadi empat slot beserta keyakinannya: layanan (setrika dikenali dengan 0,95, "wash" saja dengan 0,8, dan tanpa petunjuk sama sekali jatuh ke cuci lipat dengan 0,4), berat (angka yang diikuti "kg" atau "kilo"), kecepatan, dan waktu jemput. Fungsi `label` mengubah keyakinan menjadi kata-kata sesuai **Tabel 13.3**. Kelas `Assistant` menyimpan keadaan percakapan: slot yang sudah didengarnya, usulan yang menunggu ya atau tidak, dan pemesanan terakhir, yang dapat dibatalkan selama satu pesan berikutnya.

Kode 13.1 `laundry_assistant.py`: asisten pesanan, dengan kata keyakinan, pertanyaan untuk slot yang kosong, konfirmasi, pembatalan, dan jalan ke manusia.

```
"""Asisten pesanan Profitina Laundry: usul, keyakinan, tanya, lalu bertindak."""
import re
import sys

RATES = {"wash + fold": 8000, "wash + iron": 10000, "iron only": 6000}
MIN_KG = 3.0
DISCLOSURE = "[Automated assistant. Type STAFF at any time to reach a person.]"


def label(conf):
    # Kata, bukan persen: pelanggan membaca kata lebih cepat daripada angka.
    if conf >= 0.9:
        return "sure"
    if conf >= 0.6:
        return "likely - please check"
    return "assumed - please check"


def parse(text):
    """Ubah satu pesan menjadi slot, masing-masing pasangan (nilai, keyakinan)."""
    t = text.lower()
    slots = {}
    if "iron only" in t or "setrika saja" in t:
        slots["service"] = ("iron only", 0.95)
    elif "iron" in t or "setrika" in t:
        slots["service"] = ("wash + iron", 0.95)
    elif "wash" in t or "cuci" in t:
        slots["service"] = ("wash + fold", 0.8)
    else:
        slots["service"] = ("wash + fold", 0.4)
    m = re.search(r"(\d+(?:[.,]\d)?)\s*(kg|kilo)", t)
    if m:
        slots["kg"] = (float(m.group(1).replace(",", ".")), 0.95)
    else:
        slots["kg"] = (None, 0.0)
    fast = any(w in t for w in ("express", "kilat", "urgent", "today"))
    slots["speed"] = ("express", 0.9) if fast else ("regular", 0.5)
    m = re.search(r"(tomorrow|today)?\s*at\s*(\d{1,2})(?::(\d\d))?", t)
    if m:
        day = m.group(1) or "tomorrow"
        conf = 0.95 if m.group(1) else 0.7
        slots["pickup"] = ("%s %02d:%s" % (day, int(m.group(2)),
                                           m.group(3) or "00"), conf)
    else:
        slots["pickup"] = (None, 0.0)
    return slots
```

Kode 13.1 (Lanjutan) *laundry_assistant.py: asisten pesanan, dengan kata keyakinan, pertanyaan untuk slot yang kosong, konfirmasi, pembatalan, dan jalan ke manusia.*

```
def price(slots):
    kg = max(slots["kg"][0], MIN_KG)
    tenths = round(kg * 10)
    total = RATES[slots["service"][0]] * tenths // 10
    if slots["speed"][0] == "express":
        total = total * 3 // 2
    return kg, total


class Assistant:
    def __init__(self):
        self.known = {}          # slot yang diingat lintas pesan
        self.pending = None      # usulan yang menunggu ya/tidak
        self.booked = None       # pemesanan terakhir, bisa dibatalkan satu giliran
        self.undo_turns = 0

    def reply(self, text):
        t = text.strip().lower()
        self.undo_turns -= 1
        if t in ("staff", "human", "person"):
            return ["Passing you to Mbak Dewi (07:00-21:00).",
                    "She sees this whole chat; no need to repeat it."]
        if t == "undo":
            if self.booked and self.undo_turns >= 0:
                self.booked = None
                return ["Cancelled. Nothing will be picked up."]
            return ["There is nothing left to undo."]
        if self.pending and t in ("yes", "ok", "confirm", "ya"):
            self.booked, self.pending = self.pending, None
            self.undo_turns = 1
            return ["Booked: pickup %s." % self.booked["pickup"][0],
                    "Type UNDO in your next message to cancel."]
        if self.pending and t in ("no", "change", "tidak"):
            self.pending = None
            return ["No problem, nothing was booked.",
                    "Tell me what to change."]
        for name, (value, conf) in parse(text).items():
            # Pertahankan ucapan pelanggan sebelumnya kecuali pesan baru
            # menyatakannya lebih jelas.
            if name not in self.known or conf > self.known[name][1]:
                self.known[name] = (value, conf)
        slots = self.known
        if slots["kg"][0] is None:
            return ["How many kilograms, roughly? If you are not sure,",
```

Kode 13.1 (Lanjutan) *laundry_assistant.py: asisten pesanan, dengan kata keyakinan, pertanyaan untuk slot yang kosong, konfirmasi, pembatalan, dan jalan ke manusia.*

```
                    "we weigh it at pickup and send the exact price",
                    "before washing."]
        if slots["pickup"][0] is None:
            return ["When should we pick it up? Example: tomorrow at 9."]
        kg, total = price(slots)
        lines = ["Here is what I understood:"]
        for name in ("service", "kg", "speed", "pickup"):
            value, conf = slots[name]
            shown = "%g kg" % value if name == "kg" else value
            lines.append("  %-8s %-16s (%s)" % (name, shown, label(conf)))
        lines.append("  price    %g kg x Rp%s%s = Rp%s" % (
            kg, format(RATES[slots["service"][0]], ","),
            " x 1.5" if slots["speed"][0] == "express" else "",
            format(total, ",")))
        if kg > slots["kg"][0]:
            lines.append("  (minimum charge is %g kg)" % MIN_KG)
        lines.append("Book this pickup? (yes / no)")
        self.pending = dict(slots)
        return lines


def run(path):
    number = 0
    bot = None
    for raw in open(path, encoding="utf-8"):
        line = raw.rstrip("\n")
        if line.startswith("#"):
            continue
        if not line.strip():
            bot = None
            continue
        if bot is None:
            number += 1
            bot = Assistant()
            print("=== Conversation %d ===" % number)
            print("Bot : " + DISCLOSURE)
        print("You : " + line)
        for i, out in enumerate(bot.reply(line)):
            print(("Bot : " if i == 0 else "      ") + out)


if __name__ == "__main__":
    run(sys.argv[1] if len(sys.argv) > 1 else "../data/conversations.txt")
```

Empat keputusan desain tampak di kode. Keterbukaan dicetak sebelum hal lain. Bila berat atau waktu jemput tidak ada, asisten bertanya alih-alih menebak (pedoman G10), dan pertanyaannya tentang berat memberi jalan keluar bagi pelanggan yang tidak tahu. Setiap slot yang diusulkan membawa labelnya, sehingga “assumed” muncul tepat di tempat nilai bawaan dipakai (pedoman G11). Dan gelung penggabungan mempertahankan ucapan pelanggan di pesan sebelumnya kecuali pesan baru menyatakannya lebih jelas, sehingga pelanggan tak perlu mengulang seluruh pesanan (pedoman G13).

13.5.2 Program dalam Java

Kode 13.2 adalah asisten yang sama dalam Java. Sebuah `record Slot` menyimpan nilai dan keyakinannya; peta-petanya berupa `LinkedHashMap` agar slot ditampilkan dalam urutan tetap, seperti di Python.

Kode 13.2 `LaundryAssistant.java`: asisten yang sama dalam Java.

```java
import java.nio.file.*;
import java.util.*;
import java.util.regex.*;

/** Asisten pesanan Profitina Laundry: usul, keyakinan, tanya, bertindak. */
public class LaundryAssistant {
    static final Map<String, Integer> RATES = Map.of(
        "wash + fold", 8000, "wash + iron", 10000, "iron only", 6000);
    static final double MIN_KG = 3.0;
    static final String DISCLOSURE =
        "[Automated assistant. Type STAFF at any time to reach a person.]";

    record Slot(Object value, double conf) {}

    static String label(double conf) {
        // Kata, bukan persen: pelanggan membaca kata lebih cepat daripada angka.
        if (conf >= 0.9) return "sure";
        if (conf >= 0.6) return "likely - please check";
        return "assumed - please check";
    }

    static String num(double x) {           // 4.0 -> "4", 2.5 -> "2.5"
        return x == Math.rint(x) ? String.valueOf((long) x) : String.valueOf(x);
    }

    static boolean any(String t, String... words) {
        for (String w : words) if (t.contains(w)) return true;
        return false;
    }

    static Map<String, Slot> parse(String text) {
        String t = text.toLowerCase();
        Map<String, Slot> s = new LinkedHashMap<>();
        if (any(t, "iron only", "setrika saja"))
            s.put("service", new Slot("iron only", 0.95));
        else if (any(t, "iron", "setrika"))
            s.put("service", new Slot("wash + iron", 0.95));
        else if (any(t, "wash", "cuci"))
            s.put("service", new Slot("wash + fold", 0.8));
        else s.put("service", new Slot("wash + fold", 0.4));
        Matcher m = Pattern.compile("(\\d+(?:[.,]\\d)?)\\s*(kg|kilo)").matcher(t);
        s.put("kg", m.find()
            ? new Slot(Double.parseDouble(m.group(1).replace(",", ".")), 0.95)
            : new Slot(null, 0.0));
        s.put("speed", any(t, "express", "kilat", "urgent", "today")
            ? new Slot("express", 0.9) : new Slot("regular", 0.5));
```

Kode 13.2 (Lanjutan) *LaundryAssistant.java: asisten yang sama dalam Java.*

```
        m = Pattern.compile("(tomorrow|today)?\\s*at\\s*(\\d{1,2})(?::(\\d\\d))?")
            .matcher(t);
        if (m.find()) {
            String day = m.group(1) != null ? m.group(1) : "tomorrow";
            double conf = m.group(1) != null ? 0.95 : 0.7;
            String min = m.group(3) != null ? m.group(3) : "00";
            s.put("pickup", new Slot(String.format("%s %02d:%s", day,
                Integer.parseInt(m.group(2)), min), conf));
        } else s.put("pickup", new Slot(null, 0.0));
        return s;
    }

    static long[] price(Map<String, Slot> s) {   // {kg dalam persepuluhan, rupiah}
        double kg = Math.max((Double) s.get("kg").value(), MIN_KG);
        long tenths = Math.round(kg * 10);
        long total = RATES.get((String) s.get("service").value()) * tenths / 10;
        if (s.get("speed").value().equals("express")) total = total * 3 / 2;
        return new long[] {tenths, total};
    }

    Map<String, Slot> known = new LinkedHashMap<>();  // slot yang diingat
    Map<String, Slot> pending = null;   // usulan yang menunggu ya/tidak
    Map<String, Slot> booked = null;    // pemesanan terakhir, bisa dibatalkan
    int undoTurns = 0;

    List<String> reply(String text) {
        String t = text.strip().toLowerCase();
        undoTurns--;
        if (Set.of("staff", "human", "person").contains(t))
            return List.of("Passing you to Mbak Dewi (07:00-21:00).",
                "She sees this whole chat; no need to repeat it.");
        if (t.equals("undo")) {
            if (booked != null && undoTurns >= 0) {
                booked = null;
                return List.of("Cancelled. Nothing will be picked up.");
            }
            return List.of("There is nothing left to undo.");
        }
        if (pending != null && Set.of("yes", "ok", "confirm", "ya").contains(t)) {
            booked = pending;
            pending = null;
            undoTurns = 1;
            return List.of("Booked: pickup " + booked.get("pickup").value() + ".",
                "Type UNDO in your next message to cancel.");
        }
        if (pending != null && Set.of("no", "change", "tidak").contains(t)) {
```

Kode 13.2 (Lanjutan) *LaundryAssistant.java: asisten yang sama dalam Java.*

```
            pending = null;
            return List.of("No problem, nothing was booked.",
                "Tell me what to change.");
        }
        for (var e : parse(text).entrySet()) {
            // Pertahankan ucapan pelanggan sebelumnya kecuali pesan baru
            // menyatakannya lebih jelas.
            Slot old = known.get(e.getKey());
            if (old == null || e.getValue().conf() > old.conf())
                known.put(e.getKey(), e.getValue());
        }
        var s = known;
        if (s.get("kg").value() == null)
            return List.of("How many kilograms, roughly? If you are not sure,",
                "we weigh it at pickup and send the exact price",
                "before washing.");
        if (s.get("pickup").value() == null)
            return List.of("When should we pick it up? Example: tomorrow at 9.");
        long[] p = price(s);
        double kg = p[0] / 10.0;
        List<String> lines = new ArrayList<>();
        lines.add("Here is what I understood:");
        for (String name : List.of("service", "kg", "speed", "pickup")) {
            Slot sl = s.get(name);
            String shown = name.equals("kg")
                ? num((Double) sl.value()) + " kg" : (String) sl.value();
            lines.add(String.format("  %-8s %-16s (%s)", name, shown,
                label(sl.conf())));
        }
        String service = (String) s.get("service").value();
        lines.add(String.format(Locale.US, "  price    %s kg x Rp%,d%s = Rp%,d",
            num(kg), RATES.get(service),
            s.get("speed").value().equals("express") ? " x 1.5" : "", p[1]));
        if (kg > (Double) s.get("kg").value())
            lines.add("  (minimum charge is " + num(MIN_KG) + " kg)");
        lines.add("Book this pickup? (yes / no)");
        pending = new LinkedHashMap<>(s);
        return lines;
    }
```

Kode 13.2 (Lanjutan) *LaundryAssistant.java: asisten yang sama dalam Java.*

```
    public static void main(String[] args) throws Exception {
        String path = args.length > 0 ? args[0] : "../data/conversations.txt";
        int number = 0;
        LaundryAssistant bot = null;
        for (String line : Files.readAllLines(Path.of(path))) {
            if (line.startsWith("#")) continue;
            if (line.isBlank()) { bot = null; continue; }
            if (bot == null) {
                number++;
                bot = new LaundryAssistant();
                System.out.println("=== Conversation " + number + " ===");
                System.out.println("Bot : " + DISCLOSURE);
            }
            System.out.println("You : " + line);
            List<String> out = bot.reply(line);
            for (int i = 0; i < out.size(); i++)
                System.out.println((i == 0 ? "Bot : " : "      ") + out.get(i));
        }
    }
}
```

Dua rincian membuat program Java dan Python mencetak teks yang persis sama. `%g` di Python mencetak 4.0 sebagai “4” dan 2.5 sebagai “2.5”; fungsi bantu `num` melakukan hal yang sama di Java. Dan harga dihitung dengan berat dalam persepuluhan kilogram dan aritmetika bilangan bulat, sehingga Persamaan (13.1) memberi rupiah yang sama di kedua bahasa tanpa kejutan pembulatan.

13.5.3 Menjalankannya

Berkas `data/conversations.txt` memuat tiga percakapan uji, dipisahkan baris kosong. Jalankan asisten dari folder `python` dengan `python laundry_assistant.py`, atau dari folder `java` dengan `javac -d out *.java` dan `java -cp out LaundryAssistant`. Keduanya mencetak transkrip yang sama:

```
=== Conversation 1 ===
Bot : [Automated assistant. Type STAFF at any time to reach a person.]
You : Hi, can you wash 4 kg of clothes and iron them? Pick up tomorrow at 9.
Bot : Here is what I understood:
        service  wash + iron      (sure)
        kg       4 kg             (sure)
        speed    regular          (assumed - please check)
        pickup   tomorrow 09:00   (sure)
        price    4 kg x Rp10,000 = Rp40,000
      Book this pickup? (yes / no)
You : yes
Bot : Booked: pickup tomorrow 09:00.
      Type UNDO in your next message to cancel.
You : undo
Bot : Cancelled. Nothing will be picked up.
=== Conversation 2 ===
Bot : [Automated assistant. Type STAFF at any time to reach a person.]
You : I have a bag of laundry, pick up at 16 please
Bot : How many kilograms, roughly? If you are not sure,
      we weigh it at pickup and send the exact price
      before washing.
You : about 2.5 kg
Bot : Here is what I understood:
        service  wash + fold      (assumed - please check)
        kg       2.5 kg           (sure)
        speed    regular          (assumed - please check)
        pickup   tomorrow 16:00   (likely - please check)
        price    3 kg x Rp8,000 = Rp24,000
        (minimum charge is 3 kg)
      Book this pickup? (yes / no)
You : yes
Bot : Booked: pickup tomorrow 16:00.
      Type UNDO in your next message to cancel.
You : undo
Bot : Cancelled. Nothing will be picked up.
You : undo
Bot : There is nothing left to undo.
=== Conversation 3 ===
Bot : [Automated assistant. Type STAFF at any time to reach a person.]
You : express wash 6 kilo, today at 13:30
Bot : Here is what I understood:
        service  wash + fold      (likely - please check)
        kg       6 kg             (sure)
        speed    express          (sure)
        pickup   today 13:30      (sure)
        price    6 kg x Rp8,000 x 1.5 = Rp72,000
      Book this pickup? (yes / no)
You : no
Bot : No problem, nothing was booked.
      Tell me what to change.
You : STAFF
Bot : Passing you to Mbak Dewi (07:00-21:00).
      She sees this whole chat; no need to repeat it.
```

Bacalah transkrip itu dengan berpegang pada **Tabel 13.1**. Pada percakapan pertama pelanggan memberi semuanya kecuali kecepatan; asisten mengusulkan, menandai “regular” sebagai asumsi, memesan setelah “yes”, dan membatalkan setelah “undo”. Pada percakapan kedua, berat tidak ada, jadi asisten bertanya; ketika pelanggan menjawab “about 2.5 kg”, ia mengingat jam jemput dari pesan pertama, menandai harinya sebagai sekadar mungkin, dan menjelaskan tarif minimum. “undo” yang kedua datang terlambat dan ditolak dengan sopan: jendela pembatalan panjangnya satu pesan, sebuah pilihan desain yang dapat diubah Bu Wati. Pada percakapan ketiga, pelanggan menolak lalu meminta manusia, yang menerima seluruh chat.

13.6 Mengukur Ketergantungan dan Kalibrasi

Asisten yang menampilkan keyakinan harus diperiksa: jujurkah keyakinannya, dan tepatkah cara orang bersandar padanya? Dimas merencanakan studi kecil untuk fitur berikutnya, yaitu saran kepada staf apakah suatu pakaian aman dicuci mesin, berdasarkan foto label perawatannya. Staf melihat saran AI lalu menerima atau menolaknya. Dua versi layar dibandingkan: versi A hanya menampilkan saran, sedangkan versi B menampilkan saran beserta keyakinannya dalam kata-kata dan satu baris alasan.

Sebelum menjalankan studi dengan staf sungguhan, Dimas menulis log *simulasi* dengan program di **Kode 13.3** untuk menguji analisisnya. Simulasi itu memuat dua asumsi yang semestinya ditemukan analisis: model terlalu yakin sebesar delapan poin persentase, dan staf pada versi B lebih jarang menerima bila keyakinannya rendah atau alasannya tampak salah.

Kode 13.3 `make_log.py`: menulis log simulasi dengan seed berisi 200 kasus yang ditinjau untuk setiap versi layar.

```
"""Tulis log simulasi peninjauan Bab 13 (dengan seed, dapat diulang)."""
import random

rng = random.Random(13)
with open("../data/reliance_log.csv", "w", encoding="utf-8") as f:
    f.write("version,case,confidence,ai_correct,accepted\n")
    for version in ("A", "B"):
        for case in range(1, 201):
            conf = round(rng.uniform(0.5, 1.0), 2)
            correct = rng.random() < conf - 0.08   # model terlalu yakin
            if version == "A":                     # hanya saran
                p = 0.90
            else:                                  # keyakinan + alasan ditampilkan
                p = 0.93 if conf >= 0.8 else 0.50
                if not correct:
                    p -= 0.25                      # alasannya membuka kesalahan
            accepted = rng.random() < p
            f.write("%s,%d,%.2f,%d,%d\n" % (version, case, conf,
                                            correct, accepted))
```

Kode 13.4 menghitung tiga laju pada Persamaan (13.2), Persamaan (13.3), dan Persamaan (13.4) untuk setiap versi, serta tabel kalibrasi dan ECE pada Persamaan (13.5) dengan lima kelompok selebar 0,1 antara 0,5 dan 1,0.

Kode 13.4 `reliance.py`: ketergantungan berlebih, ketergantungan kurang, keputusan akhir yang benar, dan galat kalibrasi harapan.

```
"""Ukur sandaran staf pada saran AI dan kejujuran keyakinannya."""
import csv
import sys


def load(path):
    with open(path, encoding="utf-8") as f:
        return [(r["version"], float(r["confidence"]), r["ai_correct"] == "1",
                 r["accepted"] == "1") for r in csv.DictReader(f)]


def reliance(rows):
    right = [r for r in rows if r[2]]
    wrong = [r for r in rows if not r[2]]
    over = sum(r[3] for r in wrong) / len(wrong)          # diterima padahal salah
    under = sum(not r[3] for r in right) / len(right)     # ditolak padahal benar
    # Keputusan akhir benar bila staf menerima saran yang benar
    # atau menolak saran yang salah.
    team = (sum(r[3] for r in right) + sum(not r[3] for r in wrong)) / len(rows)
    return len(rows), len(right) / len(rows), over, under, team


def calibration(rows, bins=5):
    """Galat kalibrasi harapan atas kelompok keyakinan selebar sama."""
    table, ece = [], 0.0
    for b in range(bins):
        lo, hi = 0.5 + b * 0.5 / bins, 0.5 + (b + 1) * 0.5 / bins
        inside = [r for r in rows
                  if lo <= r[1] < hi or (b == bins - 1 and r[1] == hi)]
        if not inside:
            continue
        conf = sum(r[1] for r in inside) / len(inside)
        acc = sum(r[2] for r in inside) / len(inside)
        ece += len(inside) / len(rows) * abs(acc - conf)
        table.append((lo, hi, len(inside), conf, acc))
    return table, ece
```

Kode 13.4 (Lanjutan) *reliance.py: ketergantungan berlebih, ketergantungan kurang, keputusan akhir yang benar, dan galat kalibrasi harapan.*

```
def main(path):
    rows = load(path)
    print("version  cases  AI right  over-reliance  under-reliance  team right")
    for v in ("A", "B"):
        n, acc, over, under, team = reliance([r for r in rows if r[0] == v])
        print("%-7s  %5d  %7.1f%%  %12.1f%%  %13.1f%%  %9.1f%%" % (
            v, n, 100 * acc, 100 * over, 100 * under, 100 * team))
    table, ece = calibration(rows)
    print("\nconfidence bin   cases  mean confidence  actually right")
    for lo, hi, n, conf, acc in table:
        print("%.1f - %.1f      %5d  %14.3f  %14.3f" % (lo, hi, n, conf, acc))
    print("expected calibration error (ECE) = %.3f" % ece)


if __name__ == "__main__":
    main(sys.argv[1] if len(sys.argv) > 1 else "../data/reliance_log.csv")
```

Kode 13.5 adalah analisis yang sama dalam Java.

Kode 13.5 `Reliance.java`: analisis yang sama dalam Java.

```java
import java.nio.file.*;
import java.util.*;

/** Ukur sandaran staf pada saran AI dan kejujuran keyakinannya. */
public class Reliance {
    record Row(String version, double conf, boolean right, boolean accepted) {}

    static List<Row> load(String path) throws Exception {
        List<Row> rows = new ArrayList<>();
        List<String> lines = Files.readAllLines(Path.of(path));
        for (String line : lines.subList(1, lines.size())) {
            String[] f = line.split(",");
            rows.add(new Row(f[0], Double.parseDouble(f[2]), f[3].equals("1"),
                f[4].equals("1")));
        }
        return rows;
    }

    static double[] reliance(List<Row> rows) {
        int right = 0, wrong = 0, overN = 0, underN = 0;
        for (Row r : rows) {
            if (r.right()) { right++; if (!r.accepted()) underN++; }
            else { wrong++; if (r.accepted()) overN++; }
        }
        // Keputusan akhir benar bila staf menerima saran yang benar
        // atau menolak saran yang salah.
        double team = (double) ((right - underN) + (wrong - overN))
            / rows.size();
        return new double[] {rows.size(), (double) right / rows.size(),
            (double) overN / wrong, (double) underN / right, team};
    }

    public static void main(String[] args) throws Exception {
        List<Row> rows = load(args.length > 0 ? args[0]
            : "../data/reliance_log.csv");
        System.out.println(
            "version  cases  AI right  over-reliance  under-reliance  team right");
        for (String v : List.of("A", "B")) {
            List<Row> part = rows.stream()
                .filter(r -> r.version().equals(v)).toList();
            double[] x = reliance(part);
            System.out.printf("%-7s  %5d  %7.1f%%  %12.1f%%  %13.1f%%  %9.1f%%%n",
                v, (int) x[0], 100 * x[1], 100 * x[2], 100 * x[3], 100 * x[4]);
        }
        int bins = 5;
        double ece = 0;
```

Kode 13.5 (Lanjutan) *Reliance.java: analisis yang sama dalam Java.*

```
        System.out.println(
            "\nconfidence bin   cases  mean confidence  actually right");
        for (int b = 0; b < bins; b++) {
            double lo = 0.5 + b * 0.5 / bins, hi = 0.5 + (b + 1) * 0.5 / bins;
            int n = 0, ok = 0;
            double sum = 0;
            for (Row r : rows)
                if ((lo <= r.conf() && r.conf() < hi)
                        || (b == bins - 1 && r.conf() == hi)) {
                    n++; sum += r.conf(); if (r.right()) ok++;
                }
            if (n == 0) continue;
            double conf = sum / n, acc = (double) ok / n;
            ece += (double) n / rows.size() * Math.abs(acc - conf);
            System.out.printf("%.1f - %.1f      %5d  %14.3f  %14.3f%n",
                lo, hi, n, conf, acc);
        }
        System.out.printf("expected calibration error (ECE) = %.3f%n", ece);
    }
}
```

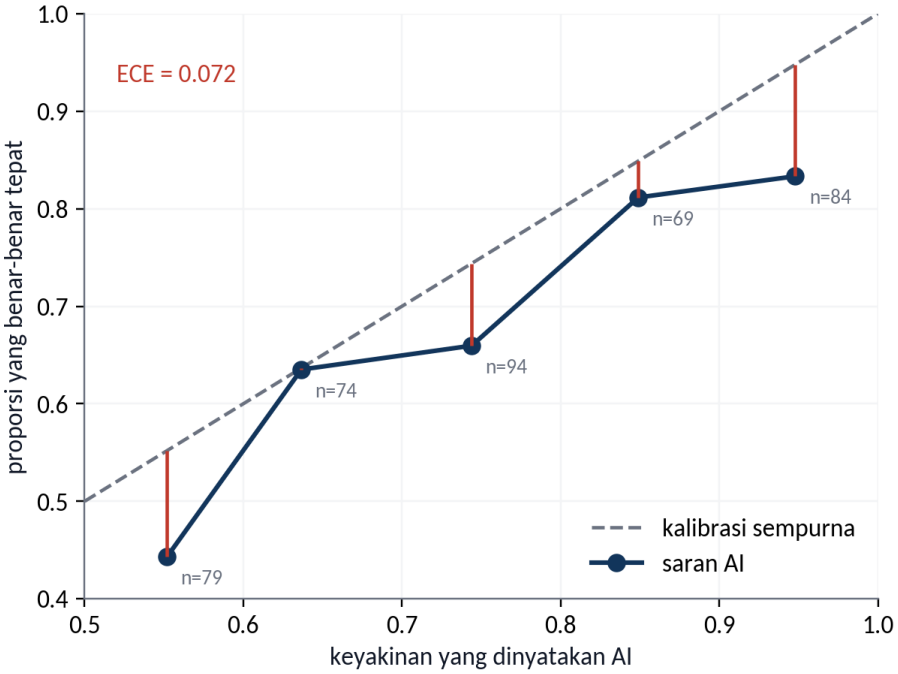
Keduanya mencetak:

```
version  cases  AI right  over-reliance  under-reliance  team right
A          200     66.0%          91.2%           6.1%       65.0%
B          200     69.0%          48.4%          26.1%       67.0%

confidence bin   cases  mean confidence  actually right
0.5 - 0.6         79           0.552           0.443
0.6 - 0.7         74           0.637           0.635
0.7 - 0.8         94           0.744           0.660
0.8 - 0.9         69           0.849           0.812
0.9 - 1.0         84           0.948           0.833
expected calibration error (ECE) = 0.072
```

Tiga temuan menonjol. Pada versi A, staf menerima 91,2% saran yang *salah*: ketergantungan berlebih klasik seperti yang digambarkan tinjauan CHI 2026 [7]. Versi B memangkas ketergantungan berlebih menjadi 48,4%, tetapi ketergantungan kurang naik dari 6,1% menjadi 26,1%, karena staf kini juga menolak banyak saran benar yang keyakinannya rendah. Proporsi keputusan akhir yang benar nyaris tak bergerak, dari 65,0% menjadi 67,0%. Menampilkan keyakinan lebih mengubah *cara* orang keliru daripada *seberapa sering*.

Paruh kedua keluaran menjelaskan sebabnya. **Gambar 13.2** menggambar tabel kalibrasi: di setiap kelompok kecuali satu, model lebih jarang benar daripada yang diklaimnya, dan di kelompok teratas ia mengklaim 0,948 tetapi benar 0,833. ECE-nya 0,072. Staf diminta memercayai angka yang tidak jujur.



Gambar 13.2 Kalibrasi model label perawatan hasil simulasi: titik di bawah diagonal berarti model mengklaim keyakinan melebihi yang pantas; garis merah adalah selisih yang dirata-ratakan ECE.

Pelajaran bagi desain sangat konkret. Kalibrasikan dulu, baru tampilkan keyakinan: tetapkan ambang kata dari apa yang benar-benar dicapai model di setiap kelompok, bukan dari apa yang diklaimnya. Lalu ukur lagi ketergantungannya, karena antarmuka yang tepat adalah antarmuka yang menaikkan proporsi keputusan akhir yang benar, bukan yang sekadar menurunkan ketergantungan berlebih.

13.7 Mengujinya

Aturan yang menentukan apa yang boleh dilakukan asisten sendiri harus diuji seperti kode lainnya. **Kode 13.6** memeriksa pengurai, harga dengan tarif minimum dan faktor kilat, dan, yang terpenting, konfirmasi serta jendela pembatalan.

Kode 13.6 `test_assistant.py`: uji penguraian, harga, dan jendela pembatalan.

```
"""Pemeriksaan asisten pesanan; jalankan: python -m unittest test_assistant"""
import unittest

from laundry_assistant import Assistant, parse, price


class AssistantTest(unittest.TestCase):
    def test_parse_and_price(self):
        s = parse("wash 4 kg and iron, tomorrow at 9")
        self.assertEqual(s["service"][0], "wash + iron")
        self.assertEqual(s["kg"][0], 4.0)
        self.assertEqual(s["pickup"][0], "tomorrow 09:00")
        self.assertEqual(price(s)[1], 40000)

    def test_minimum_and_express(self):
        s = parse("express wash 2 kg today at 8")
        self.assertEqual(price(s)[1], 36000)

    def test_confirm_and_undo_window(self):
        bot = Assistant()
        bot.reply("2 kg at 10")
        self.assertTrue(bot.reply("yes")[0].startswith("Booked"))
        bot.reply("hello")
        self.assertTrue(bot.reply("undo")[0].startswith("There is nothing"))


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

Kode 13.7 melakukan pemeriksaan yang sama dalam Java tanpa kerangka kerja pengujian, sehingga dapat dijalankan hanya dengan JDK.

Kode 13.7 `AssistantTest.java`: pemeriksaan yang sama dalam Java.

```
import java.util.*;

/** Pemeriksaan asisten pesanan; jalankan: java AssistantTest */
public class AssistantTest {
    static int failed = 0;

    static void check(boolean ok, String what) {
        System.out.println((ok ? "ok   " : "FAIL ") + what);
        if (!ok) failed++;
    }

    public static void main(String[] args) {
        var s = LaundryAssistant.parse("wash 4 kg and iron, tomorrow at 9");
        check(s.get("service").value().equals("wash + iron"), "service");
        check((Double) s.get("kg").value() == 4.0, "weight");
        check(s.get("pickup").value().equals("tomorrow 09:00"), "pickup");
        check(LaundryAssistant.price(s)[1] == 40000, "price 40,000");
        s = LaundryAssistant.parse("express wash 2 kg today at 8");
        check(LaundryAssistant.price(s)[1] == 36000, "minimum 3 kg, express");
        var bot = new LaundryAssistant();
        bot.reply("2 kg at 10");
        check(bot.reply("yes").get(0).startsWith("Booked"), "confirm books");
        bot.reply("hello");
        check(bot.reply("undo").get(0).startsWith("There is nothing"),
            "undo expires after one message");
        System.out.println(failed == 0 ? "all checks passed" : failed + " failed");
        System.exit(failed == 0 ? 0 : 1);
    }
}
```

`python -m unittest test_assistant` melaporkan tiga uji yang lulus; versi Java mencetak:

```
ok   service
ok   weight
ok   pickup
ok   price 40,000
ok   minimum 3 kg, express
ok   confirm books
ok   undo expires after one message
all checks passed
```

Perhatikan apa yang dilindungi uji terakhir. “Undo expires after one message” bukan rincian implementasi; itu janji yang dibuat antarmuka (“Type UNDO in your next message”). Uji yang mengunci janji itu mencegah perubahan di kemudian hari merusaknya diam-diam.

13.8 Membuatnya Lebih Baik

Asisten di Subbab 13.5 mengikuti pedoman, tetapi seminggu pemakaian nyata akan menyingkap lebih banyak hal. Empat perbaikan langsung mengalir dari teori di Subbab 13.3.

Jelaskan tebakannya, secara singkat. “tomorrow 16:00 (likely – please check)” memberi tahu pelanggan *bahwa* harinya ditebak, tetapi tidak *mengapa*. Menambahkan “karena Anda menulis pada malam hari dan tidak menyebut hari” hanya memakan satu baris dan memungkinkan pelanggan mengoreksi penalarannya, bukan hanya hasilnya. Buatlah penjelasan tetap pendek: panduan PAIR mengingatkan bahwa penjelasan panjang dilewati [3].

Tetapkan ambang dari data. Kata “sure” dan “likely” harus bermakna sesuatu yang terukur. Setelah pesanan nyata tercatat, hitung tabel kalibrasi seperti **Gambar 13.2** untuk setiap slot, lalu geser ambangnya agar “sure” benar setidaknya sembilan dari sepuluh kali.

Tanyakan satu hal sekali waktu, menurut urutan biaya. Asisten menanyakan berat sebelum waktu jemput karena berat yang salah mengubah harga. Bila dua slot kosong, tanyakan dulu slot yang galatnya paling mahal.

Simpan catatan yang dapat dilihat pelanggan. Setiap pemesanan sebaiknya diakhiri ringkasan pendek yang dapat diteruskan atau ditangkap layar oleh pelanggan: layanan, berat, harga, waktu jemput, dan nomor telepon toko. Catatan tertulis mengubah perselisihan (“saya bilang jam 9, bukan 16”) menjadi pemeriksaan cepat.

Perbaikan kelima menggoda tetapi keliru: membiarkan asisten memesan tanpa konfirmasi bila semua slot “sure”. Pemesanan mengirim seseorang bersepeda motor melintasi kota. Bahkan dengan keyakinan tinggi, tindakan itu tetap di anak tangga kedua, tempat satu kata “yes” hanya memakan satu detik pelanggan.

13.9 Memakainya Sungguhan

Kanvas pada **Tabel 13.3** dan kode bab ini dapat disesuaikan untuk usaha kecil mana pun yang menerima pesanan lewat chat. Ganti tabel `RATES` dan kata-kata di `parse` dengan layanan Anda sendiri, pertahankan keempat pertanyaan, dan isi kesepuluh baris kanvas sebelum menghubungkan asisten ke akun WhatsApp Business sungguhan.

Ambil contoh usaha katering Pak Joko, yang menerima pesanan nasi kotak untuk rapat kantor. Pesanannya punya slot menu, jumlah kotak, waktu antar, dan alamat, serta satu slot baru: kebutuhan diet. Slot itu *tidak pernah* boleh diasumsikan. Catatan “tanpa kacang” yang terlewat bukan galat kecil; bagi pelanggan yang alergi kacang hal itu berbahaya, jadi asisten harus selalu

menanyakan alergi dan meneruskan setiap penyebutan alergi kepada manusia. Aturan umumnya: untuk slot yang nilai bawaannya bisa mencelakakan orang bila salah, tidak ada nilai bawaan.

Tiga hal praktis berlaku bagi setiap pemakaian nyata. Pertama, catat setiap usulan, keyakinannya, dan apa yang dilakukan pelanggan terhadapnya, karena log itulah yang diperlukan Subbab 13.6. Mintalah persetujuan untuk pencatatan ini sebagaimana disyaratkan PP 33/2026, dalam kalimat tersendiri yang sederhana [12]. Kedua, putuskan sejak awal apa yang dilakukan asisten ketika model atau jaringan mati: sesuai baris 10 kanvas, ia mengatakannya dan mengalihkan ke manusia. Ketiga, bacalah contoh chat nyata setiap minggu. Aturan kata kunci dan ambangnya akan perlu disetel, dan hanya pesan nyata yang menunjukkan caranya.

13.10 IMK dalam Praktik: Profitina

Pertanyaan desain yang sama muncul di Profitina, platform AI Visibility (GEO) di app.profitina.com yang mengukur seberapa siap situs sebuah merek dikutip oleh mesin jawaban AI dan membantu memperbaiki temuannya. Content Engine-nya menyusun draf konten GEO serta bantuan schema.org dan robots.txt dengan Qwen2.5 yang berjalan lokal di server Profitina sendiri, sehingga data pengguna tidak pernah meninggalkan server untuk pemrosesan AI.

Tiga keputusan di Profitina sejalan dengan bab ini. Pertama, *kegagalan yang jujur*: bila tidak ada model yang tersedia, aplikasi mengatakannya alih-alih diam-diam menyajikan templat. Itulah baris 10 kanvas. Kedua, *gerbang mutu*: draf AI dinilai terhadap templat bawaan dan hanya disajikan bila benar-benar lebih baik; setelah revisi September 2026, draf AI menang pada 5 dari 6 kasus tolok ukur, yang berarti gerbang itu masih memilih templat sekali dari enam dan mengatakannya. Gerbang itu adalah keputusan kalibrasi yang dibuat atas nama pengguna: sistem hanya menampilkan keluaran AI di tempat keluaran itu layak. Ketiga, *aplikasi menyarankan, pengguna memutuskan*: setiap skor dan saran disertai sinyal penyebabnya, dan tidak ada yang diterbitkan atas nama pengguna tanpa tindakan eksplisit, seperti ditunjukkan fitur “Share to X” manual dengan antrean dan kuota hariannya. Alur umpan balik (kirim, kurasi, dataset, latih dan ukur, rilis) adalah pedoman “belajar seiring waktu” yang dibuat tampak, dan pengguna dapat melihat ke mana umpan baliknya pergi.

13.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Menyembunyikan bahwa ia mesin.** Nama dan foto manusia pada chatbot meruntuhkan kepercayaan begitu pelanggan tahu, dan di Uni Eropa melanggar Pasal 50 [10]. Katakan “asisten otomatis” di pesan pertama.
- **Menebak hal yang harus ditanyakan.** Mengisi berat, alamat, atau alergi dengan nilai bawaan menghemat satu pesan dan mengorbankan satu pesanan. Asumsikan hanya slot yang nilai kelirunya murah dan terlihat.
- **Menampilkan persentase yang tidak terkalibrasi.** “92% yakin” mengundang orang untuk memercayainya. Ukur kalibrasinya dulu (Persamaan (13.5)); bila buruk, pakai kata dengan ambang terukur.
- **Hanya mengukur ketergantungan berlebih.** Antarmuka yang membuat orang menolak semuanya tidak punya ketergantungan berlebih dan tidak berguna. Laporkan Persamaan (13.2), Persamaan (13.3), dan Persamaan (13.4) bersama-sama.
- **Satu tingkat otonomi untuk seluruh produk.** Menempatkan semua tindakan di anak tangga yang sama entah merecoki orang dengan hal remeh, entah membiarkan sistem membelanjakan uang tanpa bertanya. Putuskan per tindakan (**Gambar 13.1**).

- **Tanpa jalan keluar.** Chatbot yang tak dapat mengalihkan ke manusia menjebak justru pelanggan dengan masalah tersulit. Tawarkan pengalihan kapan saja dan teruskan seluruh percakapan.
- **Melupakan percakapan.** Asisten yang menanyakan lagi apa yang dikatakan pelanggan dua pesan sebelumnya terasa bodoh. Simpan slot lintas pesan, seperti yang dilakukan gelung penggabungan.
- **Pembatalan yang bukan pembatalan.** "Batal" yang hanya menyembunyikan pemesanan dari pelanggan sementara sepeda motor tetap berangkat lebih buruk daripada tanpa pembatalan. Ujilah janji itu, seperti dilakukan **Kode 13.6**.

13.12 Ringkasan Bab

- Antarmuka chat menjawab, kopilot mengusulkan di dalam alat lain, dan agen bertindak. Makin banyak yang dapat dilakukan sistem AI, makin besar kewajiban antarmukanya menunjukkan apa yang akan dilakukannya dan membiarkan orang menghentikannya.
- Setiap antarmuka AI harus menjawab empat pertanyaan (**Tabel 13.1**): apa yang dapat dilakukannya, seberapa yakin ia, apa yang akan dilakukannya berikutnya, dan bagaimana mengoreksi, menghentikan, atau membatalkannya.
- Pedoman interaksi manusia-AI [2] dan People + AI Guidebook [3] mengubah pertanyaan itu menjadi aturan konkret; **Tabel 13.2** memuat yang diperlukan asisten kecil.
- Otonomi dipilih per tindakan dari risiko dan keterbalikannya, pada tangga dari menyarankan sampai bertindak-lalu-melapor (**Gambar 13.1**) [4], [5].
- Ketergantungan yang tepat berarti menerima saran yang benar dan menolak yang salah; ketergantungan berlebih, ketergantungan kurang, dan keputusan akhir yang benar diukur dengan Persamaan (13.2), Persamaan (13.3), dan Persamaan (13.4). Keyakinan harus terkalibrasi sebelum ditampilkan (Persamaan (13.5)).
- Dalam studi simulasi, menampilkan keyakinan memangkas ketergantungan berlebih dari 91,2% menjadi 48,4% tetapi menaikkan ketergantungan kurang menjadi 26,1%, dan keputusan akhir yang benar hanya naik dari 65,0% menjadi 67,0%, karena modelnya terlalu yakin (ECE 0,072).
- Asisten pesanan, dalam Java dan Python, mengungkapkan jati dirinya, menanyakan slot yang kosong, menandai asumsi, mengusulkan dengan harga dari Persamaan (13.1), memesan hanya setelah "yes", memungkinkan pembatalan selama satu pesan, dan mengalihkan ke manusia bila diminta.
- Pada 2026, Pasal 50 EU AI Act mewajibkan pengungkapan interaksi dengan AI [10], [11]; di Indonesia, UU PDP 27/2022 dan PP 33/2026 mengatur persetujuan atas data pribadi yang ditangani asisten [12].

13.13 Soal Latihan

1. Dengan tangan, proses pesan "setrika saja 5 kg, tomorrow at 7" dengan aturan `parse`. Sebutkan setiap slot beserta keyakinan dan labelnya, serta harga menurut Persamaan (13.1). Lalu periksa jawaban Anda dengan menambahkan pesan itu sebagai percakapan baru di `data/conversations.txt`.
2. Untuk setiap tindakan asisten chat sebuah apotek berikut, pilih anak tangga pada **Gambar 13.1** dan beri satu kalimat alasan: (a) menampilkan jam buka; (b) memesan obat untuk diambil; (c) memesan ulang resep rutin; (d) membatalkan pemesanan kemarin; (e) mengirim pengingat minum obat.

3. Sebuah studi staf mencatat 300 kasus. AI benar pada 240 kasus; staf menerima 228 di antaranya dan 36 dari 60 kasus yang salah. Hitung ketergantungan berlebih, ketergantungan kurang, dan proporsi keputusan akhir yang benar dengan Persamaan (13.2), Persamaan (13.3), dan Persamaan (13.4).
4. Kasus-kasus sebuah model terbagi dalam dua kelompok: 100 kasus dengan rerata keyakinan 0,62, 58 di antaranya benar, dan 300 kasus dengan rerata keyakinan 0,91, 246 di antaranya benar. Hitung ECE dengan Persamaan (13.5) dan katakan apakah Anda akan menampilkan keyakinan model ini sebagai persentase.
5. Ubah asisten agar jendela pembatalan berlangsung dua pesan, bukan satu. Baris mana yang berubah di tiap bahasa, teks pesan mana yang harus ikut berubah, dan uji mana yang harus diperbarui?
6. Isi kesepuluh baris kanvas pada **Tabel 13.3** untuk asisten chat puskesmas yang menjawab pertanyaan jam buka dan menerima permintaan janji temu. Beri perhatian khusus pada baris 5, 6, dan 8.

13.14 Penyelesaian Soal Latihan

1. “setrika saja” cocok dengan aturan pertama, jadi layanan = iron only (0,95, sure). “5 kg” memberi berat = 5 (0,95, sure). Tidak ada kata kecepatan: regular (0,5, assumed – please check). “tomorrow at 7” memberi jemput = tomorrow 07:00 (0,95, sure). Harganya $P = 6000 \cdot \max(5,3) \cdot 1 = 30000$ rupiah, sehingga asisten mencetak `price  5 kg x Rp6,000 = Rp30,000` lalu bertanya “Book this pickup? (yes / no)”. Perhatikan bahwa “iron only” harus diuji sebelum “iron”; itulah sebabnya `if` untuk “iron only” berada paling awal di `parse`.

2. (a) Anak tangga 4, atau sekadar jawaban chat: menampilkan informasi tidak berefek samping. (b) Anak tangga 2: pemesanan menahan stok dan membuat janji; usulkan dengan nama obat dan waktu ambil, lalu tunggu “ya”. (c) Anak tangga 2, tidak pernah lebih tinggi: obat resep adalah urusan kesehatan, dan apoteker harus dapat memeriksanya; asisten mengusulkan, orang mengonfirmasi, apoteker menyetujui. (d) Anak tangga 2 dengan pernyataan jelas tentang apa yang akan hilang; setelah pembatalan, tawarkan pemesanan ulang, karena pembatalan tidak selalu dapat dibalik bila stok telanjur terjual. (e) Anak tangga 4 hanya bila orang meminta pengingat sebelumnya dan dapat menghentikannya dengan satu kata; isinya mengingatkan, tidak pernah menyarankan perubahan dosis, dan sebaiknya berbunyi “tanyakan kepada apoteker atau dokter Anda bila ada yang terasa tidak beres”.

3. AI salah pada $300 - 240 = 60$ kasus. Ketergantungan berlebih $= 36/60 = 0.60$. Ketergantungan kurang: ditolak padahal benar $= 240 - 228 = 12$, jadi $12/240 = 0.05$. Keputusan akhir yang benar $= (228 + (60 - 36))/300 = 252/300 = 0.84$. Staf jarang menolak saran yang benar, tetapi menerima sebagian besar saran yang salah: desainnya perlu membuat saran yang salah lebih mudah dikenali.

4. Kelompok 1: acc $= 58/100 = 0.58$, selisih $|0.58 - 0.62| = 0.04$, bobot $100/400 = 0.25$. Kelompok 2: acc $= 246/300 = 0.82$, selisih $|0.82 - 0.91| = 0.09$, bobot 0.75. ECE $= 0.25 \cdot 0.04 + 0.75 \cdot 0.09 = 0.01 + 0.0675 = 0.0775$. Kelompok berkeyakinan tinggi melebih-lebihkan sembilan poin; tampilan “91%” akan menyesatkan. Tampilkan kata, dan berikan “sure” hanya di atas ambang yang akurasi terukurnya mencapai tingkat yang diperlukan toko.

5. Di Python ubah `self.undo_turns = 1` menjadi `self.undo_turns = 2`; di Java ubah `undoTurns = 1;` menjadi `undoTurns = 2;`. Janjinya harus ikut berubah: “Type UNDO in your next message to cancel.” menjadi, misalnya, “Type UNDO within your next two messages to cancel.” Uji “undo

expires after one message" kini harus mengirim dua pesan biasa sebelum pembatalan yang terlambat, dan uji baru perlu memeriksa bahwa pembatalan pada pesan kedua masih berhasil. Mengubah kode tanpa mengubah teks akan membuat antarmuka berbohong.

6. Salah satu kanvas yang mungkin. (1) Tugas: menjawab pertanyaan jam buka dan layanan, menerima *permintaan* janji temu; tidak pernah memberi nasihat medis. (2) Keterbukaan: baris pertama, "asisten otomatis Puskesmas X". (3) Data: nama, telepon, hari yang diinginkan, layanan; persetujuan dalam kalimat tersendiri sesuai PP 33/2026. (4) Ketidakpastian: kata dengan ambang. (5) Tanya atau asumsikan: selalu tanyakan layanan dan hari; jangan pernah mengasumsikan apa pun tentang gejala. (6) Tindakan: menjawab jam buka, anak tangga 4; permintaan janji temu, anak tangga 2, dikonfirmasi petugas sebelum menjadi janji; pengingat, anak tangga 4 hanya setelah orang memilih ikut. (7) Batal: batalkan permintaan dengan satu kata sampai petugas mengonfirmasinya. (8) Alih ke manusia: "STAFF" kapan saja; dan setiap pesan yang menyebut nyeri dada, sesak napas, perdarahan hebat, atau bayi demam tinggi langsung dijawab "silakan segera datang ke puskesmas atau layanan gawat darurat terdekat sekarang" dan diteruskan ke petugas. (9) Ukuran: pertanyaan terjawab, permintaan terkonfirmasi, pengalihan. (10) Kegagalan: bila asisten mati, balasan otomatis memberikan nomor telepon loket pendaftaran.

Daftar Pustaka

[1] OpenAI, "Introducing ChatGPT agent: bridging research and action," Jul. 17, 2025. [Online]. Available: https://openai.com/index/introducing-chatgpt-agent/

[2] 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, doi: 10.1145/3290605.3300233. Maintained as the Microsoft HAX Toolkit: https://www.microsoft.com/en-us/haxtoolkit/ai-guidelines/

[3] Google People + AI Research (PAIR), *People + AI Guidebook*. [Online]. Available: https://pair.withgoogle.com/guidebook-v2/ (diakses Sep. 2026).

[4] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Trans. Systems, Man, and Cybernetics – Part A*, vol. 30, no. 3, pp. 286–297, 2000.

[5] E. Horvitz, "Principles of mixed-initiative user interfaces," in *Proc. CHI '99*, Pittsburgh, PA, USA, 1999, pp. 159–166.

[6] J. D. Lee and K. A. See, "Trust in automation: Designing for appropriate reliance," *Human Factors*, vol. 46, no. 1, pp. 50–80, 2004.

[7] M. Raees, V. J. Khan, I. Lykourentzou, and K. Papangelis, "Do people appropriately rely on AI-advice? An analytical review of HCI research on human-AI decision-making," in *Proc. CHI '26*, Barcelona, Spain, Apr. 2026, doi: 10.1145/3772318.3791467.

[8] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. 34th Int. Conf. Machine Learning (ICML)*, 2017, pp. 1321–1330.

[9] Model Context Protocol, "MCP joins the Agentic AI Foundation," Dec. 9, 2025. [Online]. Available: https://blog.modelcontextprotocol.io/posts/2025-12-09-mcp-joins-agentic-ai-foundation/

[10] European Commission, "Transparency obligations under Article 50 of the AI Act," FAQ, *Shaping Europe's Digital Future*. [Online]. Available: https://digital-strategy.ec.europa.eu/en/faqs/transparency-obligations-under-article-50-ai-act (diakses Sep. 2026). Regulation (EU) 2024/1689.

[11] J. M. Loring, M. Ramsden, and A. R. Lee, "Yes, August 2 still matters: The EU approved a high-risk AI delay, but most transparency obligations remain," Jones Walker LLP, Jul. 16, 2026. [Online].

Available: https://www.joneswalker.com/en/insights/blogs/ai-law-blog/yes-august-2-still-matters-the-eu-approved-a-high-risk-ai-delay-but-most-trans.html

[12] Bisnis.com, “Aturan Turunan UU PDP: Persetujuan Data Konsumen Lebih Ketat,” Aug. 29, 2026, tentang Peraturan Pemerintah No. 33 Tahun 2026 sebagai pelaksana UU No. 27 Tahun 2022. [Online]. Available: https://teknologi.bisnis.com/read/20260829/84/1999753/aturan-turunan-uu-pdp-persetujuan-data-konsumen-lebih-ketat

[13] Selular.id, “Dua Perpres AI Tunggu Tanda Tangan Presiden Prabowo,” Sep. 4, 2026. [Online]. Available: https://selular.id/2026/09/dua-perpres-ai-tunggu-tanda-tangan-presiden-prabowo/

Bab 14

UX Berbasis Data: Analitik, Eksperimen, dan Metrik

Situs pemesanan sebuah warung ramai dikunjungi, tetapi pesanan tak kunjung datang. Pemiliknya punya pendapat tentang sebabnya, anaknya punya pendapat lain, dan perancang situsnya punya pendapat ketiga. Bab ini membahas cara menyelesaikan perdebatan semacam itu dengan apa yang benar-benar dilakukan orang, diukur secara jujur dan dengan persetujuan mereka. Di akhir bab Anda akan mampu memilih metrik yang diturunkan dari tujuan dengan kerangka HEART; menggambarkan antarmuka sebagai aliran peristiwa dan mengubah aliran itu menjadi corong; membaca statistik pemakaian Indonesia terkini secara kritis; merencanakan eksperimen A/B dengan hipotesis, metrik utama, metrik pagar, dan ukuran sampel yang dihitung di muka; memeriksa ketidaksesuaian rasio sampel sebelum memercayai hasil eksperimen; menghitung selisih beserta selang kepercayaannya dan uji-*z* dua proporsi; membangun semuanya dalam Java dan Python; serta mengumpulkan data dengan cara yang menghormati UU PDP 27/2022, peraturan pelaksananya PP 33/2026, dan aturan persetujuan analitik yang berlaku kini.

14.1 Checkout Bu Tini yang Sepi

Bu Tini menjual nasi rumahan dari sebuah warung di kompleks perumahan: nasi campur, soto, sayur yang berganti-ganti, dan es teh. Bertahun-tahun pelanggannya memesan lewat telepon atau datang langsung. Tahun lalu anaknya, Raka, membuat situs pemesanan sederhana yang ditautkan dari Instagram dan status WhatsApp ibunya: pilih hidangan dari menu, masukkan ke keranjang, ketik alamat antar, bayar, selesai. Ratusan orang membuka menu setiap minggu. Pesanan yang masuk jauh lebih sedikit daripada harapan Bu Tini.

Semua orang punya penjelasan. Bu Tini menduga fotonya kurang menggugah selera. Raka menduga menunya terlalu panjang. Seorang teman yang merancang situs web berkata warnanya kuno. Setiap penjelasan menuntut seminggu kerja yang berbeda, dan setiap penjelasan hanyalah tebakan. Yang tak diketahui siapa pun adalah *di mana* orang berhenti: di menu, di keranjang, di formulir alamat, atau di langkah pembayaran.

Situs itu dapat menjawab pertanyaan tersebut bila mencatat sejumlah kecil *peristiwa* (*event*) yang dipilih dengan baik, satu setiap kali pengunjung mencapai sebuah langkah. Menghitung berapa pengunjung yang mencapai setiap langkah menghasilkan *corong* (*funnel*), dan corong menunjukkan langkah tempat kebanyakan orang menyerah. Pada corong Bu Tini, penurunan paling curam ternyata berada antara checkout, tempat orang harus mengetik alamat lengkap dengan papan ketik ponsel, dan pembayaran. Raka membuat versi kedua: alamat pelanggan lama diingat, dan pembayaran cukup satu ketukan dengan QRIS, standar pembayaran kode QR Indonesia. Kini ada dua versi, dan pertanyaan baru: *benarkah versi baru lebih baik, atau ia kebetulan saja sedang mendapat minggu yang bagus?* Menjawab pertanyaan itu dengan jujur adalah tugas eksperimen terkendali, yaitu uji A/B.

14.2 Gagasannya dengan Kata-Kata Sederhana

14.2.1 Apa yang dikatakan orang dan apa yang dilakukannya

Bab 5 dan Bab 7 mengajarkan metode yang bertanya kepada orang dan mengamati mereka: wawancara, survei, uji *think-aloud*. Metode itu menjelaskan *mengapa*. Ia bekerja dengan segelintir orang. Analitik bekerja sebaliknya: ia mencatat *apa* yang dilakukan ribuan orang tanpa bertanya, dan ia tidak dapat mengatakan mengapa. Corong dapat menunjukkan bahwa 44% pengunjung Bu Tini meninggalkan checkout; hanya percakapan dengan beberapa di antara mereka yang mengungkap bahwa mengetik "Jl. Kenanga Blok C-12 No. 7, RT 03/RW 05" di ponsel kecil adalah penyebabnya. Kedua jenis bukti saling membutuhkan. Istilah *berbasis data* dalam arti *data-informed*, bukan *data-driven*, menegaskan bahwa angka memberi masukan bagi keputusan yang diambil oleh orang yang juga memahami konteksnya.

14.2.2 Peristiwa, corong, dan metrik

> **DEFINISI.** *Peristiwa* adalah catatan bahwa sesuatu terjadi di antarmuka: siapa (pengenal pengguna anonim), apa (nama seperti `add_to_cart`), kapan, dan kadang beberapa properti. *Corong* adalah daftar langkah berurutan beserta banyaknya pengguna yang mencapai setiap langkah. *Metrik* adalah angka yang dihitung dari peristiwa untuk memantau suatu tujuan, misalnya proporsi pengunjung yang menyelesaikan pesanan. *Laju konversi* adalah metrik berbentuk "pengguna yang melakukan X dibagi pengguna yang dapat melakukan X".

Untuk corong dengan langkah $1,2,\dots,K$ dan n_k pengguna yang mencapai langkah k, ada dua laju yang penting. *Konversi langkah* menyatakan berapa banyak pengguna di satu langkah yang mencapai langkah berikutnya:

$$s_k = \frac{n_k}{n_{k-1}}, \quad k = 2,\dots,K \tag{14.1}$$

dan *konversi keseluruhan* menyatakan berapa banyak yang memulai lalu sampai di akhir:

$$c = \frac{n_K}{n_1} = s_2 \cdot s_3 \cdots s_K \tag{14.2}$$

Bentuk perkalian pada Persamaan (14.2) menjelaskan mengapa corong berguna: setiap langkah melipatgandakan kehilangan dari langkah-langkah sebelumnya, sehingga memperbaiki langkah terlemah paling banyak menaikkan konversi keseluruhan.

14.2.3 Contoh yang dikerjakan dengan tangan

Misalkan dalam seminggu 1.000 orang membuka menu Bu Tini, 610 memasukkan sesuatu ke keranjang, 430 memulai checkout, 245 mencapai langkah pembayaran, dan 228 menyelesaikan pesanan. Menurut Persamaan (14.1) konversi langkahnya adalah $610/1000 = 0.61$, $430/610 = 0.705$, $245/430 = 0.570$, dan $228/245 = 0.931$. Menurut Persamaan (14.2) konversi keseluruhannya $228/1000 = 0.228$, dan memang $0.61 \cdot 0.705 \cdot 0.570 \cdot 0.931 \approx 0.228$.

Di mana Raka sebaiknya bekerja? Langkah dari checkout ke pembayaran kehilangan 43% orang yang mencapainya, lebih banyak daripada langkah lain. Bila checkout yang lebih baik menaikkan langkah itu dari 0,570 menjadi 0,66 dan tak ada lagi yang berubah, konversi keseluruhan menjadi $0.61 \cdot 0.705 \cdot 0.66 \cdot 0.931 \approx 0.264$, kira-kira 36 pesanan lebih banyak per seribu pengunjung. Upaya yang sama pada langkah pembayaran, yang sudah 0,931, paling banyak memenangkan

$0.228 \cdot (1/0.931 - 1) \approx 0.017$, atau 17 pesanan. Corong tidak memberi tahu cara memperbaiki checkout; ia memberi tahu di mana perbaikan paling berbuah.

14.3 Teori dan Buktinya

14.3.1 Memilih metrik: kerangka HEART

Kesalahan paling umum dalam analitik adalah mengukur yang mudah (tampilan halaman, klik, lama di situs) alih-alih yang penting. Sebuah tim di Google mengusulkan kerangka HEART pada 2010 untuk mengatasi hal ini bagi pengalaman pengguna berskala besar [1]. HEART menamai lima jenis metrik yang berpusat pada pengguna: *Happiness* (kebahagiaan, yaitu sikap seperti kepuasan), *Engagement* (keterlibatan, kedalaman pemakaian), *Adoption* (adopsi, pengguna baru suatu produk atau fitur), *Retention* (retensi, pengguna yang kembali), dan *Task success* (keberhasilan tugas: efektivitas, efisiensi, galat). Tidak setiap produk memerlukan kelimanya. Disiplinnya terletak pada paruh kedua kerangka itu, proses *Goals–Signals–Metrics* (tujuan–sinyal–metrik): nyatakan dulu tujuannya dengan kata-kata, lalu sebut sinyal dalam perilaku pengguna yang menandakan kemajuan ke arah tujuan itu, dan baru kemudian definisikan metrik yang menghitung sinyal tersebut.

Tabel 14.1 menerapkan proses ini pada warung Bu Tini.

Tabel 14.1 Proses Goals–Signals–Metrics dari HEART diterapkan pada situs pemesanan warung; keterlibatan sengaja ditinggalkan, karena waktu yang lebih lama di halaman pemesanan bukanlah tujuan.

Kategori	Tujuan	Sinyal	Metrik
Happiness	Pelanggan merasa memesan itu mudah	Jawaban atas satu pertanyaan setelah memesan	Proporsi yang menjawab 6 atau 7 untuk “memesan itu mudah” (1–7)
Adoption	Pelanggan tetap yang datang langsung mencoba memesan daring	Pesanan daring pertama dari nomor telepon yang pernah dikenal	Pelanggan daring baru per minggu
Retention	Pelanggan daring memesan lagi	Pesanan kedua dalam 30 hari	Laju pesan ulang 30 hari
Task success	Orang yang ingin memesan dapat menyelesaikannya	Mencapai `order_done` setelah `view_menu`	Konversi keseluruhan, Persamaan (14.2); waktu memesan
Task success	Pesanan tiba dengan benar	Keluhan salah alamat	Pesanan salah alamat per 100

Meninggalkan *Engagement* adalah pilihan yang disengaja. Bagi jejaring sosial, waktu yang lebih lama mungkin memang tujuan. Bagi situs pemesanan, pelanggan yang menghabiskan sepuluh menit kemungkinan besar sedang kesulitan. Metrik yang naik ketika pengalaman memburuk lebih buruk daripada tidak punya metrik.

14.3.2 Di mana penggunanya: Indonesia pada 2026

Metrik selalu dibaca terhadap populasi asalnya. Menurut survei Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) yang diterbitkan Mei 2026, 81,7% penduduk Indonesia memakai internet, sekitar 235 juta orang, naik dari 80,66% pada 2025; penetrasinya 85,95% di Jawa tetapi 69,74% di Maluku dan Papua, 84,75% di kota dan 78,18% di desa, serta di atas 89% bagi Generasi Z dan milenial [2]. Warung di kompleks perumahan di Jawa dengan demikian melayani populasi yang hampir seluruhnya daring, sedangkan toko serupa di kabupaten pedesaan di Papua melayani populasi yang tiga dari sepuluh orangnya tidak daring.

Statistik perangkat menuntut kehati-hatian lebih. StatCounter, yang menghitung tampilan halaman di situs-situs yang memasang kode pelacaknya, melaporkan untuk Indonesia pada Agustus 2026 bahwa 60,59% tampilan halaman berasal dari komputer desktop, 38,98% dari perangkat seluler, dan 0,43% dari tablet [3]. Angka itu tampak bertentangan dengan pengalaman sehari-hari, ketika hampir semua orang memesan makanan lewat ponsel, dan justru menjadi pelajaran berharga dalam membaca metrik: proporsi tampilan halaman menghitung *tampilan halaman di situs tertentu*, bukan *orang*, dan dipengaruhi oleh tempat kode pelacak dipasang, oleh komputer kantor dan sekolah yang memuat banyak halaman, serta oleh lalu lintas otomatis. Satu-satunya komposisi perangkat yang penting bagi Bu Tini adalah yang ada di lognya sendiri, yang untuk situs pemesanan makanan yang ditautkan dari Instagram tentu didominasi ponsel. Selalu tanyakan apa yang dihitung sebuah angka sebelum memutuskan apa maknanya.

14.3.3 Eksperimen terkendali

Membandingkan pesanan minggu ini dengan minggu lalu tidak dapat membuktikan bahwa checkout baru berhasil, karena banyak hal berubah dari minggu ke minggu: gajian, cuaca, hari libur, diskon pesaing. *Eksperimen terkendali* menghilangkan perbedaan itu dengan menjalankan kedua versi bersamaan dan menempatkan setiap pengunjung pada salah satunya secara acak. Karena penempatannya acak, kedua kelompok serupa dalam segala hal kecuali versi yang dilihatnya, sehingga perbedaan hasil dapat diatribusikan kepada versi. Buku Kohavi, Tang, dan Xu, yang ditulis dari pengalaman bereksperimen di Microsoft, Google, dan LinkedIn, menjadi panduan bakunya [4].

Misalkan kelompok A (checkout lama, *kontrol*) memiliki n_A pengguna dengan x_A yang memesan, dan kelompok B (checkout baru, *perlakuan*) memiliki n_B dan x_B. Laju konversinya $p_A = x_A/n_A$ dan $p_B = x_B/n_B$. Pertanyaannya: apakah selisih $d = p_B - p_A$ lebih besar daripada yang dapat dihasilkan kebetulan semata? Di bawah hipotesis nol bahwa kedua versi sama baiknya, taksiran terbaik laju bersamanya adalah laju gabungan $\bar{p} = (x_A + x_B)/(n_A + n_B)$, dan statistik uji

$$z = \frac{p_B - p_A}{\sqrt{\bar{p}\,(1-\bar{p})\left(\frac{1}{n_A} + \frac{1}{n_B}\right)}} \tag{14.3}$$

kira-kira mengikuti distribusi normal baku bila kelompoknya besar. Nilai $|z|$ di atas 1,96 bersesuaian dengan nilai-p dua sisi di bawah 0,05. Nilai-p menjawab pertanyaan yang sempit: seandainya kedua versi benar-benar sama, seberapa sering selisih sebesar ini atau lebih muncul karena kebetulan?

Nilai-p saja tidak mengatakan apa-apa tentang besarnya efek, padahal itulah yang dipedulikan Bu Tini. Selang kepercayaan 95% bagi selisih memberi besaran sekaligus ketidakpastiannya:

$$d \pm 1.96\sqrt{\frac{p_A(1-p_A)}{n_A} + \frac{p_B(1-p_B)}{n_B}} \tag{14.4}$$

Bila seluruh selang berada di atas nol, versi baru lebih baik; seberapa jauh lebih baik terletak di suatu tempat di dalam selang itu.

Akhirnya, eksperimen harus cukup besar untuk mendeteksi selisih terkecil yang layak ditindaklanjuti, yaitu *efek minimum yang terdeteksi*. Untuk uji dua sisi pada taraf 5% dengan daya 80% (peluang 80% untuk mendeteksi efek sebenarnya sebesar itu), banyaknya pengguna yang diperlukan di *setiap* kelompok untuk mendeteksi perubahan dari p_1 ke p_2 kira-kira

$$n = \frac{(z_{\alpha/2} + z_{\beta})^2 [p_1(1-p_1) + p_2(1-p_2)]}{(p_2 - p_1)^2} \tag{14.5}$$

dengan $z_{\alpha/2} = 1.96$ dan $z_{\beta} = 0.8416$. Selisih kuadrat pada penyebut adalah bagian terpenting: memperkecil setengah efek yang ingin dideteksi melipatempatkan sampel yang diperlukan. Menghitung *n sebelum* eksperimen menentukan berapa lama eksperimen harus berjalan.

14.3.4 Eksperimen yang tepercaya: jebakan-jebakannya

Eksperimen dapat gagal dengan cara yang menghasilkan angka yang meyakinkan tetapi salah. Empat jebakan cukup umum sehingga perlu diperiksa setiap kali.

Ketidaksesuaian rasio sampel (*sample ratio mismatch*, SRM). Bila pengunjung dibagi 50/50, kedua kelompok semestinya kira-kira sama besar. Ketimpangan yang jelas berarti ada sesuatu yang menghilangkan pengguna dari salah satu kelompok: bug yang membuat versi baru mogok di sebagian ponsel, pengalihan halaman yang memutus pelacakan, atau penyaring bot yang memperlakukan kedua kelompok berbeda. Fabijan dan rekan-rekannya di Microsoft menemukan ketidaksesuaian semacam itu pada bagian yang tidak kecil dari eksperimen nyata dan berpendapat bahwa hasil dengan SRM tidak boleh dipercaya sebelum penyebabnya ditemukan [5]. Pemeriksaannya berupa uji khi-kuadrat atas ukuran kelompok yang teramati terhadap pembagian yang direncanakan. Dengan $n_A + n_B$ pengguna dan rencana 50/50, ukuran harapan setiap kelompok $e = (n_A + n_B)/2$ dan

$$\chi^2 = \frac{(n_A - e)^2}{e} + \frac{(n_B - e)^2}{e} \tag{14.6}$$

berderajat bebas satu. Karena pemeriksaan ini dijalankan pada setiap eksperimen, dipakai ambang yang ketat, lazimnya $p < 0.001$.

Mengintip. Melihat nilai-p setiap hari lalu berhenti begitu nilainya turun di bawah 0,05 menaikkan laju positif palsu jauh di atas 5%, karena fluktuasi kebetulan diberi banyak kesempatan melewati garis. Tetapkan ukuran sampel dengan Persamaan (14.5) di muka dan baca hasilnya sekali saja, di akhir, atau pakai metode sekuensial yang memang dirancang untuk pengamatan berulang [4].

Efek kebaruan dan efek belajar. Desain baru bisa menang pada hari-hari pertamanya hanya karena ia baru, atau kalah hanya karena pelanggan tetap harus mempelajarinya ulang. Jalankan eksperimen setidaknya satu siklus mingguan penuh, lebih baik dua, sehingga hari kerja, akhir pekan, dan hari gajian semuanya tercakup.

Hukum Goodhart. "Ketika sebuah ukuran menjadi sasaran, ia berhenti menjadi ukuran yang baik" adalah perumusan umum antropolog Marilyn Strathern atas pengamatan ekonom Charles Goodhart [6]. Bila Raka diganjar hanya berdasarkan konversi, ia dapat menaikkannya dengan menyembunyikan ongkos antar sampai setelah pembayaran, sebuah pola gelap (*dark pattern*) persis seperti yang dikecam Bab 12. Karena itu setiap eksperimen memerlukan *metrik pagar* (*guardrail metrics*) yang tidak boleh memburuk, seperti keluhan, pengembalian uang, dan pesanan salah alamat.

14.3.5 Data dengan persetujuan

Analitik termasuk pemrosesan data pribadi bila peristiwanya dapat dikaitkan dengan seseorang, dan nomor telepon atau alamat antar selalu dapat dikaitkan. Di Indonesia berlaku UU PDP

27/2022, dan peraturan pelaksananya, PP 33/2026, mensyaratkan persetujuan yang bebas, spesifik, dan tidak ambigu, tidak digabungkan dengan syarat dan ketentuan, didahului informasi yang singkat dan akurat, serta dapat ditarik kapan saja, dengan pemrosesan dihentikan dalam 3 × 24 jam setelah penarikan terverifikasi [7]. Layanan tidak boleh diturunkan hanya karena seseorang menolak memberi persetujuan, kecuali pemrosesan itu memang diperlukan untuk layanan tersebut. Bagi Bu Tini hal ini memisahkan dua jenis data: alamat yang diperlukan untuk mengantar pesanan, yang memang dituntut layanan, dan peristiwa analitik untuk memperbaiki situs, yang memerlukan persetujuan tersendiri.

Alat analitik bergerak ke arah yang sama. Google Analytics 4 memakai *consent mode*, yaitu halaman memberi tahu alat itu apakah pengunjung mengizinkan penyimpanan analitik (`analytics_storage`) dan penyimpanan iklan (`ad_storage`); sejak 15 Juni 2026 keduanya diatur terpisah, sehingga persetujuan analitik tidak lagi berarti apa pun bagi pemakaian data untuk iklan [8]. Situs kecil dapat menghindari sebagian besar kerumitan ini dengan hanya mengumpulkan peristiwa pihak pertama yang anonim di servernya sendiri, seperti yang diasumsikan program-program bab ini, dan dengan tidak pernah memasukkan alamat atau nomor telepon ke dalam analitik.

14.3.6 Metrik untuk fitur AI

Bab 13 menambahkan jenis fitur baru pada banyak produk: fitur yang menebak. Metriknya memerlukan kehati-hatian ekstra, karena metrik yang biasa justru mengganjar hal yang keliru. *Laju penerimaan saran*, yaitu proporsi saran AI yang diterima orang, naik ketika sarannya membaik, tetapi juga naik ketika orang berhenti membacanya, dan itulah ketergantungan berlebih. Bila diukur sendirian, ia tidak dapat membedakan keduanya. Tiga metrik tambahan membuat gambarannya jujur. *Laju suntingan* menghitung seberapa sering saran yang diterima diubah sebelum dipakai; banyak suntingan berarti saran itu hanya titik awal. Pasangan *penolakan-dan-hasil* mencatat, untuk setiap kasus ketika seseorang menolak AI, apakah orang itu atau AI yang ternyata benar, persis log yang dianalisis di Subbab 13.6. Dan sebuah *metrik pagar bahaya* menghitung galat yang paling penting, misalnya penjemputan keliru yang dipesan oleh asisten laundry.

Untuk eksperimen yang membandingkan dua versi fitur AI, metrik utamanya karena itu sebaiknya hasil yang dipedulikan usaha (pesanan yang selesai dengan benar, pertanyaan yang terjawab tanpa dialihkan), dengan laju penerimaan dilaporkan hanya sebagai konteks. Versi yang menaikkan penerimaan sementara proporsi keputusan akhir yang benar tetap datar telah membuat orang lebih percaya, bukan membuat sistem lebih berguna.

14.4 Metode: Rencana Eksperimen

Eksperimen hanya sebaik rencana yang ditulis *sebelum* ia dimulai. **Tabel 14.2** adalah templat satu halaman, yang diisi untuk checkout Bu Tini.

Tabel 14.2 Rencana eksperimen yang diisi untuk checkout warung; setiap baris diputuskan sebelum pengunjung pertama ditempatkan.

Butir	Tuliskan	Eksperimen checkout warung
Hipotesis	"Karena kami melihat X, kami yakin perubahan Y akan menyebabkan Z"	Karena 43% berhenti di langkah alamat, mengingat alamat dan QRIS satu ketukan akan menaikkan pesanan selesai
Varian	Kontrol dan perlakuan, secara persis	A: checkout sekarang; B: alamat diingat + QRIS
Unit dan pembagian	Siapa yang diacak, dan dengan rasio berapa	Setiap pengunjung (id anonim), 50/50
Metrik utama	Satu metrik penentu	Konversi keseluruhan, `order_done` / `view_menu`
Metrik pagar	Metrik yang tidak boleh memburuk	Pesanan salah alamat; keluhan; pembayaran gagal
Efek minimum terdeteksi	Perubahan terkecil yang layak ditindaklanjuti	+2 poin persentase
Ukuran sampel dan durasi	Dari Persamaan (14.5), dibulatkan ke minggu penuh	Sekitar 7.000 per varian; setidaknya dua minggu penuh
Pemeriksaan sebelum membaca	Apa yang harus lolos dulu	Rasio sampel (Persamaan (14.6)), $p \geq 0.001$; pencatatan diuji di kedua varian
Aturan keputusan	Hasil apa mengarah ke tindakan apa	Luncurkan B bila selang 95% pada Persamaan (14.4) di atas 0 dan metrik pagar aman
Persetujuan dan data	Apa yang dikumpulkan, dengan persetujuan apa	Hanya peristiwa anonim; persetujuan analitik tersendiri; tanpa alamat dalam peristiwa

Rencana ini punya dua guna. Ia mencegah tim memilih, setelah kejadian, metrik yang kebetulan tampak bagus. Dan ia membuat perbedaan pendapat menjadi produktif: bila Bu Tini yakin foto lebih penting, hipotesisnya mendapat rencana dan eksperimennya sendiri.

14.5 Membangun Analisis Corong

14.5.1 Log peristiwa

Program-program bab ini membaca log peristiwa dalam bentuk paling sederhana: berkas CSV dengan satu baris per peristiwa, `user_id,variant,step`. Sebelum situs Raka punya cukup lalu lintas nyata, ia menulis **Kode 14.1** untuk membuat log *simulasi* dengan sifat yang diketahui, sehingga analisisnya dapat diuji terhadap kebenaran yang ia tetapkan sendiri. Masing-masing dari 5.000 pengunjung simulasi ditempatkan di A atau B dengan lemparan koin, lalu menuruni corong dan melanjutkan setiap langkah dengan peluang yang tercantum di `GO_ON`; satu-satunya perbedaan antarvarian ada pada langkah checkout ke pembayaran, 0,58 untuk A dan 0,66 untuk B. Log kedua, `events_srm.csv`, identik kecuali bahwa 15% pengunjung varian B hilang diam-diam, seolah-olah bug pencatatan di ponsel lama membuang mereka.

Kode 14.1 `make_events.py`: menulis dua log peristiwa simulasi dengan seed, satu sehat dan satu dengan bug pencatatan di varian B.

```
"""Tulis log simulasi eksperimen checkout Bab 14 (dengan seed)."""
import random

STEPS = ["view_menu", "add_to_cart", "checkout", "pay", "order_done"]
GO_ON = {"A": [0.62, 0.70, 0.58, 0.93],     # checkout lama: ketik alamat
         "B": [0.62, 0.70, 0.66, 0.93]}     # baru: alamat diingat, QRIS


def write(path, lose_b=0.0, seed=14):
    rng = random.Random(seed)
    with open(path, "w", encoding="utf-8") as f:
        f.write("user_id,variant,step\n")
        for user in range(1, 5001):
            variant = "A" if rng.random() < 0.5 else "B"
            if variant == "B" and rng.random() < lose_b:
                continue            # bug pencatatan: pengguna ini lenyap
            f.write("u%04d,%s,%s\n" % (user, variant, STEPS[0]))
            for step, p in zip(STEPS[1:], GO_ON[variant]):
                if rng.random() >= p:
                    break
                f.write("u%04d,%s,%s\n" % (user, variant, step))


write("../data/events.csv")
write("../data/events_srm.csv", lose_b=0.15)
```

14.5.2 Corong dalam Python dan Java

Kode 14.2 membaca log menjadi himpunan pengenal pengguna untuk setiap varian dan langkah. Menghitung *pengguna yang berbeda*, bukan peristiwa, itu penting: pengunjung yang membuka menu lima kali adalah satu pengguna yang melihat menu, bukan lima. Untuk setiap langkah program mencetak banyaknya pengguna, konversi langkah menurut Persamaan (14.1), dan proporsi terhadap pengguna awal, yang pada langkah terakhir sama dengan Persamaan (14.2).

Kode 14.2 `funnel.py`: pengguna per langkah checkout dan per varian, dengan konversi langkah dan konversi keseluruhan.

```
"""Hitung pengguna di tiap langkah checkout, per varian, dari log peristiwa."""
import csv
import sys

STEPS = ["view_menu", "add_to_cart", "checkout", "pay", "order_done"]


def load(path):
    """Kembalikan {varian: {langkah: himpunan id pengguna}}."""
    users = {}
    with open(path, encoding="utf-8") as f:
        for r in csv.DictReader(f):
            steps = users.setdefault(r["variant"], {s: set() for s in STEPS})
            steps[r["step"]].add(r["user_id"])
    return users


def main(path):
    users = load(path)
    for variant in sorted(users):
        counts = [len(users[variant][s]) for s in STEPS]
        print("Variant %s" % variant)
        print("  step          users  from previous  from start")
        for i, (step, n) in enumerate(zip(STEPS, counts)):
            prev = n / counts[i - 1] if i else 1.0
            print("  %-12s %6d  %13.1f%%  %9.1f%%" % (
                step, n, 100 * prev, 100 * n / counts[0]))


if __name__ == "__main__":
    main(sys.argv[1] if len(sys.argv) > 1 else "../data/events.csv")
```

Kode 14.3 adalah program yang sama dalam Java. `TreeMap` menjaga varian dalam urutan abjad dan `LinkedHashMap` menjaga langkah dalam urutan corong, sehingga keluarannya sama dengan Python baris demi baris.

Kode 14.3 `Funnel.java`: corong yang sama dalam Java.

```java
import java.nio.file.*;
import java.util.*;

/** Hitung pengguna di tiap langkah checkout, per varian, dari log peristiwa. */
public class Funnel {
    static final List<String> STEPS = List.of(
        "view_menu", "add_to_cart", "checkout", "pay", "order_done");

    /** Kembalikan {varian: {langkah: himpunan id pengguna}}. */
    static Map<String, Map<String, Set<String>>> load(String path)
            throws Exception {
        Map<String, Map<String, Set<String>>> users = new TreeMap<>();
        List<String> lines = Files.readAllLines(Path.of(path));
        for (String line : lines.subList(1, lines.size())) {
            String[] f = line.split(",");
            var steps = users.computeIfAbsent(f[1], v -> {
                Map<String, Set<String>> m = new LinkedHashMap<>();
                for (String s : STEPS) m.put(s, new HashSet<>());
                return m;
            });
            steps.get(f[2]).add(f[0]);
        }
        return users;
    }

    public static void main(String[] args) throws Exception {
        var users = load(args.length > 0 ? args[0] : "../data/events.csv");
        for (String variant : users.keySet()) {
            int[] counts = new int[STEPS.size()];
            for (int i = 0; i < counts.length; i++)
                counts[i] = users.get(variant).get(STEPS.get(i)).size();
            System.out.println("Variant " + variant);
            System.out.println("  step          users  from previous  from start");
            for (int i = 0; i < counts.length; i++) {
                double prev = i > 0 ? (double) counts[i] / counts[i - 1] : 1.0;
                System.out.printf("  %-12s %6d  %13.1f%%  %9.1f%%%n",
                    STEPS.get(i), counts[i], 100 * prev,
                    100.0 * counts[i] / counts[0]);
            }
        }
    }
}
```

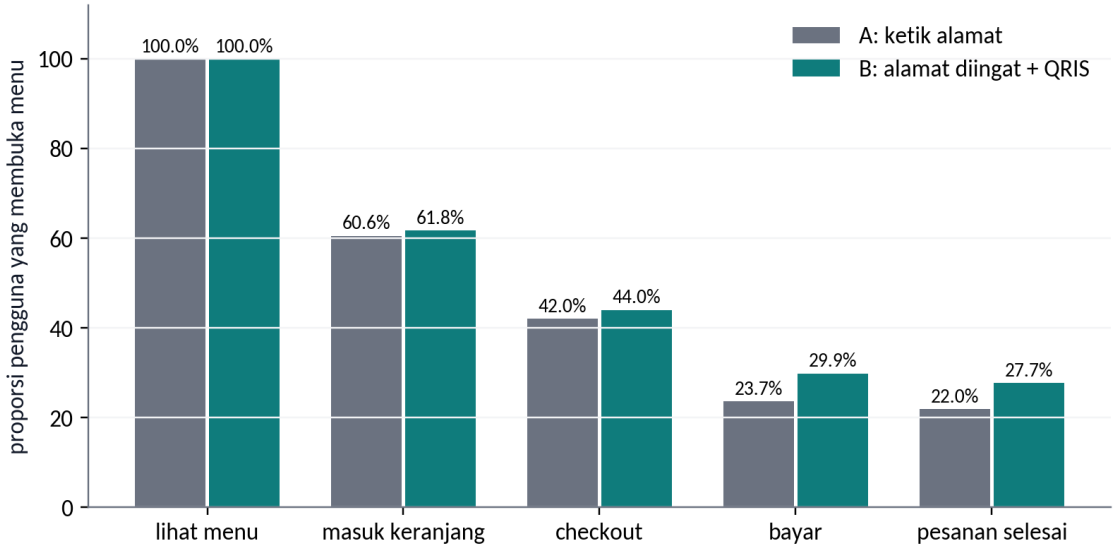
14.5.3 Menjalankannya

Dari folder `python`, `python make_events.py` menulis kedua log dan `python funnel.py` mencetak corongnya; dari folder `java`, `java -cp out Funnel` mencetak hal yang sama:

```
Variant A
  step          users  from previous  from start
  view_menu      2481          100.0%      100.0%
  add_to_cart    1503           60.6%       60.6%
  checkout       1043           69.4%       42.0%
  pay             588           56.4%       23.7%
  order_done      545           92.7%       22.0%
Variant B
  step          users  from previous  from start
  view_menu      2519          100.0%      100.0%
  add_to_cart    1556           61.8%       61.8%
  checkout       1109           71.3%       44.0%
  pay             753           67.9%       29.9%
  order_done      698           92.7%       27.7%
```

Gambar 14.1 menggambar angka-angka yang sama. Kedua varian nyaris identik sampai checkout, sesuai rancangan simulasi, lalu B mempertahankan 67,9% penggunanya di langkah

pembayaran sedangkan A mempertahankan 56,4%. Karena langkah-langkah berikutnya saling mengalikan, jaraknya melebar menjadi 27,7% lawan 22,0% di akhir.



Gambar 14.1 Corong checkout pada eksperimen simulasi: proporsi pengguna pembuka menu yang mencapai setiap langkah, untuk checkout lama (A) dan baru (B).

Corong seperti ini bersifat deskriptif. Ia menunjukkan bahwa pengguna B lebih sering membayar, tetapi dengan sendirinya ia tidak dapat menyingkirkan kemungkinan kebetulan atau pembagian yang rusak. Itulah tugas program berikutnya.

14.6 Membangun Pemeriksaan A/B

14.6.1 Program dalam Python

Kode 14.4 menjalankan pemeriksaan di Subbab 14.3 menurut urutan rencana di **Tabel 14.2**: pertama rasio sampel dengan Persamaan (14.6), dan hanya bila lolos, selisihnya, selang kepercayaannya dengan Persamaan (14.4), kenaikan relatifnya, statistik-z pada Persamaan (14.3) beserta nilai-p-nya, dan akhirnya ukuran sampel menurut Persamaan (14.5) yang diperlukan untuk mendeteksi perubahan dua poin dari laju dasar yang teramati.

Kode 14.4 `abtest.py`: pemeriksaan rasio sampel, selisih dengan selang kepercayaan 95%, uji-z, dan ukuran sampel.

```
"""Periksa uji A/B: rasio sampel dulu, lalu selisihnya, lalu daya uji."""
import math
import sys

from funnel import load


def erfc(x):
    """Fungsi galat komplementer (Numerical Recipes, galat < 1.2e-7)."""
    z = abs(x)
    t = 1 / (1 + 0.5 * z)
    r = t * math.exp(-z * z - 1.26551223 + t * (1.00002368 + t * (
        0.37409196 + t * (0.09678418 + t * (-0.18628806 + t * (
            0.27886807 + t * (-1.13520398 + t * (1.48851587 + t * (
                -0.82215223 + t * 0.17087277)))))))))
    return r if x >= 0 else 2 - r


def srm(n_a, n_b):
    """Uji khi-kuadrat pembagian 50/50 yang direncanakan (1 derajat bebas)."""
    e = (n_a + n_b) / 2
    chi2 = (n_a - e) ** 2 / e + (n_b - e) ** 2 / e
    return chi2, erfc(math.sqrt(chi2 / 2))


def compare(x_a, n_a, x_b, n_b):
    p_a, p_b = x_a / n_a, x_b / n_b
    pooled = (x_a + x_b) / (n_a + n_b)
    z = (p_b - p_a) / math.sqrt(pooled * (1 - pooled) * (1 / n_a + 1 / n_b))
    se = math.sqrt(p_a * (1 - p_a) / n_a + p_b * (1 - p_b) / n_b)
    d = p_b - p_a
    return p_a, p_b, z, erfc(abs(z) / math.sqrt(2)), d - 1.96 * se, d + 1.96 * se


def sample_size(p1, p2, z_alpha=1.96, z_beta=0.8416):
    """Pengguna per varian untuk mendeteksi p1 -> p2 (5% dua sisi, daya 80%)."""
    var = p1 * (1 - p1) + p2 * (1 - p2)
    return math.ceil((z_alpha + z_beta) ** 2 * var / (p2 - p1) ** 2)


def show_p(p):
    return "p < 0.0001" if p < 0.0001 else "p = %.4f" % p
```

Kode 14.4 (Lanjutan) *abtest.py: pemeriksaan rasio sampel, selisih dengan selang kepercayaan 95%, uji-z, dan ukuran sampel.*

```
def main(path):
    users = load(path)
    n_a = len(users["A"]["view_menu"])
    n_b = len(users["B"]["view_menu"])
    chi2, p = srm(n_a, n_b)
    print("users: A = %d, B = %d" % (n_a, n_b))
    print("sample ratio check: chi2 = %.2f, %s" % (chi2, show_p(p)))
    if p < 0.001:
        # Pembagian yang rusak membuat semua angka sesudahnya tak tepercaya.
        print("STOP: sample ratio mismatch - find the bug before reading on")
        return
    x_a = len(users["A"]["order_done"])
    x_b = len(users["B"]["order_done"])
    p_a, p_b, z, pv, lo, hi = compare(x_a, n_a, x_b, n_b)
    print("orders:  A = %d (%.2f%%), B = %d (%.2f%%)" % (
        x_a, 100 * p_a, x_b, 100 * p_b))
    print("difference = %+.2f points, 95%% CI [%+.2f, %+.2f]" % (
        100 * (p_b - p_a), 100 * lo, 100 * hi))
    print("relative lift = %+.1f%%, z = %.2f, %s" % (
        100 * (p_b / p_a - 1), z, show_p(pv)))
    print("users per variant needed to detect +2 points from %.1f%%: %d" % (
        100 * p_a, sample_size(p_a, p_a + 0.02)))


if __name__ == "__main__":
    main(sys.argv[1] if len(sys.argv) > 1 else "../data/events.csv")
```

Fungsi `erfc` menghitung fungsi galat komplementer, yang darinya kedua nilai-p diperoleh: untuk z normal baku, nilai-p dua sisi adalah $\operatorname{erfc}\left(|z|/\sqrt{2}\right)$, dan untuk khi-kuadrat berderajat bebas satu adalah $\operatorname{erfc}\left(\sqrt{\chi^2/2}\right)$. Python memiliki `math.erfc`, tetapi pustaka standar Java tidak, jadi kedua program memakai hampiran terbitan yang sama, dengan ketelitian lebih baik daripada $1{,}2 \times 10^{-7}$, sehingga keduanya mencetak digit yang sama.

14.6.2 Program dalam Java

Kode 14.5 adalah pemeriksaan yang sama dalam Java; ia memakai ulang `Funnel.load`.

Kode 14.5 `AbTest.java`: pemeriksaan yang sama dalam Java.

```java
/** Periksa uji A/B: rasio sampel dulu, lalu selisihnya, lalu daya uji. */
public class AbTest {
    /** Fungsi galat komplementer (Numerical Recipes, galat < 1.2e-7). */
    static double erfc(double x) {
        double z = Math.abs(x), t = 1 / (1 + 0.5 * z);
        double r = t * Math.exp(-z * z - 1.26551223 + t * (1.00002368 + t * (
            0.37409196 + t * (0.09678418 + t * (-0.18628806 + t * (
            0.27886807 + t * (-1.13520398 + t * (1.48851587 + t * (
            -0.82215223 + t * 0.17087277)))))))));
        return x >= 0 ? r : 2 - r;
    }

    /** Uji khi-kuadrat pembagian 50/50 yang direncanakan (1 derajat bebas). */
    static double[] srm(int nA, int nB) {
        double e = (nA + nB) / 2.0;
        double chi2 = (nA - e) * (nA - e) / e + (nB - e) * (nB - e) / e;
        return new double[] {chi2, erfc(Math.sqrt(chi2 / 2))};
    }

    static double[] compare(int xA, int nA, int xB, int nB) {
        double pA = (double) xA / nA, pB = (double) xB / nB;
        double pooled = (double) (xA + xB) / (nA + nB);
        double z = (pB - pA)
            / Math.sqrt(pooled * (1 - pooled) * (1.0 / nA + 1.0 / nB));
        double se = Math.sqrt(pA * (1 - pA) / nA + pB * (1 - pB) / nB);
        double d = pB - pA;
        return new double[] {pA, pB, z, erfc(Math.abs(z) / Math.sqrt(2)),
            d - 1.96 * se, d + 1.96 * se};
    }

    /** Pengguna per varian untuk mendeteksi p1 -> p2 (5% dua sisi, daya 80%). */
    static long sampleSize(double p1, double p2) {
        double za = 1.96, zb = 0.8416;
        double var = p1 * (1 - p1) + p2 * (1 - p2);
        double gap = p2 - p1;
        return (long) Math.ceil((za + zb) * (za + zb) * var / (gap * gap));
    }

    static String showP(double p) {
        return p < 0.0001 ? "p < 0.0001" : String.format("p = %.4f", p);
    }
```

Kode 14.5 (Lanjutan) *AbTest.java: pemeriksaan yang sama dalam Java.*

```
    public static void main(String[] args) throws Exception {
        var users = Funnel.load(args.length > 0 ? args[0] : "../data/events.csv");
        int nA = users.get("A").get("view_menu").size();
        int nB = users.get("B").get("view_menu").size();
        double[] s = srm(nA, nB);
        System.out.printf("users: A = %d, B = %d%n", nA, nB);
        System.out.printf("sample ratio check: chi2 = %.2f, %s%n",
            s[0], showP(s[1]));
        if (s[1] < 0.001) {
            // Pembagian yang rusak membuat semua angka sesudahnya tak tepercaya.
            System.out.println(
                "STOP: sample ratio mismatch - find the bug before reading on");
            return;
        }
        int xA = users.get("A").get("order_done").size();
        int xB = users.get("B").get("order_done").size();
        double[] c = compare(xA, nA, xB, nB);
        System.out.printf("orders:  A = %d (%.2f%%), B = %d (%.2f%%)%n",
            xA, 100 * c[0], xB, 100 * c[1]);
        System.out.printf("difference = %+.2f points, 95%% CI [%+.2f, %+.2f]%n",
            100 * (c[1] - c[0]), 100 * c[4], 100 * c[5]);
        System.out.printf("relative lift = %+.1f%%, z = %.2f, %s%n",
            100 * (c[1] / c[0] - 1), c[2], showP(c[3]));
        System.out.printf(
            "users per variant needed to detect +2 points from %.1f%%: %d%n",
            100 * c[0], sampleSize(c[0], c[0] + 0.02));
    }
}
```

14.6.3 Menjalankannya pada log yang sehat

`python abtest.py` (atau `java -cp out AbTest`) mencetak:

```
users: A = 2481, B = 2519
sample ratio check: chi2 = 0.29, p = 0.5910
orders:  A = 545 (21.97%), B = 698 (27.71%)
difference = +5.74 points, 95% CI [+3.35, +8.13]
relative lift = +26.1%, z = 4.70, p < 0.0001
users per variant needed to detect +2 points from 22.0%: 6940
```

Bacalah keluaran dari atas. Pembagiannya 2.481 lawan 2.519, dan pemeriksaan rasio sampel memberi $\chi^2 = 0.29$ dengan $p = 0.59$: ketimpangan sebesar ini sepenuhnya wajar, jadi eksperimen lolos gerbang pertama. Checkout baru berkonversi 27,71% lawan 21,97%, selisih 5,74 poin persentase dengan selang 95% dari +3,35 sampai +8,13 poin, kenaikan relatif 26,1%, dan $z = 4.70$. Seluruh selang berada di atas nol dan jauh di atas batas minimum dua poin dalam rencana. Baris terakhir menunjukkan bahwa mendeteksi perubahan *dua* poin memerlukan 6.940 pengguna per varian; eksperimen ini cukup besar hanya karena efek sebenarnya lebih besar. Seandainya efeknya dua poin, 2.500 pengguna per varian terlalu sedikit, dan "tidak ada perbedaan yang signifikan" akan berarti "kami tidak dapat membedakan", bukan "tidak ada perbedaan".

14.6.4 Menjalankannya pada log yang rusak

Sekarang jalankan program yang sama pada log dengan bug pencatatan: `python abtest.py ../data/events_srm.csv`.

```
users: A = 2472, B = 2155
sample ratio check: chi2 = 21.72, p < 0.0001
STOP: sample ratio mismatch - find the bug before reading on
```

Program berhenti. Dengan 2.472 pengguna di A dan 2.155 di B, $\chi^2 = 21.72$ dan nilai-p di bawah 0,0001: pembagian setimpang ini nyaris tidak pernah terjadi karena kebetulan bila penempatannya benar-benar 50/50. Seandainya program berlanjut, ia akan melaporkan kenaikan bagi B, dan tak seorang pun dapat mengatakan apakah 15% pengguna B yang hilang itu adalah mereka yang akan memesan atau mereka yang akan pergi. Itulah sebabnya pemeriksaan ini didahulukan dan program menolak melanjutkan: menurut Fabijan dan rekan-rekannya, hasil dengan ketidaksesuaian rasio sampel harus dianggap tidak tepercaya sampai penyebabnya ditemukan [5].

14.7 Mengujinya

Kode statistik dapat keliru dengan cara yang tetap mencetak angka yang masuk akal, jadi ia harus diuji terhadap nilai yang diketahui secara independen. **Kode 14.6** memeriksa fungsi galat di dua titik, pembagian yang sempurna, laju yang sama, dan ukuran sampel yang dihitung dengan tangan dengan cara yang sama seperti Soal 4 di Subbab 14.13.

Kode 14.6 `test_abtest.py`: uji fungsi galat, pemeriksaan rasio, uji-z, dan ukuran sampel.

```
"""Pemeriksaan aritmetika A/B; jalankan: python -m unittest test_abtest"""
import math
import unittest

from abtest import compare, erfc, sample_size, srm


class AbTestTest(unittest.TestCase):
    def test_erfc(self):
        self.assertAlmostEqual(erfc(0), 1, places=6)
        self.assertAlmostEqual(erfc(1.96 / math.sqrt(2)), 0.05, places=4)

    def test_srm_and_equal_rates(self):
        self.assertEqual(srm(500, 500)[0], 0)
        self.assertEqual(compare(100, 1000, 100, 1000)[2], 0)

    def test_sample_size(self):
        self.assertEqual(sample_size(0.20, 0.22), 6507)


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

Kode 14.7 melakukan pemeriksaan yang sama dalam Java.

Kode 14.7 `AbTestTest.java`: pemeriksaan yang sama dalam Java.

```java
/** Pemeriksaan aritmetika A/B; jalankan: java AbTestTest */
public class AbTestTest {
    static int failed = 0;

    static void check(boolean ok, String what) {
        System.out.println((ok ? "ok   " : "FAIL ") + what);
        if (!ok) failed++;
    }

    public static void main(String[] args) {
        check(Math.abs(AbTest.erfc(0) - 1) < 1e-6, "erfc(0) = 1");
        check(Math.abs(AbTest.erfc(1.96 / Math.sqrt(2)) - 0.05) < 1e-4,
            "|z| = 1.96 gives p = 0.05");
        check(AbTest.srm(500, 500)[0] == 0, "a perfect split has chi2 = 0");
        double[] c = AbTest.compare(100, 1000, 100, 1000);
        check(c[2] == 0, "equal rates give z = 0");
        check(AbTest.sampleSize(0.20, 0.22) == 6507, "sample size 20% -> 22%");
        System.out.println(failed == 0 ? "all checks passed" : failed + " failed");
        System.exit(failed == 0 ? 0 : 1);
    }
}
```

`python -m unittest test_abtest` melaporkan tiga uji yang lulus, dan versi Java mencetak:

```
ok   erfc(0) = 1
ok   |z| = 1.96 gives p = 0.05
ok   a perfect split has chi2 = 0
ok   equal rates give z = 0
ok   sample size 20% -> 22%
all checks passed
```

Pemeriksaan bahwa $|z| = 1.96$ memberi $p = 0.05$ adalah yang paling berharga. Ia menambatkan kode pada angka yang disepakati setiap tabel statistik, sehingga kesalahan seperti lupa faktor $\sqrt{2}$ langsung tertangkap.

14.8 Membuatnya Lebih Baik

Buat segmen, tetapi rencanakan segmennya. Kenaikan keseluruhan dapat menyembunyikan perbedaan: mungkin B jauh lebih membantu pelanggan lama, yang alamatnya diingat, daripada pelanggan baru. Tambahkan kolom seperti `returning` pada peristiwa dan laporkan corong untuk setiap segmen. Namun tetapkan segmennya di dalam rencana; menjelajahi dua puluh segmen setelah kejadian pasti menemukan satu yang "signifikan" karena kebetulan.

Kurangi variansi dengan data dari sebelum eksperimen. Metode bernama CUPED, yang dikembangkan di Microsoft, menyesuaikan hasil setiap pengguna dengan perilaku pengguna yang sama sebelum eksperimen, yang dapat menyempitkan selang kepercayaan cukup besar sehingga memperpendek eksperimen [9]. Metode ini memerlukan pengguna yang sudah pernah terlihat, jadi lebih membantu pelanggan lama Bu Tini daripada pengunjung baru.

Laporkan metrik pagar di samping metrik utama. Perluas `abtest.py` agar membaca keluhan dan pesanan salah alamat per varian dan mencetaknya dalam format yang sama. Kenaikan pesanan yang disertai kenaikan salah alamat bukanlah kemenangan.

Lihat di mana orang mengetuk. Corong memberi tahu di langkah mana orang pergi, bukan apa yang mereka coba lakukan di sana. Mencatat posisi setiap ketukan di halaman checkout, relatif terhadap lebar dan tinggi halaman, lalu menggambar ketukan itu sebagai peta kerapatan menunjukkan tempat perhatian dan frustrasi berkumpul: gerombolan ketukan pada teks abu-abu ongkos antar, misalnya, berarti pelanggan mengira teks itu tombol yang menjelaskan ongkos.

Peta panas ketukan (*tap heatmap*) semacam ini adalah kerabat murah pelacakan mata. Ia hanya menunjukkan tempat jari mendarat, bukan tempat mata memandang, dan tidak mencatat apa pun tentang orangnya, sehingga posisinya dapat disimpan tanpa alamat atau nomor telepon.

Pantau metrik setelah peluncuran. Eksperimen mengukur dua atau tiga minggu. Setelah B menjadi satu-satunya checkout, tampilkan corongnya pada bagan mingguan. Penurunan perlahan ke arah konversi lama mengisyaratkan efek kebaruan; penurunan mendadak menunjuk ke bug atau perubahan di tempat lain, misalnya versi baru aplikasi pembayaran. Bagan itu mengubah angka tunggal eksperimen menjadi tren yang terus diperhatikan tim.

Pasangkan angka dengan manusia. Setelah B menang, mintalah lima pelanggan yang memakainya untuk memesan sambil berpikir keras, seperti di Bab 7. Eksperimen mengatakan B lebih baik; sesi itu mengatakan apa yang masih sulit, dan menyumbangkan hipotesis bagi eksperimen berikutnya.

14.9 Memakainya Sungguhan

Program-program bab ini bekerja pada log apa pun dalam bentuk `user_id,variant,step`. Untuk memakainya pada situs sungguhan diperlukan tiga bagian.

Kumpulkan peristiwa di server sendiri. Di halaman, kirim permintaan kecil seperti `POST /event` beserta nama langkah setiap kali sebuah langkah tercapai; di server, tambahkan satu baris ke berkas atau tabel basis data dengan pengenal anonim yang disimpan dalam cookie pihak pertama atau penyimpanan lokal. Lakukan ini hanya setelah pengunjung memberi persetujuan analitik, lewat pertanyaan tersendiri yang sederhana, sebagaimana disyaratkan PP 33/2026 [7]. Jangan pernah memasukkan nama, nomor telepon, atau alamat ke dalam peristiwa.

Tempatkan varian secara deterministik. Alih-alih pilihan acak setiap kali halaman dimuat, hitung varian dari pengenal anonim, misalnya dari *hash*-nya modulo 2, sehingga pengunjung yang sama selalu melihat versi yang sama. Simpan varian di setiap peristiwa.

Simpan catatan keputusan. Untuk setiap eksperimen, simpan rencana yang terisi seperti **Tabel 14.2** bersama keluaran `abtest.py` dan keputusan yang diambil. Setelah setahun catatan itu menunjukkan gagasan mana yang berhasil, dan melindungi tim dari mengulang eksperimen yang pernah gagal.

Usaha lain memakai bagian-bagian yang sama. Laundry Pak Made di Bali, misalnya, dapat menguji apakah menampilkan harga total sebelum waktu jemput dipilih, bukan sesudahnya, menaikkan pemesanan yang selesai. Langkah corongnya berubah (`open_form`, `choose_service`, `choose_time`, `confirm`), tetapi program, rencana, dan pemeriksaannya tidak.

14.10 IMK dalam Praktik: Profitina

Profitina, platform AI Visibility (GEO) di app.profitina.com, sendiri merupakan latihan membuat metrik yang terbaca oleh orang yang bukan analis. Dasbor analitik historisnya menampilkan skor visibilitas AI mingguan setiap merek sebagai tren, mengekspornya sebagai CSV, dan memindai ulang secara otomatis dengan irama yang ditentukan tingkat langganan, disertai peringatan bila skor berubah. Pertanyaan desainnya sama dengan Subbab 14.3.1: *tujuan* pemilik merek adalah dikutip oleh mesin jawaban AI, *sinyalnya* adalah sifat-sifat situs yang terukur, dan *metriknya* adalah skor. Karena skor hanyalah metrik, dasbor menjelaskan setiap sinyal di baliknya, agar pemilik tidak mengejar angka alih-alih tujuan, yang berarti hukum Goodhart sedang bekerja.

Kehati-hatian yang sama tampak di tempat-tempat kecil. Penghitung "N baru" pada kanal komunitas yang diikuti hanya menghitung unggahan tingkat atas, bukan balasan, karena penghitung yang melonjak dengan setiap balasan di utas yang ramai akan melebih-lebihkan apa yang baru bagi pembaca. Dasbor menghimpun notifikasi, kanal yang diikuti, peringkat, dan aktivitas dengan satu permintaan, sehingga layar pertama muncul cepat dan setiap butir langsung mengarah ke bagiannya. Dan sejalan dengan UU PDP 27/2022, pengguna dapat mengekspor datanya, mengganti surel dan kata sandi, serta menghapus akunnya sepenuhnya: data di balik metrik tetap milik pengguna.

14.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Mengukur yang mudah alih-alih yang penting.** Tampilan halaman dan lama di situs naik ketika orang tersesat. Mulailah dari tujuan dengan Goals–Signals–Metrics (**Tabel 14.1**).
- **Menghitung peristiwa alih-alih pengguna.** Satu pengunjung yang memuat ulang menu lima kali bukan lima pengunjung. Hitung pengenal pengguna yang berbeda per langkah, seperti `funnel.py`.
- **Membaca proporsi tampilan halaman sebagai proporsi orang.** Angka desktop 60,59% dari StatCounter untuk Indonesia menghitung tampilan halaman di situs yang dilacak [3]. Pakai log Anda sendiri untuk pengguna Anda sendiri.
- **Membandingkan sebelum dan sesudah.** Perubahan pesanan dari minggu ke minggu mencampurkan desain baru dengan gajian, cuaca, dan hari libur. Acaklah dan jalankan kedua versi bersamaan.
- **Melewatkan pemeriksaan rasio sampel.** Pembagian yang timpang berarti ada pengguna yang hilang, dan pengguna yang hilang membiaskan segalanya. Jalankan Persamaan (14.6) lebih dulu, dan berhenti bila $p < 0.001$.
- **Mengintip dan berhenti terlalu dini.** Berhenti ketika p pertama kali turun di bawah 0,05 menggelembungkan positif palsu. Tetapkan ukuran sampel dengan Persamaan (14.5) di muka.
- **Melaporkan nilai-p tanpa besaran efek.** "Signifikan" tidak mengatakan seberapa besar. Laporkan selisih beserta selangnya menurut Persamaan (14.4).
- **Menganggap "tidak signifikan" sama dengan "tidak ada efek".** Dengan pengguna yang terlalu sedikit, efek nyata tak tampak. Bandingkan sampel dengan Persamaan (14.5) sebelum menyimpulkan apa pun.
- **Tanpa metrik pagar.** Kenaikan konversi yang dibeli dengan biaya tersembunyi atau persetujuan yang membingungkan adalah pola gelap, bukan perbaikan.
- **Mengumpulkan data pribadi dalam analitik.** Alamat dan nomor telepon tidak pantas berada dalam peristiwa, dan analitik memerlukan persetujuan tersendiri menurut PP 33/2026 [7].

14.12 Ringkasan Bab

- Analitik mencatat apa yang dilakukan banyak orang; riset dengan segelintir orang menjelaskan mengapa. Desain berbasis data memakai keduanya.
- Peristiwa mencatat satu hal yang terjadi; corong menghitung pengguna berbeda yang mencapai setiap langkah; konversi langkah dan keseluruhan mengikuti Persamaan (14.1) dan Persamaan (14.2), dan langkah terlemah adalah tempat perbaikan paling berbuah.
- Kerangka HEART dengan proses Goals–Signals–Metrics menurunkan metrik dari tujuan (**Tabel 14.1**) [1]; metrik yang naik ketika pengalaman memburuk harus dibuang.

- Pada 2026, 81,7% penduduk Indonesia memakai internet, dengan perbedaan antarwilayah yang besar [2]; proporsi perangkat dari StatCounter menghitung tampilan halaman, bukan orang [3].
- Eksperimen terkendali mengacak pengguna antarversi; selisihnya diuji dengan Persamaan (14.3), besarnya dinyatakan dengan selang pada Persamaan (14.4), dan eksperimennya direncanakan dengan ukuran sampel pada Persamaan (14.5) [4].
- Eksperimen yang tepercaya memeriksa rasio sampel lebih dulu (Persamaan (14.6)) [5], tidak mengintip, berjalan melewati siklus mingguan penuh, dan melindungi metrik pagar dari hukum Goodhart [6].
- Pada data warung simulasi, checkout baru menaikkan pesanan dari 21,97% menjadi 27,71% (selang 95% +3,35 sampai +8,13 poin); pada log dengan bug pencatatan, pemeriksaan rasio sampel ($\chi^2 = 21.72$) menghentikan analisis sebelum sempat menyesatkan.
- Analitik yang dapat dikaitkan dengan seseorang memerlukan persetujuan tersendiri yang tidak digabung, menurut UU PDP 27/2022 dan PP 33/2026 [7]; GA4 memisahkan persetujuan analitik dan iklan sejak 15 Juni 2026 [8].

14.13 Soal Latihan

1. Dalam seminggu 1.200 orang membuka halaman pemesanan sebuah laundry, 780 memilih layanan, 520 memilih waktu, 290 mencapai konfirmasi, dan 270 mengonfirmasi. Hitung setiap konversi langkah dan konversi keseluruhan dengan Persamaan (14.1) dan Persamaan (14.2), lalu katakan langkah mana yang akan Anda garap lebih dulu.
2. Sebuah eksperimen berakhir dengan A: 120 pesanan dari 1.000 pengguna dan B: 150 dari 1.000. Hitung laju gabungan, z dengan Persamaan (14.3), selang 95% selisihnya dengan Persamaan (14.4), lalu nyatakan keputusan Anda.
3. Eksperimen yang direncanakan 50/50 memiliki 5.210 pengguna di A dan 4.790 di B. Hitung χ^2 dengan Persamaan (14.6). Apa langkah Anda berikutnya, dan mengapa?
4. Konversi saat ini 10%. Berapa pengguna per varian yang diperlukan untuk mendeteksi kenaikan menjadi 11% pada taraf 5% dengan daya 80%? Pakai Persamaan (14.5), lalu periksa dengan `sample_size(0.10, 0.11)`. Bila situs itu punya 1.500 pengunjung sehari, berapa lama eksperimen harus berjalan?
5. Untuk kios antrean puskesmas, terapkan proses Goals–Signals–Metrics pada dua kategori HEART pilihan Anda. Kategori mana yang sengaja Anda tinggalkan, dan mengapa?
6. Perluas `funnel.py` dan `Funnel.java` agar, setelah tabel setiap varian, keduanya mencetak langkah dengan konversi langkah terendah. Baris apa yang Anda tambahkan, dan apa yang dicetak program untuk log yang sehat?

14.14 Penyelesaian Soal Latihan

1. Konversi langkah: $780/1200 = 0.650$; $520/780 = 0.667$; $290/520 = 0.558$; $270/290 = 0.931$. Keseluruhan: $270/1200 = 0.225$. Langkah dari memilih waktu ke mencapai konfirmasi kehilangan 44% dan merupakan yang terlemah; garaplah lebih dulu, dan cari tahu *mengapa* lewat beberapa sesi *think-aloud* (barangkali slot waktunya ditampilkan dengan cara yang menyembunyikan slot kosong).

2. Laju gabungan $\bar{p} = 270/2000 = 0.135$. Galat baku di bawah hipotesis nol: $\sqrt{0.135 \cdot 0.865 \cdot (1/1000 + 1/1000)} = \sqrt{0.00023355} = 0.015282$. $z = 0.03/0.015282 = 1.963$, jadi $p \approx 0.0496$, tepat di bawah 0,05. Selang: $\sqrt{0.12 \cdot 0.88/1000 + 0.15 \cdot 0.85/1000} = 0.015268$,

dan $0.03 \pm 1.96 \cdot 0.015268 = 0.03 \pm 0.0299$, dari $+0.0001$ sampai $+0.0599$, yaitu +0,01 sampai +5,99 poin. Hasilnya nyaris tidak signifikan dan selangnya membentang dari hampir nol sampai enam poin. Keputusan yang hati-hati adalah meluncurkan B hanya bila biayanya kecil dan tidak ada masalah metrik pagar, atau lebih baik, menjalankan eksperimen sampai ukuran yang dihitung dengan Persamaan (14.5) untuk efek terkecil yang penting.

3. $e = 10000/2 = 5000$; $\chi^2 = 210^2/5000 + 210^2/5000 = 8.82 + 8.82 = 17.64$; $p = \operatorname{erfc}\left(\sqrt{8.82}\right) \approx 0.000027$, jauh di bawah 0,001. Berhentilah: jangan baca laju konversinya. Carilah apa yang menghilangkan pengguna dari B: galat pada peramban atau ponsel tertentu, pengalihan halaman, penyaring bot, atau perbedaan saat pencatatan dimulai. Perbaiki lalu ulangi; hasil dari pembagian yang timpang tak dapat diperbaiki belakangan.

4. $(1.96 + 0.8416)^2 = 7.849$; $p_1(1-p_1) + p_2(1-p_2) = 0.09 + 0.0979 = 0.1879$; $(p_2 - p_1)^2 = 0.0001$. Jadi $n = 7.849 \cdot 0.1879/0.0001 \approx 14748.3$, dibulatkan ke atas menjadi 14.749 per varian, sebagaimana dikonfirmasi `sample_size(0.10, 0.11)`. Kedua varian bersama memerlukan 29.498 pengguna; dengan 1.500 sehari itu sekitar 20 hari, jadi jalankan tiga minggu penuh.

5. Salah satu jawaban yang mungkin. *Task success* — tujuan: pasien mendapat nomor antrean yang benar dengan cepat; sinyal: nomor dicetak untuk layanan yang kemudian didatangi pasien; metrik: proporsi nomor yang dipakai di loket yang sesuai, dan median detik dari sentuhan pertama sampai nomor keluar. *Happiness* — tujuan: pasien lansia merasa percaya diri memakai kios; sinyal: jawaban atas satu pertanyaan lisan di loket; metrik: proporsi pasien berusia 60+ yang menjawab "mudah". *Engagement* ditinggalkan: pasien yang berlama-lama di kios antrean sedang kesulitan, bukan sedang terlibat.

6. Di Python, di dalam gelung varian dan setelah tabel, tambahkan tiga baris: hitung `rates = [counts[i] / counts[i - 1] for i in range(1, len(STEPS))]`, cari `worst = rates.index(min(rates)) + 1`, lalu cetak `"  weakest step: %s (%.1f%%)" % (STEPS[worst], 100 * rates[worst - 1])`. Di Java, lakukan gelung atas `i` mulai 1, simpan indeks dengan `(double) counts[i] / counts[i - 1]` terkecil, lalu cetak teks yang sama dengan `printf`. Untuk log yang sehat program mencetak `weakest step: pay (56.4%)` untuk A dan `weakest step: add_to_cart (61.8%)` untuk B: begitu checkout diperbaiki, menu menjadi langkah terlemah, dan gagasan Bu Tini tentang foto menjadi eksperimen berikutnya yang wajar.

Daftar Pustaka

[1] K. Rodden, H. Hutchinson, and X. Fu, "Measuring the user experience on a large scale: User-centered metrics for web applications," in *Proc. CHI '10*, Atlanta, GA, USA, 2010, pp. 2395–2398.

[2] ANTARA News, "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] StatCounter Global Stats, "Desktop vs Mobile vs Tablet Market Share Indonesia," August 2026. [Online]. Available: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/indonesia/

[4] R. Kohavi, D. Tang, and Y. Xu, *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge, UK: Cambridge University Press, 2020.

[5] A. Fabijan *et al.*, "Diagnosing sample ratio mismatch in online controlled experiments: A taxonomy and rules of thumb for practitioners," in *Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining (KDD '19)*, Anchorage, AK, USA, 2019, pp. 2156–2164, doi: 10.1145/3292500.3330722.

[6] M. Strathern, "'Improving ratings': Audit in the British university system," *European Review*, vol. 5, no. 3, pp. 305–321, 1997.

[7] Bisnis.com, “Aturan Turunan UU PDP: Persetujuan Data Konsumen Lebih Ketat,” Aug. 29, 2026, tentang Peraturan Pemerintah No. 33 Tahun 2026 sebagai pelaksana UU No. 27 Tahun 2022. [Online]. Available: https://teknologi.bisnis.com/read/20260829/84/1999753/aturan-turunan-uu-pdp-persetujuan-data-konsumen-lebih-ketat

[8] ALM Corp, “Google Analytics 4 and Google Ads consent controls split on June 15, 2026: What ad_storage now controls and what teams must fix,” 2026. [Online]. Available: https://almcorp.com/blog/ga4-google-ads-consent-controls-split-june-2026/

[9] A. Deng, Y. Xu, R. Kohavi, and T. Walker, “Improving the sensitivity of online controlled experiments by utilizing pre-experiment data,” in *Proc. 6th ACM Int. Conf. Web Search and Data Mining (WSDM ’13)*, Rome, Italy, 2013, pp. 123–132.

Bab 15

Antarmuka Masa Depan: Spasial, Wearable, dan Ambient

Antarmuka mulai meninggalkan persegi panjang ponsel dan monitor. Ia melayang di udara di depan pemakai *headset*, duduk di sudut kacamata, mengetuk pergelangan tangan, berpendar dari lampu di dinding, atau bersuara dari pengeras di langit-langit. Bab ini membahas perancangan untuk tempat-tempat baru itu tanpa melupakan orang-orang yang tidak akan pernah memiliki perangkatnya. Di akhir bab Anda akan mampu menjelaskan antarmuka spasial, *wearable*, dan *ambient* serta perangkat yang membawanya pada 2026; menyatakan ukuran target apa pun sebagai sudut pandang dan mengubahnya ke sentimeter pada jarak berapa pun; menjelaskan mengapa masukan pandangan mata memerlukan konfirmasi terpisah dan cara memilih waktu tatap bila konfirmasi itu tidak ada; merancang notifikasi yang dapat dibaca sekilas, berbasis getaran, dan tenang; memeriksa kenyamanan, keselamatan, privasi, dan aksesibilitas antarmuka yang dikenakan di tubuh atau dipasang di ruangan; membangun kalkulator ukuran target dan analisis waktu tatap dalam Java dan Python; serta menerapkan daftar periksa konteks pemakaian sebelum memilih antarmuka baru mana pun.

15.1 Mbah Sumi dan Layar Antrean

Setiap Selasa kedua, Mbah Sumi yang berusia tujuh puluh dua tahun naik becak ke puskesmas untuk memeriksakan tekanan darahnya. Loket pendaftaran memberinya secarik kertas bernomor L-017, lalu ia duduk di bangku panjang di ruang tunggu. Di dinding seberang, kira-kira lima meter jauhnya, sebuah televisi menampilkan nomor yang sedang dilayani; suara rekaman memanggil setiap nomor lewat pengeras suara. Mata Mbah Sumi tak lagi tajam dan pendengarannya lebih buruk lagi. Suara itu bergema, televisinya tampak kecil dari tempat ia duduk, dan sudah dua kali ia melewatkan gilirannya lalu menunggu satu jam lagi.

Perawat puskesmas, Mbak Ratna, melihat hal yang sama menimpa pasien lansia lain setiap minggu. Ia membaca bahwa kacamata terbaru dapat menampilkan teks di sudut lensa, bahwa jam tangan dapat mengetuk pergelangan pemakainya, bahwa *headset* dapat menempatkan layar di mana saja di ruangan, dan ia bertanya-tanya apakah semua itu dapat membantu. Keponakannya, mahasiswa informatika, bersemangat. Mbak Ratna berhati-hati: sebagian besar pasiennya tidak punya jam tangan pintar, tak seorang pun punya kacamata pintar, dan ruang tunggu itu harus berfungsi bagi mereka semua.

Keduanya benar, dan tarik-menarik di antara keduanya menjadi pokok bab ini. Antarmuka masa depan membuka kemungkinan yang tak dapat ditawarkan ponsel: informasi yang muncul di tempat orang memandang, ketukan di pergelangan yang tak didengar orang lain, lampu yang berubah warna ketika giliran sudah dekat. Ia juga membawa cara-cara baru untuk gagal: teks yang terlalu kecil dari kejauhan, tombol yang aktif hanya karena dipandang, notifikasi yang menyela pada saat yang salah, kamera yang tidak disetujui orang di sekitarnya. Dan pelajaran terpenting datang sebelum perangkat apa pun dipilih: perbaikan pertama bagi ruang tunggu Mbah Sumi adalah nomor yang lebih besar di televisi, diukur dari jarak bangku, yang tidak

memakan biaya dan menolong semua orang. Geometri yang sama yang mengukur tombol melayang di *headset* juga mengukur nomor itu.

15.2 Gagasannya dengan Kata-Kata Sederhana

15.2.1 Melampaui persegi panjang

> **DEFINISI.** Antarmuka *spasial* menempatkan konten digital di ruang tiga dimensi di sekitar orang, melalui *headset* atau kacamata, sehingga jendela, tombol, dan objek memiliki posisi dan jarak. Antarmuka *wearable* (dikenakan) dibawa di tubuh, seperti jam tangan, cincin, gelang, atau kacamata, dan dipakai dalam lirikan singkat sementara orang melakukan hal lain. Antarmuka *ambient* (sekitar) tertanam di lingkungan, seperti lampu, bunyi, atau tampilan di dinding, dan menyampaikan informasi di tepi perhatian tanpa memintanya.

Ketiganya berbagi satu sifat yang tidak dimiliki antarmuka desktop dan ponsel: perhatian orang biasanya berada di tempat lain. Pemakai *headset* juga sedang bergerak di dalam ruangan; pemakai jam tangan sedang berjalan, memasak, atau bercakap; pasien di ruang tunggu sedang mengobrol, terkantuk-kantuk, atau menjaga anak. Karena itu antarmukanya harus *mudah ditangkap sekilas* dan *sulit terpicu tanpa sengaja*. Sebagian besar isi bab ini mengalir dari dua tuntutan itu.

15.2.2 Ukuran adalah sudut

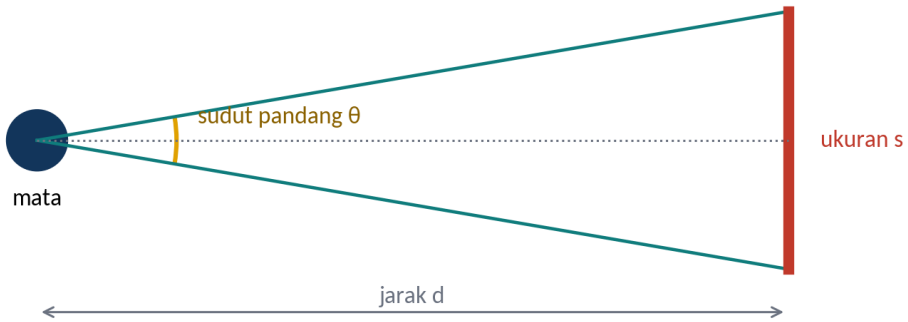
Di ponsel kita menyebut ukuran tombol dalam milimeter atau *point*. Untuk apa pun yang dilihat dari jarak yang berubah-ubah, ukuran yang penting bagi mata adalah *sudut pandang* (*visual angle*): sudut yang ditutup benda itu di mata. **Gambar 15.1** menunjukkan geometrinya. Benda berukuran s pada jarak d menutup sudut

$$\theta = 2\arctan\left(\frac{s}{2d}\right) \tag{15.1}$$

dan sebaliknya, benda harus berukuran

$$s = 2d\tan\left(\frac{\theta}{2}\right) \tag{15.2}$$

untuk menutup sudut θ tertentu pada jarak d.



Gambar 15.1 Sudut pandang θ benda berukuran s pada jarak d: sudut yang sama dapat dihasilkan benda kecil yang dekat atau benda besar yang jauh.

Untuk sudut kecil yang lazim pada antarmuka sehari-hari, $\tan x \approx x$ dengan x dalam radian, dan satu radian kira-kira 57,3 derajat, sehingga hampiran praktis dalam derajat adalah

$$\theta_{\text{deg}} \approx 57.3 \, \frac{s}{d} \tag{15.3}$$

yang menyatakan bahwa benda selebar seperseratus jaraknya menutup sedikit lebih dari setengah derajat. Kuku ibu jari pada jarak sepanjang lengan menutup kira-kira satu sampai dua derajat, penggaris yang mudah dibawa ke mana-mana.

15.2.3 Contoh yang dikerjakan dengan tangan

Nomor antrean di televisi puskesmas tingginya 4 cm, dan bangku Mbah Sumi berjarak 5 m dari dinding. Menurut Persamaan (15.1), $\theta = 2\arctan(4/1000) = 2 \cdot 0.0040$ radian $= 0.0080$ radian, atau sekitar 0,46 derajat; Persamaan (15.3) juga memberi $57.3 \cdot 4/500 = 0.46$ derajat. Itu sekitar 27,5 menit busur.

Cukupkah itu? Huruf pada baris 20/20 bagan mata Snellen menutup 5 menit busur. Orang yang hanya dapat membaca baris 20/100 memerlukan huruf lima kali lebih besar, 25 menit, sehingga bagi pasien dengan penglihatan seperti itu nomor di dinding berada tepat di batas keterbacaan, bukan tempat yang nyaman untuk membaca sambil terkantuk di bangku. Bila puskesmas ingin nomornya menutup 1,5 derajat (90 menit busur, beberapa kali ambang bahkan bagi mata yang lemah) dari jarak 5 m, Persamaan (15.2) memberi $s = 2 \cdot 500 \cdot \tan(0.75^\circ) = 1000 \cdot 0.01309 = 13.1$ cm. Nomor itu harus sekitar tiga kali lebih tinggi daripada sekarang. Pada televisi 43 inci, yang tinggi layarnya sekitar 53 cm, nomor setinggi 13 cm masih menyisakan tempat untuk nama layanan dan dua nomor berikutnya.

15.3 Teori dan Buktinya

15.3.1 Perangkat 2025 dan 2026

Perangkat berganti setiap tahun; prinsipnya tidak. **Tabel 15.1** memuat perangkat yang menentukan arah desain pada 2026, digambarkan dengan kata-kata, bukan dengan logonya.

Tabel 15.1 Perangkat yang membentuk desain antarmuka spasial dan wearable pada 2026, beserta masukan dan keluaran yang harus direncanakan perancang; harga adalah harga peluncuran di Amerika Serikat.

Perangkat (diumumkan)	Jenis	Masukan utama	Keluaran utama	Harga peluncuran
Apple Vision Pro dengan M5 (Okt. 2025) [1]	*Headset*, spasial	Mata, tangan, suara; pengendali dan pena opsional	Layar micro-OLED hingga 120 Hz, audio spasial	mulai US$3,499
Samsung Galaxy XR dengan Android XR (Okt. 2025) [2]	*Headset*, spasial	Mata, tangan, suara dengan Gemini	Layar dengan *passthrough*, audio	US$1,799.99
Meta Ray-Ban Display dengan Neural Band (Sep. 2025) [3]	Kacamata, *wearable*	Gestur gelang EMG, suara, sentuhan	Layar kecil berwarna di satu sisi, audio	US$799
Ray-Ban Meta Audio (Sep. 2026) [4]	Kacamata, *wearable*	Suara, sentuhan	Hanya audio, tanpa kamera	US$349
Kacamata audio Android XR (diumumkan Mei 2026, musim gugur 2026) [5]	Kacamata, *wearable*	Suara dengan Gemini, ketukan pada bingkai	Audio	belum diumumkan
Apple Watch dengan watchOS 26 [6]	Jam tangan, *wearable*	Sentuhan, *crown*, *double tap*, *wrist flick*	Layar, getaran, audio	beragam

Tiga kecenderungan tampak dalam tabel. Masukan bergeser dari menyentuh permukaan ke arah mata, gerakan kecil tangan dan pergelangan, serta suara. Keluaran menyusut menjadi lirikan dan bunyi: layar "di sisi" yang "tidak selalu menyala" [3], atau tanpa layar sama sekali. Dan asisten AI tertanam di kebanyakan perangkat itu, sehingga antarmuka suara di Bab 9 dan antarmuka AI di Bab 13 bertemu dalam perangkat tanpa papan ketik. Perangkat lunak Apple untuk Vision Pro juga bergerak: visionOS 26 hadir bersama model M5, dan visionOS 27, yang diumumkan Juni 2026, menambahkan perluasan notifikasi dengan mata dan Siri yang dapat menjawab pertanyaan tentang apa yang sedang dipandang pemakainya [1], [7].

Tidak semuanya tumbuh. Pada April 2026 MacRumors melaporkan, berdasarkan sumber yang tidak disebut namanya, bahwa pembaruan M5 tidak menghidupkan kembali penjualan Vision Pro, bahwa harga, bobot, dan ketidaknyamanan pada pemakaian lama menjadi keluhan utama, dan bahwa Apple mengalihkan upayanya ke kacamata pintar yang lebih ringan [8]. *Headset* mahal tetap merupakan pasar yang kecil. Bagi perancang di Indonesia pada 2026 akibat praktisnya jelas: ponsel tetap menjadi antarmuka utama, perangkat *wearable* dan suara menjangkau lebih banyak orang daripada *headset*, dan gagasan dari komputasi spasial paling berguna bila memperbaiki layar dan ruangan biasa.

15.3.2 Masukan di ruang: memandang bukan memilih

Di dalam *headset*, penunjuk yang paling alami adalah mata: orang memandang apa yang diinginkannya sebelum meraihnya. Namun mata juga alat untuk *membaca*, dan ia bergerak tiga atau empat kali sedetik entah orangnya menginginkan sesuatu atau tidak. Pada 1990 Robert Jacob menamai masalah yang timbul sebagai *sentuhan Midas* (*Midas touch*): bila semua yang dipandang teraktifkan, seperti segala yang disentuh Raja Midas berubah menjadi emas, orang tidak dapat memandang apa pun tanpa memicunya [9]. Dua jalan keluar klasiknya adalah sinyal kedua yang berarti "ya, yang ini" dan *waktu tatap* (*dwell time*), yaitu lama pandangan harus berdiam pada target sebelum target aktif.

Headset masa kini memilih jalan pertama. Pada Vision Pro dan Galaxy XR, orang memandang target lalu mengonfirmasi dengan cubitan kecil ibu jari dan jari telunjuk; mata menunjuk, tangan memutuskan [1], [2]. Neural Band dari Meta melakukan hal yang sama untuk kacamata dengan

membaca aktivitas listrik otot pergelangan (elektromiografi, EMG), sehingga cubitan, usapan ibu jari, atau putaran pergelangan dapat dikenali dengan tangan tetap santai di samping badan [3]. Waktu tatap tetap penting ketika tangan tidak dapat dipakai: bagi penyandang disabilitas motorik, bagi tangan yang harus steril atau sedang kotor, dan bagi kios umum dengan pelacak mata. Memilih panjangnya adalah tawar-menawar yang diukur di Subbab 15.6.

Penunjukan dengan pandangan juga tidak presisi. Pelacak mata memperkirakan tempat orang memandang dengan galat yang sangat beragam antarorang, antara kedua matanya, serta menurut pencahayaan dan kacamata; Feit dan rekan-rekannya mengukur dua pelacak dengan 80 partisipan dalam kondisi sehari-hari dan menunjukkan cara menurunkan ukuran target dari data semacam itu, bukan dari kasus terbaik versi pabrikan [10]. Itulah sebabnya elemen interaktif di antarmuka spasial lebih besar daripada di ponsel. Dalam panduannya pada 2023 untuk antarmuka spasial, Apple meminta elemen interaktif berukuran setidaknya 60 *point*, dibandingkan batas minimum 44 *point* yang lama dipakai di iOS [11], dan menempatkan konten visionOS pada jarak yang nyaman sehingga *point* itu menutup sudut pandang yang lapang.

15.3.3 Wearable: lirik, ketuk, lanjut

Jam tangan atau kacamata dipakai dalam lirikan yang berlangsung satu atau dua detik. Tiga aturan mengikutinya. *Tampilkan satu hal:* nomor antrean dan berapa orang di depan, bukan menu. *Utamakan getaran dan bunyi pendek daripada teks:* ketukan di pergelangan bersifat pribadi dan tidak perlu dibaca. *Sediakan gestur yang tidak perlu dibidik: double tap* pada Apple Watch (mencubitkan telunjuk dan ibu jari dua kali), yang diperkenalkan pada 2023, dan *wrist flick* yang ditambahkan di watchOS 26 (memutar pergelangan menjauh lalu kembali dengan cepat) menutup atau membisukan notifikasi, panggilan, pengatur waktu, dan alarm tanpa menyentuh layar [6]. Gestur ini adalah fitur aksesibilitas sekaligus kemudahan: ia menolong siapa saja yang tangan satunya sedang sibuk menggendong anak, memegang tongkat, atau menenteng belanjaan.

Pola getaran dapat membawa kosakata kecil bila jumlahnya sedikit dan berbeda jelas. Bagi Mbah Sumi, dua pola sudah cukup: ketukan ganda yang lembut ketika tiga orang lagi di depannya (“bersiaplah”) dan getaran panjang yang kuat ketika nomornya dipanggil (“berangkatlah sekarang”). Lebih dari itu harus dipelajari dan akan tertukar.

Suara adalah masukan lain yang tidak perlu dibidik, dan kacamata audio hampir sepenuhnya bergantung padanya [4], [5]. Suara membawa aturan desainnya sendiri. Keluaran lisan bersifat linear dan cepat berlalu, jadi buatlah singkat dan letakkan kata terpenting di depan: “L-017, giliran Anda, pintu tiga”, bukan kalimat yang baru menyebut nomornya di akhir. Konfirmasikan tindakan dengan mengulang apa yang dipahami sebelum bertindak, seperti asisten laundry di Bab 13 mengulang pesanan. Ingatlah bahwa berbicara itu terdengar orang lain: pasien mungkin tidak ingin menyebut keluhannya keras-keras di ruang tunggu, jadi antarmuka suara harus menyediakan alternatif tanpa suara. Dan rencanakan untuk bahasa yang benar-benar dipakai orang. Di Indonesia satu ruang tunggu dapat mendengar bahasa Indonesia, Jawa, Madura, dan campurannya, sedangkan pengenal suara yang terutama dilatih dengan bahasa Indonesia baku lebih kurang andal mengenali tuturan daerah; ujilah dengan suara para pengguna sesungguhnya sebelum memercayai perintah suara.

15.3.4 Teknologi ambient dan tenang

Pada 1991 Mark Weiser menggambarkan masa depan tempat komputer “menghilang” ke dalam benda sehari-hari, dan pada 1996, bersama John Seely Brown, ia berpendapat bahwa teknologi semacam itu harus *tenang* (*calm*): memberi tahu tanpa menuntut perhatian, memindahkan

informasi antara pusat dan *pinggiran* perhatian sesuai kebutuhan [12], [13]. Tampilan ambient yang baik diperhatikan saat penting dan diabaikan saat tidak, seperti bunyi mesin mobil yang baru didengar pengemudi ketika bunyinya berubah.

Bagi ruang tunggu, hal ini mengisyaratkan lampu di atas setiap pintu layanan yang berpendar lembut dan makin terang ketika antrean bergerak, atau pita berwarna di televisi yang menunjukkan seberapa dekat kelompok nomor berikutnya. Tak seorang pun perlu membaca apa pun, dan orang yang melirik ke atas langsung mengetahui keadaan antrean. Namun desain yang tenang punya syarat: sinyal ambient hanya membawa sedikit informasi, jadi ia harus dipasangkan dengan saluran yang presisi (nomor di layar, suara, atau ketukan di pergelangan) untuk saat bertindak.

15.3.5 Kenyamanan, keselamatan, privasi, dan aksesibilitas

Antarmuka masa depan membawa risiko yang tidak dimiliki layar.

Kenyamanan dan mabuk. Headset dapat menimbulkan *cybersickness*, ketidaknyamanan serupa mabuk perjalanan ketika apa yang dilihat mata dan apa yang dirasakan organ keseimbangan tidak sepakat; paling parah ketika pandangan virtual bergerak sementara tubuh diam. Jagalah dunia virtual tetap stabil, biarkan orang yang menggerakkan pandangan, hindari sesi panjang, dan jangan pernah mewajibkan *headset* untuk tugas yang esensial [14].

Keselamatan. Orang yang memakai *headset* atau membaca kacamata sambil berjalan kurang memperhatikan dunia. Konten yang dibaca sekilas harus tetap pendek, dan antarmuka spasial harus menjaga ruangan nyata tetap terlihat serta memperingatkan adanya rintangan.

Privasi. Kacamata dengan kamera dan mikrofon yang selalu mendengar merekam orang yang tidak menyetujuinya. Itulah sebabnya sebagian pabrikan kini menawarkan kacamata audio tanpa kamera [4], dan mengapa lampu yang terlihat saat merekam itu penting. Di Indonesia, gambar dan suara orang yang dapat dikenali adalah data pribadi menurut UU PDP 27/2022, dan ruang tunggu puskesmas adalah tempat yang persis tidak diharapkan pasien untuk direkam.

Aksesibilitas. Masukan pandangan, tangan, dan suara masing-masing menyingkirkan seseorang: penyandang gangguan mata, orang yang tidak dapat memakai tangannya, orang yang tidak dapat berbicara atau ucapannya sulit dipahami pengenal suara. XR Accessibility User Requirements dari W3C mendaftar kebutuhan seperti alternatif untuk setiap mode masukan, teks keterangan dan audiodeskripsi untuk konten imersif, serta kemampuan menyesuaikan ukuran, warna, dan gerak [15]. Aturan untuk setiap antarmuka masa depan sama dengan aturan untuk antrean: sediakan setidaknya dua jalan masuk dan dua jalan keluar, dan salah satunya bekerja tanpa perangkat baru itu.

15.3.6 Gagasan spasial di layar datar

Gagasan dari komputasi spasial sudah sampai ke ponsel biasa. Pada Juni 2025 Apple memperkenalkan *Liquid Glass*, material tembus pandang untuk kontrol dan navigasi yang membiaskan dan memantulkan konten di belakangnya, terilhami kedalaman visionOS dan diterapkan di seluruh iOS 26 serta sistem Apple lainnya [16]. Permukaan berlapis yang tembus pandang memberi kesan kedalaman dan menjaga konten tetap terlihat di balik kontrol, sekaligus memunculkan pertanyaan lama dari Bab 8 dan Bab 11 dalam bentuk baru: teks di atas permukaan tembus pandang tidak punya latar tetap, sehingga kontrasnya berubah mengikuti apa pun yang digulir di belakangnya. Label yang lolos rasio kontras 4,5:1 WCAG 2.2 di atas halaman putih bisa gagal di atas foto. Perancang yang memakai material semacam ini harus menguji

keterbacaan di atas latar terburuk yang dapat dihasilkan kontennya, dan menghormati pengaturan sistem yang dipakai penyandang low vision, seperti mengurangi transparansi dan menaikkan kontras, yang mengganti kaca dengan warna pejal.

Kehati-hatian yang sama berlaku bagi setiap tren yang berpindah dari *headset* ke ponsel: kedalaman, gerak, dan transparansi dapat menuntun perhatian, tetapi juga dapat mengorbankan keterbacaan dan menimbulkan ketidaknyamanan bagi orang yang peka terhadap gerak. Sediakan versi yang tenang dan pejal sebagai pengaturan, dan pastikan semua hal esensial berfungsi di dalamnya.

15.4 Metode: Daftar Periksa Konteks Pemakaian

Sebelum memilih antarmuka masa depan, gambarkan konteks tempat ia akan dipakai. **Tabel 15.2** adalah daftar periksa beserta jawabannya untuk ruang tunggu puskesmas.

Tabel 15.2 Daftar periksa konteks pemakaian untuk antarmuka masa depan, dijawab untuk ruang tunggu puskesmas.

Pertanyaan	Mengapa penting	Ruang tunggu puskesmas
Seberapa jauh orang dari informasinya?	Ukuran menurut Persamaan (15.2)	Bangku ke dinding: 3–5 m
Apakah tangannya bebas?	Sentuh, cubit, atau suara	Sering tidak: anak, tas, tongkat
Di mana perhatiannya?	Lirikan, ambient, atau terfokus	Pada pendamping, ponsel, terkantuk
Seberapa bising dan terang?	Keterbacaan suara dan layar	Suara ramai, gema; jendela terang
Siapa lagi yang hadir?	Privasi yang ditampilkan dan diucapkan	Pasien lain; nama tidak boleh ditampilkan
Perangkat apa yang dimiliki orang?	Jangkauan dan keadilan	Kebanyakan ponsel sederhana; sedikit jam tangan; tanpa kacamata
Kemampuan apa yang beragam?	Aksesibilitas	Low vision, gangguan dengar, literasi rendah, banyak bahasa
Apa yang terjadi bila gagal?	Cadangan	Petugas memanggil nomor dengan suara di loket
Seberapa sering dipakai, oleh orang yang sama?	Kemudahan dipelajari	Kunjungan bulanan; tidak boleh ada yang perlu dilatih

Daftar periksa ini biasanya menunjukkan bahwa perangkat baru adalah langkah terakhir, bukan yang pertama. Di ruang tunggu ia menunjuk tiga perbaikan menurut urutan jangkauannya: nomor yang lebih besar dan berkontras lebih tinggi di televisi, diukur dengan Persamaan (15.2); lampu berwarna yang tenang di atas setiap pintu; dan, bagi pasien yang memilihnya, pesan SMS atau WhatsApp serta ketukan pergelangan opsional ketika gilirannya dekat. Kacamata datang paling akhir, kalaupun datang.

15.5 Membangun Kalkulator Ukuran Target

Program pertama mengubah Persamaan (15.1) dan Persamaan (15.2) menjadi tabel yang dapat dibaca perancang. Ia mencetak ukuran fisik yang diperlukan untuk sudut pandang pilihan pada empat jarak khas (ponsel di tangan, layar sentuh kios, panel virtual di *headset*, dan televisi antrean di dinding), serta sudut pandang tombol selebar satu sentimeter pada setiap jarak.

Kode 15.1 `visual_angle.py`: ukuran fisik untuk suatu sudut pandang pada empat jarak, dan sudut tombol 1 cm.

```python
"""Seberapa besar target? Sudut pandang <-> ukuran fisik pada suatu jarak."""
import math
import sys


def size_for(angle_deg, distance_cm):
    """Ukuran fisik (cm) yang menutup angle_deg pada jarak distance_cm."""
    return 2 * distance_cm * math.tan(math.radians(angle_deg) / 2)


def angle_of(size_cm, distance_cm):
    """Sudut pandang (derajat) benda selebar size_cm pada jarak distance_cm."""
    return math.degrees(2 * math.atan(size_cm / (2 * distance_cm)))


PLACES = [("phone in the hand", 30), ("kiosk touchscreen", 50),
          ("virtual panel (headset)", 150), ("queue TV on the wall", 300)]


def main(angles):
    print("%-24s %8s" % ("place", "distance") +
          "".join("%9s" % ("%g deg" % a) for a in angles))
    for name, d in PLACES:
        cells = "".join("%6.2f cm" % size_for(a, d) for a in angles)
        print("%-24s %5d cm %s" % (name, d, cells))
    print("\nA 1.0 cm button seen from each place:")
    for name, d in PLACES:
        print("  %-24s %5.2f deg" % (name, angle_of(1.0, d)))


if __name__ == "__main__":
    main([float(a) for a in sys.argv[1:]] or [1.0, 2.0, 3.0])
```

Kode 15.2 adalah kalkulator yang sama dalam Java. Fungsi bantu `g` mencetak 1.0 sebagai “1”, seperti `%g` di Python, sehingga judul kolomnya sama.

Kode 15.2 `VisualAngle.java`: kalkulator yang sama dalam Java.

```java
/** Seberapa besar target? Sudut pandang <-> ukuran fisik pada suatu jarak. */
public class VisualAngle {
    /** Ukuran fisik (cm) yang menutup angleDeg pada jarak distanceCm. */
    static double sizeFor(double angleDeg, double distanceCm) {
        return 2 * distanceCm * Math.tan(Math.toRadians(angleDeg) / 2);
    }

    /** Sudut pandang (derajat) benda selebar sizeCm pada jarak distanceCm. */
    static double angleOf(double sizeCm, double distanceCm) {
        return Math.toDegrees(2 * Math.atan(sizeCm / (2 * distanceCm)));
    }

    static final String[] NAMES = {"phone in the hand", "kiosk touchscreen",
        "virtual panel (headset)", "queue TV on the wall"};
    static final int[] DIST = {30, 50, 150, 300};

    static String g(double x) {             // seperti %g Python: 1.0 -> "1"
        return x == Math.rint(x) ? String.valueOf((long) x) : String.valueOf(x);
    }

    public static void main(String[] args) {
        double[] angles = args.length == 0 ? new double[] {1.0, 2.0, 3.0}
            : new double[args.length];
        for (int i = 0; i < args.length; i++)
            angles[i] = Double.parseDouble(args[i]);
        StringBuilder head = new StringBuilder(
            String.format("%-24s %8s", "place", "distance"));
        for (double a : angles) head.append(String.format("%9s", g(a) + " deg"));
        System.out.println(head);
        for (int p = 0; p < NAMES.length; p++) {
            StringBuilder cells = new StringBuilder();
            for (double a : angles)
                cells.append(String.format("%6.2f cm", sizeFor(a, DIST[p])));
            System.out.printf("%-24s %5d cm %s%n", NAMES[p], DIST[p], cells);
        }
        System.out.println("\nA 1.0 cm button seen from each place:");
        for (int p = 0; p < NAMES.length; p++)
            System.out.printf("  %-24s %5.2f deg%n", NAMES[p],
                angleOf(1.0, DIST[p]));
    }
}
```

Dijalankan tanpa argumen, keduanya mencetak ukuran untuk 1, 2, dan 3 derajat:

```
place                    distance    1 deg    2 deg    3 deg
phone in the hand           30 cm   0.52 cm  1.05 cm  1.57 cm
kiosk touchscreen           50 cm   0.87 cm  1.75 cm  2.62 cm
virtual panel (headset)    150 cm   2.62 cm  5.24 cm  7.86 cm
queue TV on the wall       300 cm   5.24 cm 10.47 cm 15.71 cm

A 1.0 cm button seen from each place:
  phone in the hand         1.91 deg
  kiosk touchscreen         1.15 deg
  virtual panel (headset)   0.38 deg
  queue TV on the wall      0.19 deg
```

Tabel itu memperlihatkan skala masalahnya. Target yang menutup 2 derajat berukuran sekitar 1 cm di ponsel, 1,75 cm di kios, lebih dari 5 cm di panel virtual, dan lebih dari 10 cm di dinding. Bacalah paruh keduanya dari arah sebaliknya: tombol satu sentimeter, yang sangat memadai di tangan dengan 1,91 derajat, menyusut menjadi 0,19 derajat di televisi di seberang ruangan. Sudut juga dapat diberikan lewat baris perintah; `python visual_angle.py 0.5 1.5` mencetak:

```
place                    distance  0.5 deg  1.5 deg
phone in the hand           30 cm   0.26 cm  0.79 cm
kiosk touchscreen           50 cm   0.44 cm  1.31 cm
virtual panel (headset)    150 cm   1.31 cm  3.93 cm
queue TV on the wall       300 cm   2.62 cm  7.85 cm

A 1.0 cm button seen from each place:
  phone in the hand         1.91 deg
  kiosk touchscreen         1.15 deg
  virtual panel (headset)   0.38 deg
  queue TV on the wall      0.19 deg
```

Kolom 1,5 derajat menegaskan contoh di Subbab 15.2.3: pada jarak 3 m nomor memerlukan 7,85 cm, dan pada jarak 5 m bangku Mbah Sumi diperlukan 13,1 cm.

15.6 Memilih Waktu Tatap

15.6.1 Log pandangan simulasi

Misalkan puskesmas mencoba kios yang dikendalikan mata bagi pasien yang tidak dapat memakai tangannya, dengan tombol yang aktif bila dipandang cukup lama. Berapa lama itu cukup? Terlalu singkat, dan pandangan biasa memicu tombol (sentuhan Midas). Terlalu lama, dan orang yang sungguh menginginkan tombol menyerah sebelum tombol itu aktif, atau merasa kiosnya lamban. Jawabannya bergantung pada berapa lama orang memandang sesuatu ketika hanya membaca, dan berapa lama mereka bersedia menatap ketika menginginkan sesuatu.

Untuk mempelajari tawar-menawar ini sebelum uji nyata, **Kode 15.3** menulis log simulasi berisi 1.000 lirikan. Kira-kira satu dari lima adalah pilihan yang *disengaja*: orangnya terus memandang sampai tombol aktif atau kesabarannya habis; kesabaran diambil dari distribusi log-normal dengan median 1.200 ms. Sisanya lirikan biasa saat membaca label, dengan median 260 ms, dekat dengan rerata durasi fiksasi sekitar 225 sampai 250 ms yang dilaporkan untuk membaca dalam hati [17]. Kedua distribusi adalah asumsi yang harus diganti data dari uji rintis sungguhan.

Kode 15.3 `make_gaze.py`: menulis log simulasi dengan seed berisi 1.000 lirikan, dengan tanda mana yang merupakan pilihan disengaja.

```python
"""Tulis log simulasi pandangan Bab 15 (dengan seed, dapat diulang)."""
import math
import random

rng = random.Random(15)
with open("../data/gaze_log.csv", "w", encoding="utf-8") as f:
    f.write("glance,duration_ms,intended\n")
    for g in range(1, 1001):
        if rng.random() < 0.2:
            # Pengguna menginginkan tombol ini dan terus menatap sampai aktif
            # atau sampai kesabarannya habis.
            ms = math.exp(rng.gauss(math.log(1200), 0.35))
            f.write("%d,%d,1\n" % (g, ms))
        else:
            # Melihat biasa: membaca label, memeriksa nomor antrean.
            ms = math.exp(rng.gauss(math.log(260), 0.45))
            f.write("%d,%d,0\n" % (g, ms))
```

15.6.2 Analisis dalam Python dan Java

Untuk waktu tatap T, lirikan biasa yang lebih lama daripada T adalah *aktivasi keliru*, dan pilihan disengaja yang kesabarannya lebih pendek daripada T *terlewat*. Misalkan $F(T)$ adalah aktivasi

keliru per 100 lirikan dan $M(T)$ persentase pilihan disengaja yang terlewat. Satu angka yang hendak diminimalkan memerlukan pertimbangan tentang galat mana yang lebih buruk; bagi kios yang mengeluarkan nomor antrean, mengaktifkan layanan yang salah lebih merugikan daripada harus memandang lagi, jadi program memakai biaya

$$J(T) = 3\,F(T) + M(T) \tag{15.4}$$

yang menghitung satu aktivasi keliru sama beratnya dengan tiga pilihan terlewat. Bobot 3 adalah keputusan desain, bukan hukum alam, dan Soal 5 menanyakan apa yang terjadi bila bobot itu berubah. **Kode 15.4** menilai waktu tatap dari 300 sampai 1.200 ms dan melaporkan waktu dengan biaya terendah.

Kode 15.4 `dwell.py`: aktivasi keliru dan pilihan terlewat untuk setiap waktu tatap, serta waktu tatap dengan biaya terendah.

```
"""Memilih waktu tatap untuk seleksi pandangan: sentuhan Midas lawan menunggu."""
import csv
import sys


def load(path):
    with open(path, encoding="utf-8") as f:
        return [(int(r["duration_ms"]), r["intended"] == "1")
                for r in csv.DictReader(f)]


def evaluate(glances, dwell_ms):
    """Hasil (keliru per 100 lirikan, % disengaja terlewat) satu waktu tatap."""
    false = sum(1 for ms, want in glances if not want and ms >= dwell_ms)
    wanted = [ms for ms, want in glances if want]
    missed = sum(1 for ms in wanted if ms < dwell_ms)
    return 100 * false / len(glances), 100 * missed / len(wanted)


def main(path):
    glances = load(path)
    print("glances: %d, of which intended selections: %d" % (
        len(glances), sum(w for _, w in glances)))
    print("dwell (ms)  false activations /100  intended missed")
    best = None
    for dwell in range(300, 1300, 100):
        false, missed = evaluate(glances, dwell)
        print("%10d  %22.1f  %14.1f%%" % (dwell, false, missed))
        # Biaya sederhana: aktivasi keliru 3 kali lebih merugikan.
        cost = 3 * false + missed
        if best is None or cost < best[0]:
            best = (cost, dwell)
    print("lowest cost (3 x false + missed) at dwell = %d ms" % best[1])


if __name__ == "__main__":
    main(sys.argv[1] if len(sys.argv) > 1 else "../data/gaze_log.csv")
```

Kode 15.5 adalah analisis yang sama dalam Java.

Kode 15.5 `Dwell.java`: analisis yang sama dalam Java.

```java
import java.nio.file.*;
import java.util.*;

/** Memilih waktu tatap untuk seleksi pandangan: sentuhan Midas lawan menunggu. */
public class Dwell {
    record Glance(int ms, boolean intended) {}

    static List<Glance> load(String path) throws Exception {
        List<Glance> out = new ArrayList<>();
        List<String> lines = Files.readAllLines(Path.of(path));
        for (String line : lines.subList(1, lines.size())) {
            String[] f = line.split(",");
            out.add(new Glance(Integer.parseInt(f[1]), f[2].equals("1")));
        }
        return out;
    }

    /** Hasil {keliru per 100 lirikan, % disengaja terlewat} satu waktu tatap. */
    static double[] evaluate(List<Glance> glances, int dwellMs) {
        int falseN = 0, wanted = 0, missed = 0;
        for (Glance g : glances) {
            if (!g.intended() && g.ms() >= dwellMs) falseN++;
            if (g.intended()) { wanted++; if (g.ms() < dwellMs) missed++; }
        }
        return new double[] {100.0 * falseN / glances.size(),
            100.0 * missed / wanted};
    }

    public static void main(String[] args) throws Exception {
        var glances = load(args.length > 0 ? args[0] : "../data/gaze_log.csv");
        long wanted = glances.stream().filter(Glance::intended).count();
        System.out.printf("glances: %d, of which intended selections: %d%n",
            glances.size(), wanted);
        System.out.println("dwell (ms)  false activations /100  intended missed");
        double bestCost = Double.MAX_VALUE;
        int bestDwell = 0;
        for (int dwell = 300; dwell < 1300; dwell += 100) {
            double[] r = evaluate(glances, dwell);
            System.out.printf("%10d  %22.1f  %14.1f%%%n", dwell, r[0], r[1]);
            // Biaya sederhana: aktivasi keliru 3 kali lebih merugikan.
            double cost = 3 * r[0] + r[1];
            if (cost < bestCost) { bestCost = cost; bestDwell = dwell; }
        }
        System.out.printf("lowest cost (3 x false + missed) at dwell = %d ms%n",
            bestDwell);
    }
```

Kode 15.5 (Lanjutan) *Dwell.java: analisis yang sama dalam Java.*

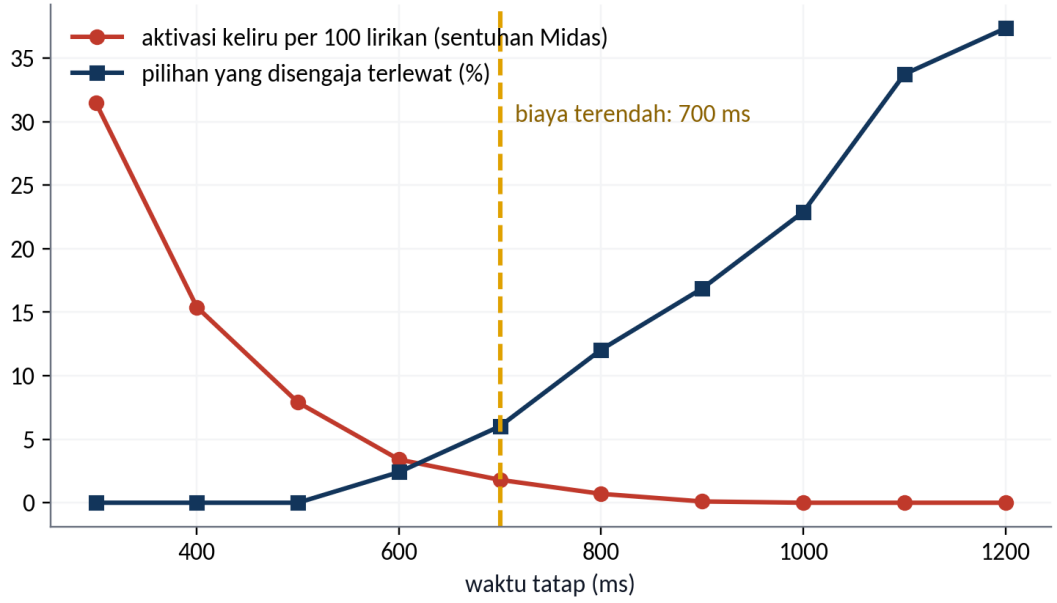
```
}
```

15.6.3 Membaca hasilnya

Kedua program mencetak:

```
glances: 1000, of which intended selections: 166
dwell (ms)  false activations /100  intended missed
       300                    31.5             0.0%
       400                    15.4             0.0%
       500                     7.9             0.0%
       600                     3.4             2.4%
       700                     1.8             6.0%
       800                     0.7            12.0%
       900                     0.1            16.9%
      1000                     0.0            22.9%
      1100                     0.0            33.7%
      1200                     0.0            37.3%
lowest cost (3 x false + missed) at dwell = 700 ms
```

Gambar 15.2 menggambar kedua kolom itu. Pada 300 ms hampir sepertiga dari semua lirikan akan memicu tombol: kios itu tak dapat dipakai. Aktivasi keliru turun cepat seiring bertambahnya waktu tatap, di bawah 2 per 100 pada 700 ms dan nol pada 1.000 ms, sedangkan pilihan terlewat mulai naik sejak 600 ms dan melewati 20% pada 1.000 ms. Biaya menurut Persamaan (15.4) terendah pada 700 ms, ketika 1,8 dari 100 lirikan masih memicu tombol secara keliru dan 6% pilihan disengaja terlewat.



Gambar 15.2 Tawar-menawar waktu tatap pada log pandangan simulasi: aktivasi keliru turun dan pilihan terlewat naik seiring bertambahnya waktu tatap; biaya terendah menurut Persamaan (15.4) ada pada 700 ms.

Tak satu pun galat hilang pada waktu tatap berapa pun, dan itulah pelajaran sesungguhnya dari kurva ini. Antarmuka yang hanya mengandalkan waktu tatap selalu menukar satu galat dengan galat lainnya; satu-satunya jalan keluar adalah sinyal kedua. Itulah sebabnya *headset* memasangkan pandangan dengan cubitan, dan mengapa kios umum yang dikendalikan mata harus menambahkan langkah konfirmasi untuk apa pun yang tak dapat dibatalkan: “Anda memilih Lansia. Pandang YA selama satu detik untuk mencetak nomor Anda.”

15.7 Mengujinya

Kode 15.6 memeriksa bahwa Persamaan (15.1) dan Persamaan (15.2) saling invers, bahwa satu derajat pada 3 m kira-kira 5,24 cm, dan bahwa analisis waktu tatap berperilaku seperti dituntut

tawar-menawarnya: waktu tatap yang lebih lama tidak pernah memberi lebih banyak aktivasi keliru atau lebih sedikit pilihan terlewat.

Kode 15.6 `test_spatial.py`: uji geometri dan arah tawar-menawar waktu tatap.

```
"""Pemeriksaan fungsi bantu spasial; jalankan: python -m unittest test_spatial"""
import unittest

from dwell import evaluate, load
from visual_angle import angle_of, size_for


class SpatialTest(unittest.TestCase):
    def test_round_trip(self):
        self.assertAlmostEqual(angle_of(size_for(2.0, 50), 50), 2.0)
        self.assertAlmostEqual(size_for(1.0, 300), 5.236, places=3)

    def test_dwell_trade_off(self):
        glances = load("../data/gaze_log.csv")
        low, high = evaluate(glances, 300), evaluate(glances, 1000)
        self.assertGreater(low[0], high[0])
        self.assertLessEqual(low[1], high[1])


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

Kode 15.7 adalah versi Java-nya.

Kode 15.7 `SpatialTest.java`: pemeriksaan yang sama dalam Java.

```
/** Pemeriksaan fungsi bantu spasial; jalankan: java SpatialTest */
public class SpatialTest {
    static int failed = 0;

    static void check(boolean ok, String what) {
        System.out.println((ok ? "ok   " : "FAIL ") + what);
        if (!ok) failed++;
    }

    public static void main(String[] args) throws Exception {
        double s = VisualAngle.sizeFor(2.0, 50);
        check(Math.abs(VisualAngle.angleOf(s, 50) - 2.0) < 1e-9, "round trip");
        check(Math.abs(VisualAngle.sizeFor(1.0, 300) - 5.236) < 0.001,
            "1 degree at 3 m is about 5.24 cm");
        var glances = Dwell.load("../data/gaze_log.csv");
        double[] low = Dwell.evaluate(glances, 300);
        double[] high = Dwell.evaluate(glances, 1000);
        check(low[0] > high[0], "longer dwell, fewer false activations");
        check(low[1] <= high[1], "longer dwell, more missed selections");
        System.out.println(failed == 0 ? "all checks passed" : failed + " failed");
        System.exit(failed == 0 ? 0 : 1);
    }
}
```

`python -m unittest test_spatial` melaporkan dua uji yang lulus; versi Java mencetak:

```
ok   round trip
ok   1 degree at 3 m is about 5.24 cm
ok   longer dwell, fewer false activations
ok   longer dwell, more missed selections
all checks passed
```

Kelompok uji kedua memeriksa sebuah *sifat*, bukan sebuah angka: apa pun datanya, menaikkan waktu tatap tidak dapat menambah aktivasi keliru. Uji sifat seperti ini tetap sahih ketika log simulasi diganti data nyata, justru saat uji itu paling dibutuhkan.

15.8 Membuatnya Lebih Baik

Konfirmasikan, jangan hanya menatap. Untuk kios yang dikendalikan mata, pakai waktu tatap 700 ms untuk *memilih* dan langkah kedua yang disengaja untuk *memastikan*: target YA tersendiri, pola kedipan yang dipilih pengguna, atau sakelar bagi orang yang dapat menekannya. Galat yang tersisa kemudian memerlukan dua kekeliruan berturut-turut.

Sesuaikan waktu tatap dengan orangnya. Kecepatan membaca dan kesabaran berbeda-beda. Kios dapat memulai dengan 700 ms lalu memperpendek waktu tatap bagi orang yang memilih dengan cepat dan tidak pernah membatalkan, atau memperpanjangnya setelah aktivasi keliru yang dibatalkan orang itu.

Beri umpan balik selama menatap. Cincin yang terisi di sekeliling target selama waktu tatap menunjukkan bahwa kios telah melihat pandangan itu dan memberi kesempatan orang untuk mengalihkan pandangan tepat waktu. Pengatur waktu yang tak terlihat berubah menjadi janji yang terlihat.

Ukur target dari kasus terburuk yang wajar. Pakai sudut dari **Kode 15.1** dengan jarak kursi terjauh, bukan yang terdekat, dan dengan penglihatan para pasien, bukan para perancang. Untuk pandangan, tambahkan galat pelacak yang terukur bagi orang-orang yang akan memakainya, seperti disarankan Feit dan rekan-rekannya [10].

Rancang saluran ambient bersama saluran yang presisi. Lampu yang makin terang ketika antrean mendekat memberi tahu pasien untuk memperhatikan; nomor di layar, suara, dan pesan opsional di ponsel memberi tahu dengan tepat kapan harus maju. Ujilah bahwa keduanya sepakat: lampu yang menunjukkan “sebentar lagi” sementara layar menunjukkan sepuluh orang di depan mengajari orang untuk mengabaikan lampu.

15.9 Memakainya Sungguhan

Mbak Ratna memakai alat-alat bab ini dalam tiga langkah. Pertama, ia mengukur jarak dari bangku terjauh ke televisi (6 m) dan menjalankan `python visual_angle.py 1.5`; hasilnya, sekitar 15,7 cm, menjadi spesifikasi nomor di layar, ditampilkan dengan huruf tebal berkontras tinggi seperti dituntut Bab 11. Kedua, ia mengusulkan strip berwarna di atas setiap pintu layanan, digerakkan oleh data antrean yang sama, yang berubah dari abu-abu ke kuning tua ketika tersisa lima nomor dan ke hijau toska ketika nomornya dipanggil. Ketiga, pasien yang mendaftarkan nomor telepon dan menyetujuinya dalam kalimat tersendiri yang sederhana menerima SMS ketika tiga nomor lagi di depannya; mereka yang memakai jam tangan yang menampilkan notifikasi ponsel merasakannya di pergelangan.

Baru setelah itu ia menimbang pilihan berbasis mata atau kacamata, dan hanya bagi pasien yang memintanya. Bagi pasien yang tidak dapat memakai tangannya, kios pandangan dengan waktu tatap 700 ms dan langkah konfirmasi akan menjadi cara alternatif mengambil nomor, di samping kios sentuh di Bab 16 dan loket pendaftaran.

Biayanya layak dituliskan, karena biayalah yang menentukan apa yang benar-benar dapat dilakukan puskesmas. Nomor yang lebih besar di televisi yang sudah ada hanya memerlukan

perubahan pengaturan huruf di perangkat lunak antrean. Strip lampu berwarna di atas setiap pintu tidak jauh lebih mahal daripada lampunya sendiri dan sebuah pengendali kecil yang terhubung ke sistem antrean. Pengingat SMS memerlukan beberapa ratus rupiah per pesan dan sedikit pemrograman. Kios pandangan memerlukan pelacak mata, layar yang kokoh, dan pengujian cermat bersama pasien yang membutuhkannya. Urutan daftar itu, dari yang termurah ke yang termahal, juga urutan banyaknya pasien yang ditolong setiap langkah, dan dalam pekerjaan aksesibilitas hal itu jarang merupakan kebetulan.

Metode yang sama melayani tempat-tempat lain. Di terminal bus, papan keberangkatan diukur dari ujung aula yang terjauh; di pabrik, lampu peringatan tetap tenang sampai saat penting lalu mustahil terlewat; di pasar, papan harga kios diukur untuk lorong, bukan untuk meja jualan.

15.10 IMK dalam Praktik: Profitina

Profitina adalah aplikasi web, bukan aplikasi *headset*, tetapi pergeseran ke kacamata dan suara langsung mengubah tujuannya. Profitina mengukur seberapa siap situs sebuah merek dikutip oleh mesin jawaban AI. Di ponsel atau desktop, mesin jawaban dapat menampilkan beberapa sumber dan tautan. Melalui kacamata audio atau asisten suara, jenis yang diumumkan Meta dan Google untuk 2026 [4], [5], orang mendengar *satu* jawaban lisan, barangkali menyebut satu sumber. Menjadi sumber yang dikutip jawaban itu makin berharga ketika tidak ada halaman tautan untuk digulir. Itulah argumen Generative Engine Optimization dalam satu kalimat, dan itulah sebabnya konten yang mudah diekstraksi, terstruktur dengan schema.org, dan kredibel secara teknis penting bahkan bagi sebuah warung.

Antarmuka Profitina sendiri mengikuti pelajaran ambient bab ini dengan cara-cara kecil. Komunitas *real-time*-nya memakai Server-Sent Events untuk menjaga penghitung “N baru” tetap mutakhir tanpa pengguna memuat ulang, sinyal pinggiran yang tenang alih-alih interupsi, dan kokpit dasbor menghimpun notifikasi di satu tempat, masing-masing mengarah ke bagiannya, alih-alih mendorong setiap notifikasi sebagai peringatan. Dan karena AI di dalam Profitina berjalan lokal di server Profitina sendiri, antarmuka suara atau kacamata di masa depan tidak perlu mengirim data pengguna ke layanan AI luar.

15.11 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Mengukur dalam piksel atau sentimeter tanpa jarak.** Ukuran tak bermakna sebelum jarak pandang diketahui. Tetapkan target sebagai sudut pandang dan ubah dengan Persamaan (15.2) untuk jarak realistis yang terjauh.
- **Merancang dari kursi terdekat.** Perancang berdiri dekat layar; pasien duduk di belakang. Ukurlah dari tempat penggunanya berada.
- **Membiarkan pandangan mengaktifkan.** Pandangan tanpa sinyal kedua terkena sentuhan Midas [9]. Konfirmasikan dengan cubitan, sakelar, target tersendiri, atau waktu tatap yang dipilih dengan baik (**Gambar 15.2**).
- **Terlalu banyak pola getaran.** Dua atau tiga pola yang jelas berbeda dapat dipelajari; delapan tidak. Pasangkan setiap pola dengan satu makna.
- **Sinyal ambient yang membawa informasi presisi.** Lampu tidak dapat mengatakan “nomor Anda L-017”. Pakai sinyal ambient untuk “perhatikan” dan saluran presisi untuk “maju sekarang”.

- **Mewajibkan perangkat baru.** Bila ruang tunggu hanya berfungsi dengan jam tangan pintar, sebagian besar pasien tersingkir. Setiap antarmuka masa depan memerlukan jalur yang bekerja tanpanya.
- **Melupakan orang di sekitar.** Kamera dan mikrofon di kacamata merekam orang yang tidak menyetujuinya. Utamakan desain yang tidak memerlukannya dan buat perekaman terlihat.
- **Mengabaikan kenyamanan.** Sesi *headset* yang panjang dan pandangan virtual yang bergerak sendiri menimbulkan mabuk [14]. Buat sesi tetap singkat dan opsional.

15.12 Ringkasan Bab

- Antarmuka spasial menempatkan konten di ruang tiga dimensi, *wearable* dipakai dalam lirikan di tubuh, dan antarmuka ambient memberi tahu dari lingkungan; ketiganya harus mudah ditangkap sekilas dan sulit terpicu tanpa sengaja.
- Ukuran yang penting bagi mata adalah sudut pandang (Persamaan (15.1)); ukuran fisik yang diperlukan untuk suatu sudut pada suatu jarak mengikuti Persamaan (15.2), dan Persamaan (15.3) adalah hampiran cepatnya.
- Pada 2026 *headset* memakai mata, tangan, dan suara [1], [2]; kacamata bergerak ke arah gestur gelang, suara, dan audio [3], [4], [5]; jam tangan menambahkan gestur tanpa sentuh seperti *double tap* dan *wrist flick* [6] (**Tabel 15.1**).
- Memandang bukan memilih: masukan pandangan memerlukan sinyal kedua atau waktu tatap untuk menghindari sentuhan Midas [9], dan target harus memberi ruang bagi galat pelacak yang beragam antarorang [10]; antarmuka spasial memakai target yang lebih besar, seperti batas minimum 60 *point* dari Apple [11].
- Teknologi yang tenang memindahkan informasi antara pinggiran dan pusat perhatian [12], [13]; sinyal ambient sebaiknya mengatakan “perhatikan” dan menyerahkan informasi presisi kepada saluran lain.
- Kenyamanan, keselamatan, privasi orang di sekitar, dan aksesibilitas memerlukan desain yang eksplisit [14], [15]; setiap antarmuka masa depan memerlukan alternatif yang bekerja tanpanya.
- Kalkulator ukuran target menunjukkan bahwa tombol 1 cm menutup 1,91 derajat di tangan dan 0,19 derajat di dinding pada jarak 3 m; analisis waktu tatap atas log pandangan simulasi menemukan biaya terendah (Persamaan (15.4)) pada 700 ms, tanpa galat mana pun lenyap pada waktu tatap berapa pun.
- Daftar periksa konteks pemakaian (**Tabel 15.2**) biasanya menunjukkan bahwa perbaikan termurah bagi semua orang datang lebih dulu, dan perangkat baru paling akhir.

15.13 Soal Latihan

1. Sebuah apotek menampilkan nomor pelanggan berikutnya di layar yang berjarak 4 m dari kursi. Tinggi nomornya 6 cm. Hitung sudut pandangnya dengan Persamaan (15.1) dan Persamaan (15.3), dalam derajat dan menit busur, serta tinggi yang diperlukan untuk 1,5 derajat dengan Persamaan (15.2).
2. Jalankan `python visual_angle.py 0.5 1.5` dan pakai keluarannya untuk menjawab: pada jarak mana dari keempat jarak itu tombol 1 cm lebih kecil daripada 0,5 derajat? Apa yang akan Anda ubah di setiap tempat itu?

3. Dengan keluaran `dwell.py`, temukan waktu tatap berbiaya terendah bila satu aktivasi keliru hanya seberat satu pilihan terlewat, $J(T) = F(T) + M(T)$. Jelaskan dalam satu kalimat mengapa jawabannya bergeser ke arah itu.
4. Untuk setiap situasi berikut, pilih spasial, *wearable*, ambient, atau tidak satu pun, dan beri alasan dengan baris-baris **Tabel 15.2**: (a) memberi tahu pengemudi motor sebuah laundry bahwa ada penjemputan baru; (b) menunjukkan kepada juru masak di dapur warung pesanan mana yang berikutnya; (c) membantu teknisi membaca diagram kabel mesin sementara kedua tangannya memegang perkakas.
5. Ubah `dwell.py` dan `Dwell.java` agar bobot aktivasi keliru dibaca dari baris perintah (bawaan 3). Baris mana yang berubah? Apa yang dilaporkan program dengan bobot 10, dan mengapa?
6. Rancang dua pola getaran dan satu sinyal ambient untuk antrean Mbah Sumi, lalu tulis teks SMS yang diterimanya. Periksa masing-masing terhadap Kesalahan yang Sering Terjadi di Subbab 15.11.

15.14 Penyelesaian Soal Latihan

1. $\theta = 2\arctan(6/800) = 2 \cdot 0.0075 = 0.0150$ radian $= 0.86$ derajat, sekitar 51,6 menit busur; Persamaan (15.3) memberi $57.3 \cdot 6/400 = 0.86$ derajat. Untuk 1,5 derajat, $s = 2 \cdot 400 \cdot \tan(0.75°) = 800 \cdot 0.01309 = 10.5$ cm. Nomor itu semestinya hampir dua kali lebih tinggi.

2. Tombol 1 cm menutup 1,91 derajat pada 30 cm, 1,15 pada 50 cm, 0,38 pada 150 cm, dan 0,19 pada 300 cm, jadi ia di bawah 0,5 derajat di panel virtual dan di dinding. Di panel virtual, perbesar target interaktif (untuk 1,5 derajat pada 150 cm tabel memberi 3,93 cm) atau dekatkan panelnya; di dinding, tidak ada yang semestinya berupa *tombol* sama sekali, dan teks diukur dari kolom 1,5 derajat, 7,85 cm pada 3 m.

3. Dengan $J = F + M$: 300 ms memberi $31.5 + 0 = 31.5$; 400 memberi 15,4; 500 memberi 7,9; 600 memberi $3.4 + 2.4 = 5.8$; 700 memberi $1.8 + 6.0 = 7.8$; 800 memberi $0.7 + 12.0 = 12.7$. Biaya terendah ada pada 600 ms. Bila aktivasi keliru dianggap kurang merugikan, waktu tatap terbaik bergeser lebih pendek, menerima lebih banyak aktivasi tak sengaja demi lebih sedikit pilihan terlewat.

4. (a) *Wearable* atau ponsel: tangan dan mata pengemudi tertuju ke jalan, jadi tanpa layar; pola getaran pendek di jam tangan atau bunyi di *headset* helm mengatakan "penjemputan baru", dan rinciannya baru dibaca ketika berhenti (baris tangan, perhatian, dan keselamatan). (b) Ambient ditambah layar: tampilan besar di dinding dapur yang diukur dari kompor dengan Persamaan (15.2), dengan lampu berwarna untuk "ada pesanan menunggu" (jarak, tangan sibuk, bising). (c) Spasial, bila anggaran memungkinkan: kacamata atau *headset* yang menampilkan diagram di samping mesin, dikendalikan suara karena tangannya sibuk; diagram kertas tetap menjadi cadangan (tangan, kegagalan).

5. Di Python, baca `weight = float(sys.argv[2]) if len(sys.argv) > 2 else 3.0`, teruskan ke `main`, lalu ganti `3 * false` dengan `weight * false` dan teks "3 x false" dengan "%g x false" % weight. Di Java, urai `args[1]` dengan cara yang sama dan pakai di `cost` serta di `printf` terakhir. Dengan bobot 10, biayanya $10F + M$: 700 ms memberi $18 + 6 = 24$, 800 ms memberi $7 + 12 = 19$, 900 ms memberi $1 + 16.9 = 17.9$, 1.000 ms memberi $0 + 22.9 = 22.9$, jadi biaya terendah bergeser ke 900 ms: makin ditakuti aktivasi keliru, makin panjang waktu tatapnya.

6. Salah satu rancangan yang mungkin. Getaran: ketukan ganda yang lembut ketika tiga nomor lagi di depan ("bersiaplah"); satu getaran panjang yang kuat ketika nomornya dipanggil ("maju sekarang"). Ambient: strip cahaya di atas pintu layanan lansia yang berubah kuning tua ketika

tersisa lima nomor dan hijau toska ketika nomornya dipanggil. SMS: "Puskesmas Sehat Selalu: nomor L-017, 3 orang lagi di depan Anda. Mohon tetap di dekat pintu Lansia." Pemeriksaan: hanya dua pola getaran; sinyal ambient mengatakan "perhatikan", dan nomor yang presisi datang lewat SMS dan layar; SMS tidak memerlukan perangkat baru; isinya tanpa nama atau diagnosis, jadi tidak membuka apa pun kepada orang lain yang membaca ponsel itu; dan pasien yang tidak mendaftar tetap punya layar, suara, dan petugas.

Daftar Pustaka

[1] Apple Inc., "Apple Vision Pro upgraded with the M5 chip and Dual Knit Band," press release, Oct. 15, 2025. [Online]. Available: https://www.apple.com/newsroom/2025/10/apple-vision-pro-upgraded-with-the-m5-chip-and-dual-knit-band/

[2] Samsung Electronics, "Introducing Galaxy XR: Opening new worlds," Oct. 2025. [Online]. Available: https://news.samsung.com/us/introducing-galaxy-xr-opening-new-worlds/

[3] Meta, "Meta Ray-Ban Display: AI glasses with an EMG wristband," Sep. 17, 2025. [Online]. Available: https://about.fb.com/news/2025/09/meta-ray-ban-display-ai-glasses-emg-wristband/

[4] Engadget, "Everything announced at Meta Connect 2026," Sep. 2026. [Online]. Available: https://www.engadget.com/2267230/everything-announced-at-meta-connect-2026/

[5] Google, "Intelligent eyewear with Gemini is coming this fall," *The Keyword*, May 19, 2026. [Online]. Available: https://blog.google/products-and-platforms/platforms/android/android-xr-io-2026/

[6] 9to5Mac, "watchOS 26 added a new Apple Watch trick that I've been loving," Feb. 4, 2026. [Online]. Available: https://9to5mac.com/2026/02/04/watchos-26-added-a-new-apple-watch-trick-that-ive-been-loving/

[7] 9to5Mac, "visionOS 27 announced with new features for Vision Pro," Jun. 8, 2026. [Online]. Available: https://9to5mac.com/2026/06/08/visionos-27-announced-with-new-features-for-vision-pro/

[8] J. Clover, "Apple has given up on the Vision Pro after M5 refresh flop," MacRumors, Apr. 29, 2026. [Online]. Available: https://www.macrumors.com/2026/04/29/apple-vision-pro-m5-flop/

[9] R. J. K. Jacob, "What you look at is what you get: Eye movement-based interaction techniques," in *Proc. CHI '90*, Seattle, WA, USA, 1990, pp. 11–18.

[10] A. M. Feit *et al.*, "Toward everyday gaze input: Accuracy and precision of eye tracking and implications for design," in *Proc. CHI '17*, Denver, CO, USA, 2017, pp. 1118–1130, doi: 10.1145/3025453.3025599.

[11] Apple Inc., "Design for spatial user interfaces," WWDC23 session 10076, Jun. 2023. [Online]. Available: https://developer.apple.com/videos/play/wwdc2023/10076/

[12] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 94–104, 1991.

[13] M. Weiser and J. S. Brown, "Designing calm technology," *PowerGrid Journal*, vol. 1, no. 1, 1996.

[14] J. J. LaViola Jr., E. Kruijff, R. P. McMahan, D. A. Bowman, and I. Poupyrev, *3D User Interfaces: Theory and Practice*, 2nd ed. Boston, MA, USA: Addison-Wesley, 2017.

[15] W3C, "XR Accessibility User Requirements," W3C Group Note, Aug. 25, 2021. [Online]. Available: https://www.w3.org/TR/xaur/

[16] Apple Inc., "Apple introduces a delightful and elegant new software design," press release, Jun. 9, 2025. [Online]. Available: https://www.apple.com/newsroom/2025/06/apple-introduces-a-delightful-and-elegant-new-software-design/

[17] K. Rayner, "Eye movements in reading and information processing: 20 years of research," *Psychological Bulletin*, vol. 124, no. 3, pp. 372–422, 1998.

Bab 16

Proyek Akhir: Merancang dan Mengevaluasi Sistem Interaktif Utuh

Lima belas bab masing-masing telah mengajarkan satu bagian interaksi manusia dan komputer. Bab ini menyatukan semuanya pada satu masalah nyata, dari kunjungan pertama kepada orang-orang yang mengalaminya sampai bukti bahwa sebuah desain menolong mereka. Masalahnya cukup kecil untuk dituntaskan dan cukup serius untuk berarti: kios layan mandiri tempat pasien puskesmas mengambil nomor antrean. Di akhir bab Anda akan mampu menjalankan proyek desain yang utuh dan iteratif: mempelajari konteks dan orang-orangnya; mengubah temuan menjadi kebutuhan, persona, dan skenario; menerapkan hukum faktor manusia, heuristik desain, kriteria sukses aksesibilitas, dan aturan etika sebagai batasan desain; membuat prototipe di atas kertas lalu sebagai aplikasi web yang berjalan dalam Python dan Java; mengujinya bersama pengguna, termasuk lansia, dan mengukurnya dengan penyelesaian tugas, System Usability Scale, UMUX-Lite, dan pemeriksaan kontras warna; mengubah temuan menjadi daftar perubahan berprioritas; serta menyajikan keputusan desain sehingga orang lain dapat memeriksanya. Setiap langkah menyebut bab buku ini yang metodenya diterapkan.

16.1 Kios untuk Puskesmas Sehat Selalu

Puskesmas Sehat Selalu, di pinggiran sebuah kota di Jawa Timur, melayani sekitar dua ratus pasien pada pagi yang sibuk. Pendaftaran berlangsung di satu loket. Pasien datang sejak pukul tujuh, menunggu dalam antrean yang mengular sampai ke luar pintu, menyebutkan nama dan keluhannya, lalu menerima secarik kertas bernomor untuk salah satu dari empat layanan: poli umum, kesehatan ibu dan anak (KIA), poli gigi, atau poli lansia. Bab 15 telah mempertemukan kita dengan salah seorang dari mereka, Mbah Sumi, yang dua kali melewatkan gilirannya karena tak dapat membaca nomor di televisi.

Kepala puskesmas ingin memperpendek antrean di loket. Gagasannya adalah kios layar sentuh di pintu masuk, tempat pasien yang sudah tahu layanan tujuannya mengambil nomor sendiri; loket lalu hanya melayani pasien baru dan pertanyaan. Ia meminta tim tiga mahasiswa informatika merancangnya, dengan Mbak Ratna, sang perawat, sebagai pemandu. Para mahasiswa punya satu semester, tanpa anggaran selain sebuah tablet sumbangan, dan satu aturan dari kepala puskesmas: *tidak boleh ada yang lebih dirugikan daripada hari ini.*

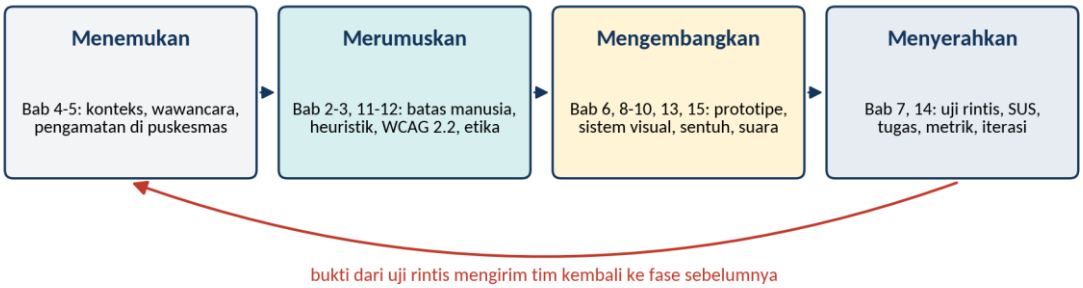
Aturan itulah pokok sesungguhnya dari proyek akhir ini. Kios yang cepat bagi anak muda berponsel tetapi membingungkan Mbah Sumi akan memperpendek antrean sekaligus membuat puskesmas lebih buruk bagi orang yang paling membutuhkannya. Karena itu para mahasiswa harus melakukan apa yang diajarkan buku ini: mencari tahu siapa penggunanya dan apa kebutuhannya, merancang bagi yang paling tidak beruntung di antara mereka, membangun sesuatu yang berfungsi, dan membuktikan dengan bukti bahwa ia menolong.

16.2 Gagasannya dengan Kata-Kata Sederhana

16.2.1 Sistem interaktif yang utuh

Sistem interaktif yang *utuh* lebih dari sekadar layar yang berfungsi. Ia adalah layanan yang cocok dengan tempat, orang, dan aturan di sekitarnya: kiosnya, kertas nomor yang digantikannya, loket yang diringankannya, perawat yang memanggil nomor, pasien yang tidak dapat atau tidak mau memakainya, dan hukum yang melindungi data serta akses mereka. ISO 9241-210, standar internasional untuk desain berpusat pada manusia bagi sistem interaktif, menggambarkan pekerjaan ini sebagai siklus empat kegiatan yang diulang sampai desain memenuhi kebutuhan: memahami dan merinci konteks pemakaian, merinci kebutuhan pengguna, menghasilkan solusi desain, dan mengevaluasi desain terhadap kebutuhan [1]. Bab 4 mengajarkan siklus itu; bab ini menjalankannya sekali secara utuh dan memulai putaran kedua.

Gambar 16.1 menunjukkan jalur proyek akhir dalam empat fase Double Diamond, proses desain yang diterbitkan Design Council Inggris, beserta bab-bab yang metodenya dipakai di setiap fase [2]. Panah di bagian bawah adalah bagian terpenting gambar itu: bukti dari evaluasi mengirim tim kembali ke fase sebelumnya.



Gambar 16.1 Jalur proyek akhir: empat fase, bab-bab yang metodenya diterapkan di setiap fase, dan putaran balik yang mengubah satu desain menjadi proses iteratif.

16.2.2 Setiap bab sebagai keputusan

Cara paling sederhana melihat bagaimana buku ini menyatu adalah mendaftar, untuk setiap bab, keputusan yang diambilnya dalam proyek kios. **Tabel 16.1** melakukannya untuk Bab 2 sampai 15. Bacalah sekarang sebagai peta; subbab-subbab berikutnya menelusurinya.

Tabel 16.1 Metode Bab 2–15 dan keputusan yang disumbangkan masing-masing bagi kios antrean.

Bab	Metode dari bab itu	Keputusannya bagi kios
2	Persepsi, memori, hukum Hick dan Fitts	Hanya empat pilihan; target besar dekat tangan; nomor ditampilkan, tidak perlu diingat
3	Prinsip dan heuristik desain	Status terlihat (nomor dan orang di depan), sesuai dunia nyata (nama layanan seperti di loket), pencegahan galat
4	Proses berpusat pada pengguna	Satu siklus riset, desain, uji, dan iterasi kedua yang direncanakan
5	Riset pengguna	Pengamatan di loket selama dua pagi; wawancara dengan enam pasien dan dua petugas
6	Prototipe	Prototipe kertas, lalu prototipe HTML yang berjalan, dilayani Python dan Java
7	Evaluasi usabilitas	Uji rintis dengan delapan partisipan: tugas, SUS, UMUX-Lite
8	Desain visual dan design system	Satu jenis huruf, satu biru tua untuk tindakan, satu merah hanya untuk bantuan darurat; jarak konsisten
9	Gaya interaksi	Sentuh untuk memilih; tanpa gestur yang perlu dipelajari; loket dan suara tetap menjadi alternatif
10	UX seluler dan responsif	Halaman yang sama berfungsi di tablet, ponsel, dan peramban desktop
11	Aksesibilitas (WCAG 2.2)	Kontras, ukuran target, tautan lompat, bahasa bagian, pesan status, mode huruf besar
12	Etika dan kepercayaan	Tanpa data pribadi; tanpa tekanan; catatan bantuan darurat yang mengarahkan ke manusia
13	Antarmuka AI	Tanpa AI di kios: tak ada yang perlu ditebak di sini, dan tebakan bisa mencelakakan
14	UX berbasis data	Metrik HEART dipilih; hanya hitungan anonim; endpoint status untuk pemantauan
15	Antarmuka masa depan	Ukuran dari jarak pandang; televisi antrean diukur dari bangku terjauh

Baris untuk Bab 13 layak dilihat dua kali. Memilih untuk *tidak* memakai suatu teknologi juga keputusan desain. Kios yang meminta pasien menceritakan keluhannya kepada AI, yang lalu memilihkan layanan, akan menambahkan tebakan di tempat pasien sudah tahu jawabannya, dan tebakan yang salah mengirim orang sakit ke ruang yang salah. Proyek akhir ini memakai setiap metode di tempat metode itu layak, dan tidak di tempat lain.

16.3 Menemukan: Riset di Loket Pendaftaran

Mengikuti Bab 5, tim memulai dengan mengamati. Selama dua pagi, dengan izin kepala puskesmas dan setelah menjelaskan kepada pasien apa yang mereka lakukan, dua mahasiswa duduk di dekat loket pendaftaran dari pukul tujuh sampai sepuluh dan mencatat apa yang terjadi, tanpa merekam nama atau wajah. Mereka mewawancarai enam pasien selagi menunggu, dipilih agar mencakup dua lansia, seorang ibu dengan bayinya, dan seorang pekerja muda, dan mereka mewawancarai Mbak Ratna serta petugas pendaftaran.

Empat temuan membentuk semua yang menyusul. Pertama, sekitar dua pertiga pasien pada pagi-pagi itu adalah pasien *lama* yang sudah tahu layanannya; mereka mengantre hanya untuk mendapatkan kertas nomor. Kedua, sebagian besar pasien lansia datang bersama pendamping, sering cucunya, tetapi ada yang datang sendiri, dan beberapa berkata mereka “tidak berani” menyentuh layar di tempat umum karena takut merusak sesuatu atau tampak bodoh. Ketiga, petugas pendaftaran sering mendahulukan pasien yang tampak sakit berat, dan ia khawatir kios akan menyembunyikan pasien semacam itu di antara yang lain. Keempat, pasien berbicara

bahasa Indonesia, Jawa, dan Madura, dan seorang narasumber, pekerja pabrik dari provinsi lain, meminta formulir dalam "bahasa Indonesia yang sederhana, bukan bahasa kantor".

Dari temuan ini tim menulis dua persona dan satu skenario, seperti dianjurkan Bab 4. *Bu Lestari*, 34 tahun, membawa bayinya ke KIA setiap bulan untuk imunisasi, tahu persis tujuannya, dan ingin mendapat nomor dalam sepuluh detik sambil menggendong bayi dengan satu lengan. *Mbah Sumi*, 72 tahun, datang sendiri setiap Selasa kedua untuk memeriksakan tekanan darah, hanya dapat membaca huruf besar, belum pernah memakai kios layar sentuh, dan tidak akan memakainya bila kios itu membuatnya merasa lamban. Skenarionya mengikuti Mbah Sumi dari becak sampai bangku, dan memuat kalimat yang menjadi kebutuhan pertama: *ia harus dapat mengambil nomor dengan satu sentuhan, atau mengabaikan kios sama sekali dan pergi ke loket seperti biasa.*

16.4 Merumuskan: Kebutuhan dan Batasan

16.4.1 Kebutuhan

Tim menulis kebutuhan sebagai pernyataan yang dapat diuji, masing-masing dapat ditelusuri ke sebuah temuan.

1. Pasien lama mengambil nomor untuk salah satu dari empat layanan dengan satu sentuhan setelah membaca layar (temuan 1).
2. Kios berfungsi tanpa data pribadi apa pun: tanpa nama, tanpa NIK, tanpa nomor telepon (Bab 12; temuan 2).
3. Siapa pun yang bergejala gawat langsung diarahkan kepada manusia (temuan 3).
4. Layar dapat dipakai dalam bahasa Indonesia dan Inggris, dengan kata-kata yang dipakai di loket, dan dapat beralih ke huruf yang lebih besar (temuan 4; Bab 11).
5. Tak seorang pun dipaksa memakai kios: loket tetap buka, dan kertas nomor setara dengan nomor dari kios (aturan kepala puskesmas).
6. Nomor dan jumlah orang di depan ditampilkan dengan huruf besar, dan pasien tidak perlu mengingat apa pun (Bab 2).

16.4.2 Anggaran faktor manusia

Bab 2 memberikan dua hukum yang mengubah kebutuhan 1 menjadi angka. Hukum Hick menyatakan bahwa waktu untuk memilih di antara n pilihan yang sama mungkinnya bertambah menurut logaritma n [3]:

$$T_{\text{pilih}} = a + b\log_2(n+1) \tag{16.1}$$

Dengan empat layanan, pilihannya melibatkan $\log_2 5 = 2.32$ bit; dengan delapan layanan yang tercantum di draf pertama (empat poli ditambah apotek, laboratorium, pertanyaan pendaftaran, dan "lainnya"), ia akan melibatkan $\log_2 9 = 3.17$ bit, lebih dari sepertiga lebih banyak. Tim mempertahankan empat, dan menyerahkan keperluan lain kepada loket.

Hukum Fitts menyatakan bahwa waktu untuk mencapai target bertambah menurut indeks kesulitan, yang bergantung pada jarak D ke target dan lebarnya W searah gerakan [4]:

$$ID = \log_2\left(\frac{D}{W} + 1\right) \tag{16.2}$$

Pada tablet yang dipasang setinggi pinggang, tangan yang bertumpu di tepi bawah menempuh paling jauh sekitar 300 piksel ke tombol tertinggi. Dengan tombol setinggi 64 piksel, $ID = \log_2(300/64 + 1) = \log_2 5.69 = 2.51$ bit; dengan tautan teks setinggi 32 piksel seperti di situs web biasa, nilainya $\log_2(300/32 + 1) = 3.37$ bit. Membuat tombol selebar layar dan tinggi itu murah dan menolong semua orang, terutama mereka yang tangannya gemetar.

16.4.3 Aksesibilitas, hukum, dan etika sebagai batasan

Aksesibilitas bukan fitur yang ditambahkan di akhir; ia sekumpulan batasan yang diketahui sejak awal. WCAG 2.2, yang menjadi standar internasional ISO/IEC 40500:2025 pada Oktober 2025, menyediakan kriteria sukses yang dapat diuji [5]. **Tabel 16.2** memuat kriteria yang harus dipenuhi kios dan cara memenuhinya.

Tabel 16.2 Kriteria sukses WCAG 2.2 yang diperiksa untuk kios antrean dan cara masing-masing dipenuhi.

Kriteria sukses WCAG 2.2	Tuntutan	Cara kios memenuhinya
1.4.3 Contrast (Minimum), AA	Kontras teks setidaknya 4,5:1	Diperiksa dengan `evaluate.py`; pasangan terendah 7,56:1
1.4.11 Non-text Contrast, AA	Indikator fokus dan kontrol setidaknya 3:1	Cincin fokus diganti dari kuning tua (2,27:1) ke biru tua (12,46:1)
1.4.4 Resize Text, AA	Teks dapat diperbesar tanpa kehilangan isi	Tautan “Perbesar huruf”; tata letak mengalir ulang
2.4.1 Bypass Blocks, A	Cara melompati konten berulang	Tautan “Langsung ke pilihan layanan”, tampak saat difokus
2.4.7 Focus Visible, AA	Fokus papan ketik terlihat	Garis tepi tebal dengan jarak pada setiap tautan dan tombol
2.5.8 Target Size (Minimum), AA	Target setidaknya 24 × 24 piksel CSS	Tombol setinggi 64 piksel, selebar layar; tautan sekitar 34 piksel
3.1.1 dan 3.1.2 Language of Page dan of Parts, A dan AA	Bahasa dinyatakan	`lang` pada halaman dan pada tautan ke bahasa lain
4.1.3 Status Messages, AA	Perubahan diumumkan ke teknologi bantu	Nomor ditampilkan dalam wilayah `role="status"` dengan `aria-live`

Hukum menambahkan batasan lain. UU 8/2016 tentang Penyandang Disabilitas menjadikan pelayanan publik yang aksesibel sebagai hak [6], dan PP 42/2020 mengatur aksesibilitas pelayanan publik bagi penyandang disabilitas [7]; puskesmas adalah pelayanan publik. Di luar Indonesia, European Accessibility Act berlaku sejak 28 Juni 2025 bagi banyak produk dan layanan, termasuk terminal layan mandiri seperti mesin tiket dan mesin *check-in* [8], sebuah tanda ke mana dunia bergerak. Dan karena kios tidak mengumpulkan data pribadi sama sekali, ia tidak memerlukan layar persetujuan menurut UU PDP 27/2022 dan PP 33/2026 [9]; cara termurah mematuhi undang-undang pelindungan data adalah tidak mengumpulkan apa pun yang dilindunginya.

Aturan etika Bab 12 melengkapi batasannya: tanpa penghitung mundur, tanpa “yakin ingin keluar?”, tanpa iklan di kios kesehatan, dan catatan bantuan darurat yang mengarahkan ke manusia alih-alih mencoba mendiagnosis.

16.5 Mengembangkan: dari Kertas ke Kios yang Berjalan

16.5.1 Prototipe kertas

Bab 6 menganjurkan memulai di atas kertas, dan tim melakukannya. Tiga sketsa di kertas A4, dipegang di depan dudukan tablet, cukup untuk menguji alurnya bersama empat pasien di ruang tunggu. Dua pelajaran langsung muncul. Sketsa pertama dibuka dengan layar sambutan dan tombol "Mulai"; keempat pasien justru menyentuh logonya, dan seorang bertanya "rusak, ya?". Sketsa kedua menaruh keempat layanan di layar pertama, dan keempat pasien memilih dengan benar tanpa ragu. Mbah Sumi, yang bersedia mencoba, berkata tulisannya "terlalu kecil untuk mata tua", dan dari situlah mode huruf besar bermula. Prototipe seharga sepuluh menit menghapus satu layar utuh dari desain.

16.5.2 Keputusan desain

Keputusan-keputusan berikutnya tampak pada kios yang berjalan di **Gambar 16.2**. Tautan bahasa dan ukuran huruf berada di atas (penanda 1). Catatan bantuan darurat, satu-satunya elemen merah, meminta siapa pun yang mengalami nyeri dada, sesak napas, perdarahan hebat, atau membawa bayi demam tinggi untuk segera ke loket pendaftaran (penanda 2): kios tidak melakukan triase, ia menyerahkan kepada manusia, persis seperti praktik petugas pendaftaran. Keempat layanan berupa tombol selebar layar dalam satu warna biru tua, dinamai seperti loket menamainya (penanda 3). Setelah disentuh, layar kedua menampilkan layanan, nomor dengan huruf sangat besar, dan jumlah orang di depan, di dalam wilayah status yang diumumkan pembaca layar (penanda 4), dengan satu tombol "Selesai".



Gambar 16.2 Kios yang berjalan, ditangkap dengan Playwright: layar utama dan layar nomor. Penanda: 1 bahasa dan ukuran huruf, 2 catatan bantuan darurat, 3 keempat layanan, 4 nomor dan jumlah orang di depan dalam wilayah status.

Dua keputusan sengaja ditinggalkan. Tidak ada perkiraan waktu tunggu, karena puskesmas tidak dapat memperkirakannya dengan jujur; "orang di depan Anda" adalah fakta, perkiraan menit adalah tebakan. Dan tidak ada JavaScript: setiap halaman berupa HTML biasa dan formulir, sehingga kios berjalan di tablet lama, dengan fitur aksesibilitas bawaan peramban, dan tidak dapat rusak karena galat skrip.

16.6 Membangun Kios

16.6.1 Kios dalam Python

Kode 16.1 adalah seluruh kios sebagai aplikasi Flask. Kamus `TEXT` menyimpan setiap teks dalam kedua bahasa, sehingga berganti bahasa hanya mengubah kata-kata dan tidak ada yang lain. Fungsi `page` membungkus isi halaman dalam tata letak bersama: `lang` pada halaman, tautan lompat, tautan alat (dengan `lang` pada tautan ke bahasa lain), dan lembar gaya. Rute `/` menampilkan layanan sebagai tombol dalam satu formulir; `/take` mengeluarkan nomor berikutnya untuk layanan yang dipilih dan menampilkannya bersama jumlah orang di depan; dan `/status.json` mengembalikan hitungan yang dapat dibaca loket atau dasbor.

Kode 16.1 `kiosk_app.py`: kios antrean dalam Python dengan Flask, dalam dua bahasa dan dengan mode huruf besar.

```
"""Kios antrean puskesmas: target besar, dua bahasa, tanpa JavaScript."""
from html import escape

from flask import Flask, redirect, request

TEXT = {
    "id": {"title": "Ambil Nomor Antrean", "skip": "Langsung ke pilihan layanan",
           "pick": "Pilih layanan yang Anda tuju:",
           "urgent": "Nyeri dada, sesak napas, perdarahan hebat, atau bayi "
                     "demam tinggi? Jangan menunggu: datangi loket "
                     "pendaftaran sekarang dan beri tahu petugas.",
           "your": "Nomor Anda", "ahead": "Orang di depan Anda",
           "show": "Tunjukkan nomor ini di loket saat dipanggil.",
           "done": "Selesai", "bigger": "Perbesar huruf",
           "smaller": "Huruf normal", "other": "English",
           "services": {"U": "Poli Umum", "K": "KIA (Ibu dan Anak)",
                        "G": "Poli Gigi", "L": "Lansia (60 tahun ke atas)"}},
    "en": {"title": "Take a Queue Number", "skip": "Skip to the services",
           "pick": "Choose the service you need:",
           "urgent": "Chest pain, hard to breathe, heavy bleeding, or a baby "
                     "with a high fever? Do not wait: go to the registration "
                     "counter now and tell the staff.",
           "your": "Your number", "ahead": "People ahead of you",
           "show": "Show this number at the counter when it is called.",
           "done": "Done", "bigger": "Larger text",
           "smaller": "Normal text", "other": "Bahasa Indonesia",
           "services": {"U": "General clinic", "K": "Mother and child",
                        "G": "Dental clinic", "L": "Older adults (60+)"}},
}
CSS = """body{font:1.4rem/1.5 system-ui,sans-serif;margin:0;color:#111827;
background:#fff}body.big{font-size:1.9rem}
main{max-width:40rem;margin:auto;padding:1rem}
h1{color:#12355B;margin:.5rem 0}
.skip{position:absolute;left:-99em}.skip:focus{left:1rem}
.urgent{background:#FDECEA;border-left:.5rem solid #C0392B;
padding:.75rem}
button,.btn{display:block;width:100%;min-height:4rem;margin:.6rem 0;
font:inherit;color:#fff;background:#12355B;border:0;
border-radius:.5rem;text-align:center;text-decoration:none;
padding:.8rem}.btn{display:flex;align-items:center;
justify-content:center}.tools a{color:#0B5E5E;margin-right:1.5rem}
a:focus,button:focus{outline:.25rem solid #12355B;outline-offset:.2rem}
.number{font-size:4rem;color:#12355B;font-weight:bold;margin:0}"""
queues = {code: 0 for code in "UKGL"}     # nomor yang diberikan hari ini
served = {code: 0 for code in "UKGL"}     # nomor yang sudah dipanggil
```

Kode 16.1 (Lanjutan) *kiosk_app.py: kios antrean dalam Python dengan Flask, dalam dua bahasa dan dengan mode huruf besar.*

```
def page(lang, big, body):
    t = TEXT[lang]
    other = "en" if lang == "id" else "id"
    size = "0" if big else "1"
    return ("<!DOCTYPE html>\n<html lang=\"%s\"><head><meta charset=\"utf-8\">"
            "<meta name=\"viewport\" content=\"width=device-width\">"
            "<title>%s</title><style>%s</style></head>\n<body%s>"
            "<a class=\"skip\" href=\"#services\">%s</a><main>"
            "<p class=\"tools\"><a href=\"/?lang=%s&amp;big=%s\" lang=\"%s\">%s</a>"
            "<a href=\"/?lang=%s&amp;big=%s\">%s</a></p>\n%s</main></body></html>"
            % (lang, t["title"], CSS, " class=\"big\"" if big else "",
               t["skip"], other, "1" if big else "0", other, t["other"],
               lang, size, t["smaller"] if big else t["bigger"], body))


app = Flask(__name__)


def options():
    lang = request.values.get("lang", "id")
    return (lang if lang in TEXT else "id"), request.values.get("big") == "1"


@app.get("/")
def home():
    lang, big = options()
    t = TEXT[lang]
    buttons = "".join(
        "<button name=\"service\" value=\"%s\">%s</button>" % (c, escape(n))
        for c, n in t["services"].items())
    body = ("<h1>%s</h1><p class=\"urgent\" role=\"note\">%s</p>"
            "<form method=\"post\" action=\"/take\" id=\"services\">"
            "<input type=\"hidden\" name=\"lang\" value=\"%s\">"
            "<input type=\"hidden\" name=\"big\" value=\"%s\">"
            "<p>%s</p>%s</form>" % (t["title"], t["urgent"], lang,
                                    "1" if big else "0", t["pick"], buttons))
    return page(lang, big, body)
```

Kode 16.1 (Lanjutan) *kiosk_app.py: kios antrean dalam Python dengan Flask, dalam dua bahasa dan dengan mode huruf besar.*

```
@app.post("/take")
def take():
    lang, big = options()
    code = request.form.get("service", "")
    if code not in queues:
        return redirect("/?lang=" + lang)
    queues[code] += 1
    t = TEXT[lang]
    body = ("<h1>%s</h1><div role=\"status\" aria-live=\"polite\">"
            "<p>%s:</p><p class=\"number\">%s-%03d</p>"
            "<p>%s: <strong>%d</strong></p></div><p>%s</p>"
            "<a class=\"btn\" href=\"/?lang=%s&amp;big=%s\">%s</a>"
            % (escape(t["services"][code]), t["your"], code, queues[code],
               t["ahead"], queues[code] - served[code] - 1, t["show"],
               lang, "1" if big else "0", t["done"]))
    return page(lang, big, body)


@app.get("/status.json")
def status():
    return {c: {"given": queues[c], "served": served[c]} for c in queues}


if __name__ == "__main__":
    app.run(host="127.0.0.1", port=4160)
```

Empat rincian membawa keputusan desain ke dalam kode. Teks di-*escape* dengan `html.escape` di tempat yang dapat memuat karakter khusus. Formulir mengirim ke `/take` dan hasilnya halaman baru, sehingga tombol Kembali di peramban maupun tombol “Selesai” sama-sama kembali ke awal yang bersih. Kode layanan yang tak dikenal cukup dialihkan ke halaman utama, sehingga formulir yang diutak-atik tidak dapat menciptakan antrean aneh. Dan garis fokusnya biru tua, warna yang dituntut pemeriksaan kontras di Subbab 16.7.

16.6.2 Kios dalam Java

Kode 16.2 adalah kios yang sama dengan `com.sun.net.httpserver` bawaan JDK, sehingga dapat berjalan hanya dengan Java terpasang. Ia menyajikan HTML yang persis sama dengan versi Python; tim memeriksanya dengan mengambil kedua halaman lewat `curl` dan membandingkannya bita demi bita.

Kode 16.2 `KioskServer.java`: kios yang sama dengan server HTTP bawaan JDK.

```
import com.sun.net.httpserver.*;
import java.io.*;
import java.net.*;
import java.nio.charset.StandardCharsets;
import java.util.*;

/** Kios antrean puskesmas: target besar, dua bahasa, tanpa JavaScript. */
public class KioskServer {
    static final Map<String, Map<String, String>> TEXT = Map.of(
        "id", Map.ofEntries(Map.entry("title", "Ambil Nomor Antrean"),
            Map.entry("skip", "Langsung ke pilihan layanan"),
            Map.entry("pick", "Pilih layanan yang Anda tuju:"),
            Map.entry("urgent", "Nyeri dada, sesak napas, perdarahan hebat, "
                + "atau bayi demam tinggi? Jangan menunggu: datangi loket "
                + "pendaftaran sekarang dan beri tahu petugas."),
            Map.entry("your", "Nomor Anda"),
            Map.entry("ahead", "Orang di depan Anda"),
            Map.entry("show", "Tunjukkan nomor ini di loket saat dipanggil."),
            Map.entry("done", "Selesai"), Map.entry("bigger", "Perbesar huruf"),
            Map.entry("smaller", "Huruf normal"), Map.entry("other", "English"),
            Map.entry("U", "Poli Umum"), Map.entry("K", "KIA (Ibu dan Anak)"),
            Map.entry("G", "Poli Gigi"),
            Map.entry("L", "Lansia (60 tahun ke atas)")),
        "en", Map.ofEntries(Map.entry("title", "Take a Queue Number"),
            Map.entry("skip", "Skip to the services"),
            Map.entry("pick", "Choose the service you need:"),
            Map.entry("urgent", "Chest pain, hard to breathe, heavy bleeding, "
                + "or a baby with a high fever? Do not wait: go to the "
                + "registration counter now and tell the staff."),
            Map.entry("your", "Your number"),
            Map.entry("ahead", "People ahead of you"),
            Map.entry("show",
                "Show this number at the counter when it is called."),
            Map.entry("done", "Done"), Map.entry("bigger", "Larger text"),
            Map.entry("smaller", "Normal text"),
            Map.entry("other", "Bahasa Indonesia"),
            Map.entry("U", "General clinic"), Map.entry("K", "Mother and child"),
            Map.entry("G", "Dental clinic"), Map.entry("L", "Older adults (60+)")));
    static final String CSS = """
        body{font:1.4rem/1.5 system-ui,sans-serif;margin:0;color:#111827;
        background:#fff}body.big{font-size:1.9rem}
        main{max-width:40rem;margin:auto;padding:1rem}
        h1{color:#12355B;margin:.5rem 0}
        .skip{position:absolute;left:-99em}.skip:focus{left:1rem}
        .urgent{background:#FDECEA;border-left:.5rem solid #C0392B;
        padding:.75rem}
```

Kode 16.2 (Lanjutan) *KioskServer.java: kios yang sama dengan server HTTP bawaan JDK.*

```
        button,.btn{display:block;width:100%;min-height:4rem;margin:.6rem 0;
        font:inherit;color:#fff;background:#12355B;border:0;
        border-radius:.5rem;text-align:center;text-decoration:none;
        padding:.8rem}.btn{display:flex;align-items:center;
        justify-content:center}.tools a{color:#0B5E5E;margin-right:1.5rem}
        a:focus,button:focus{outline:.25rem solid #12355B;outline-offset:.2rem}
        .number{font-size:4rem;color:#12355B;font-weight:bold;margin:0}""";
    static final String CODES = "UKGL";
    static final Map<String, Integer> queues = new LinkedHashMap<>();  // diberikan
    static final Map<String, Integer> served = new LinkedHashMap<>();  // dipanggil

    static String esc(String s) {
        return s.replace("&", "&amp;").replace("<", "&lt;").replace(">", "&gt;")
            .replace("\"", "&quot;").replace("'", "&#x27;");
    }

    static String page(String lang, boolean big, String body) {
        var t = TEXT.get(lang);
        String other = lang.equals("id") ? "en" : "id";
        return String.format("<!DOCTYPE html>\n<html lang=\"%s\"><head>"
            + "<meta charset=\"utf-8\"><meta name=\"viewport\" "
            + "content=\"width=device-width\"><title>%s</title><style>%s</style>"
            + "</head>\n<body%s><a class=\"skip\" href=\"#services\">%s</a><main>"
            + "<p class=\"tools\"><a href=\"/?lang=%s&amp;big=%s\" "
            + "lang=\"%s\">%s</a><a href=\"/?lang=%s&amp;big=%s\">%s</a></p>"
            + "\n%s</main></body></html>",
            lang, t.get("title"), CSS, big ? " class=\"big\"" : "", t.get("skip"),
            other, big ? "1" : "0", other, t.get("other"), lang, big ? "0" : "1",
            big ? t.get("smaller") : t.get("bigger"), body);
    }

    static Map<String, String> form(HttpExchange ex) throws IOException {
        String q = ex.getRequestURI().getRawQuery();
        String body = new String(ex.getRequestBody().readAllBytes(),
            StandardCharsets.UTF_8);
        Map<String, String> m = new HashMap<>();
        for (String part : ((q == null ? "" : q) + "&" + body).split("&")) {
            String[] kv = part.split("=", 2);
            if (kv.length == 2)
                m.put(URLDecoder.decode(kv[0], StandardCharsets.UTF_8),
                    URLDecoder.decode(kv[1], StandardCharsets.UTF_8));
        }
        return m;
    }
```

Kode 16.2 (Lanjutan) *KioskServer.java: kios yang sama dengan server HTTP bawaan JDK.*

```
    static void send(HttpExchange ex, int code, String type, String text)
            throws IOException {
        byte[] b = text.getBytes(StandardCharsets.UTF_8);
        ex.getResponseHeaders().set("Content-Type", type + "; charset=utf-8");
        ex.sendResponseHeaders(code, b.length);
        try (OutputStream os = ex.getResponseBody()) { os.write(b); }
    }

    static void handle(HttpExchange ex) throws IOException {
        var f = form(ex);
        String lang = f.getOrDefault("lang", "id");
        if (!TEXT.containsKey(lang)) lang = "id";   // Map.of menolak kunci null
        boolean big = "1".equals(f.get("big"));
        var t = TEXT.get(lang);
        String path = ex.getRequestURI().getPath();
        if (path.equals("/status.json")) {
            StringBuilder j = new StringBuilder("{");
            for (String c : queues.keySet())
                j.append(String.format("%s\"%s\": {\"given\": %d, \"served\": %d}",
                    j.length() > 1 ? ", " : "", c, queues.get(c), served.get(c)));
            send(ex, 200, "application/json", j + "}");
        } else if (path.equals("/take") && ex.getRequestMethod().equals("POST")) {
            String code = f.getOrDefault("service", "");
            if (code.length() != 1 || !CODES.contains(code)) {
                ex.getResponseHeaders().set("Location", "/?lang=" + lang);
                send(ex, 303, "text/plain", "");
                return;
            }
            int n = queues.merge(code, 1, Integer::sum);
            String body = String.format("<h1>%s</h1><div role=\"status\" "
                + "aria-live=\"polite\"><p>%s:</p><p class=\"number\">%s-%03d</p>"
                + "<p>%s: <strong>%d</strong></p></div><p>%s</p>"
                + "<a class=\"btn\" href=\"/?lang=%s&amp;big=%s\">%s</a>",
                esc(t.get(code)), t.get("your"), code, n, t.get("ahead"),
                n - served.get(code) - 1, t.get("show"), lang, big ? "1" : "0",
                t.get("done"));
            send(ex, 200, "text/html", page(lang, big, body));
        } else if (path.equals("/")) {
            StringBuilder buttons = new StringBuilder();
            for (char c : CODES.toCharArray())
                buttons.append(String.format(
                    "<button name=\"service\" value=\"%c\">%s</button>",
                    c, esc(t.get(String.valueOf(c)))));
            String body = String.format("<h1>%s</h1><p class=\"urgent\" "
                + "role=\"note\">%s</p><form method=\"post\" action=\"/take\" "
                + "id=\"services\"><input type=\"hidden\" name=\"lang\" "
```

Kode 16.2 (Lanjutan) *KioskServer.java: kios yang sama dengan server HTTP bawaan JDK.*

```
                + "value=\"%s\"><input type=\"hidden\" name=\"big\" value=\"%s\">"
                + "<p>%s</p>%s</form>", t.get("title"), t.get("urgent"), lang,
                big ? "1" : "0", t.get("pick"), buttons);
            send(ex, 200, "text/html", page(lang, big, body));
        } else send(ex, 404, "text/plain", "not found");
    }

    public static void main(String[] args) throws IOException {
        for (char c : CODES.toCharArray()) {
            queues.put(String.valueOf(c), 0);
            served.put(String.valueOf(c), 0);
        }
        HttpServer server = HttpServer.create(
            new InetSocketAddress("127.0.0.1", 4161), 0);
        server.createContext("/", ex -> {
            try {
                handle(ex);
            } catch (Exception e) {   // tampilkan galat, tetap melayani
                e.printStackTrace();
                send(ex, 500, "text/plain", "internal error");
            }
        });
        server.start();
        System.out.println("Kiosk running at http://127.0.0.1:4161/");
    }
}
```

Dua perbedaan kecil antarbahasa memberi tim pelajaran. `Map.of` di Java menolak kunci `null` dengan pengecualian, sehingga versi pertama mogok pada setiap permintaan tanpa parameter `lang`; perbaikannya membaca parameter dengan nilai bawaan sebelum mencarinya, dan penangan kini menangkap pengecualian, mencatatnya, dan menjawab dengan halaman galat alih-alih menutup koneksi. Kios tidak boleh macet diam-diam: pasien di depan layar yang membeku tidak tahu harus menunggu atau pergi.

16.6.3 Menjalankannya

Jalankan kios Python dengan `python kiosk_app.py` di folder `python` lalu buka http://127.0.0.1:4160/, atau kios Java dengan `javac -d out *.java` dan `java -cp out KioskServer` di folder `java` lalu buka http://127.0.0.1:4161/. Kios juga dapat dikendalikan dari terminal, dan begitulah tim mengujinya sebelum ada pengguna yang melihatnya. Mengambil dua nomor untuk poli lansia

```
$ curl -s -X POST -d "lang=en&service=L" http://127.0.0.1:4160/take
$ curl -s -X POST -d "lang=en&service=L" http://127.0.0.1:4160/take
```

mengembalikan dua halaman yang baris-baris kuncinya adalah

```
L-001
ahead of you: 0
L-002
ahead of you: 1
```

dan endpoint status, yang dicetak rapi dengan `curl -s http://127.0.0.1:4160/status.json | python -m json.tool`, kemudian melaporkan

```
{
    "G": {
        "given": 0,
        "served": 0
    },
    "K": {
        "given": 0,
        "served": 0
    },
    "L": {
        "given": 2,
        "served": 0
    },
    "U": {
        "given": 0,
        "served": 0
    }
}
```

Tangkapan layar di bab ini berasal dari kios yang sedang berjalan itu sendiri, ditangkap oleh skrip Playwright pada **Kode 16.3** [10]. Skrip itu membuka halaman utama dalam kedua bahasa, menekan tombol keempat, membuka mode huruf besar, dan menekan Tab sekali sehingga tautan lompat muncul bersama cincin fokusnya. **Gambar 16.3** menunjukkan hasilnya: huruf yang lebih besar mengalir ulang tanpa terpotong, dan cincin fokus biru tua tampak jelas di sekeliling tautan lompat.

Kode 16.3 `shoot.py`: tangkapan layar Playwright atas kios kita sendiri yang sedang berjalan, dalam kedua bahasa.

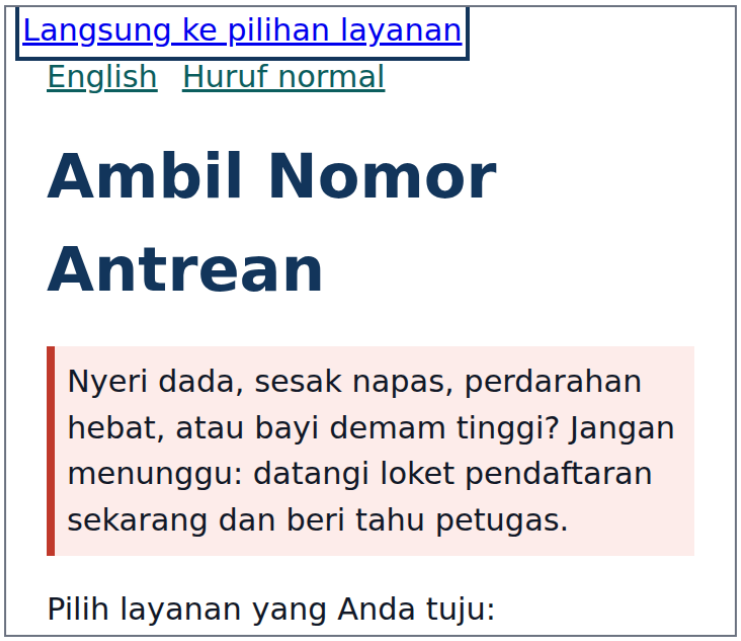
```python
"""Tangkapan layar kios kita yang sedang berjalan (jalankan kiosk_app.py dulu)."""
import sys

from playwright.sync_api import sync_playwright

BASE = "http://127.0.0.1:4160"


def main(out_dir):
    with sync_playwright() as p:
        browser = p.chromium.launch()
        page = browser.new_page(viewport={"width": 720, "height": 900})
        for lang in ("en", "id"):
            page.goto(BASE + "/?lang=" + lang)
            page.screenshot(path="%s/kiosk_home_%s.png" % (out_dir, lang))
            page.get_by_role("button").nth(3).click()     # lansia
            page.screenshot(path="%s/kiosk_number_%s.png" % (out_dir, lang))
            page.goto(BASE + "/?lang=%s&big=1" % lang)
            page.keyboard.press("Tab")                    # tautan lompat
            page.screenshot(path="%s/kiosk_big_%s.png" % (out_dir, lang))
        browser.close()


if __name__ == "__main__":
    main(sys.argv[1] if len(sys.argv) > 1 else ".")
```



Gambar 16.3 Mode huruf besar setelah satu kali menekan tombol Tab: tautan lompat muncul dengan cincin fokus biru tua, dan teks mengalir ulang.

16.7 Mengevaluasi Kios

16.7.1 Uji rintis

Bab 7 menjelaskan uji usabilitas; tim menjalankan uji kecil di ruang rapat puskesmas dengan delapan partisipan: tiga berusia 18–39, dua berusia 40–59, dan tiga berusia 60 tahun ke atas. Delapan orang cukup untuk menemukan sebagian besar masalah serius, sesuai temuan Nielsen dan Landauer bahwa sejumlah kecil pengguna menyingkap sebagian besar masalah bila uji diulang setelah setiap perancangan ulang [11], tetapi jauh terlalu sedikit untuk rerata yang presisi, dan analisisnya harus mengatakan demikian. Setiap partisipan mengerjakan tiga tugas pada **Tabel 16.3** sambil berpikir keras, lalu menjawab sepuluh pernyataan SUS dan dua pernyataan UMUX-Lite.

Tabel 16.3 Tiga tugas uji rintis dan kriteria keberhasilannya.

Tugas	Instruksi yang dibacakan kepada partisipan	Berhasil berarti
T1	“Anda datang untuk poli umum. Ambillah nomor Anda.”	Nomor U tampil
T2	“Teman Anda lebih lancar membaca bahasa Inggris. Ubah layar ke bahasa Inggris dan ambil nomor untuk KIA.”	Layar berbahasa Inggris dan nomor K tampil
T3	“Hurufnya terlalu kecil bagi Anda. Perbesar hurufnya dan ambil nomor untuk poli lansia.”	Huruf besar aktif dan nomor L tampil

Hasilnya tersimpan dalam tiga berkas CSV di `data/`: `sus.csv` berisi sepuluh jawaban setiap partisipan dari 1 sampai 5, `umux_lite.csv` berisi dua jawaban UMUX-Lite dari 1 sampai 7, dan `tasks.csv` berisi keberhasilan dan detik per tugas. Ketiganya adalah set data latihan yang berbentuk seperti uji rintis nyata, sehingga analisisnya dapat dipelajari sebelum partisipan sungguhan direkrut.

16.7.2 Ukuran-ukurannya

System Usability Scale memiliki sepuluh pernyataan, bergantian positif (butir ganjil) dan negatif (butir genap), masing-masing dijawab dari 1 (sangat tidak setuju) sampai 5 (sangat setuju) [12]. Setiap butir ganjil menyumbang jawabannya dikurangi 1, setiap butir genap 5 dikurangi jawabannya, dan jumlahnya dikalikan 2,5:

$$\text{SUS} = 2.5\left[\sum_{i\in\{1,3,5,7,9\}}(a_i - 1) + \sum_{i\in\{2,4,6,8,10\}}(5 - a_i)\right] \tag{16.3}$$

yang memberi skor 0 sampai 100. Skor ini bukan persentase. Di banyak studi rerata skor SUS sekitar 68, sehingga skor di atas 68 lebih baik daripada rata-rata [13].

UMUX-Lite hanya mengajukan dua pernyataan, “Kemampuan sistem ini memenuhi kebutuhan saya” dan “Sistem ini mudah dipakai”, masing-masing dari 1 sampai 7 [14]. Skor mentahnya, dan skor hasil penyesuaian regresi yang diusulkan Lewis, Utesch, dan Maher agar sebanding dengan SUS, adalah

$$\text{UMUX-Lite} = \frac{(u_1 - 1) + (u_2 - 1)}{12}\cdot 100, \qquad \text{setara SUS} = 0.65\cdot\text{UMUX-Lite} + 22.9 \tag{16.4}$$

Untuk penyelesaian tugas dengan sedikit partisipan, rumus selang biasa gagal pada 8 dari 8 (ia akan mengklaim kepastian). Selang Wald yang disesuaikan (*adjusted Wald*), yang dianjurkan Sauro dan Lewis untuk sampel kecil [13], menambahkan $z^2/2$ keberhasilan dan z^2 percobaan sebelum menghitung selang, dengan $z = 1.96$ untuk 95%:

$$\hat{p} = \frac{x + z^2/2}{n + z^2}, \qquad \hat{p} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n + z^2}} \tag{16.5}$$

Waktu tugas bersifat menceng, beberapa partisipan yang lambat menarik rerata aritmetika ke atas, jadi yang dilaporkan adalah rerata geometrik waktu dari tugas yang berhasil.

Akhirnya, kontras warna WCAG 2.2 dihitung dari *luminans relatif* L setiap warna, dengan setiap kanal sRGB c antara 0 dan 1 dilinearkan lebih dulu, $c_{\text{lin}} = c/12.92$ untuk $c \le 0.04045$ dan $((c + 0.055)/1.055)^{2.4}$ untuk lainnya, lalu dibobot [5]:

$$L = 0.2126\,R_{\text{lin}} + 0.7152\,G_{\text{lin}} + 0.0722\,B_{\text{lin}}, \qquad \text{rasio} = \frac{L_1 + 0.05}{L_2 + 0.05} \tag{16.6}$$

dengan L_1 warna yang lebih terang dan L_2 yang lebih gelap. Teks memerlukan setidaknya 4,5:1, sedangkan indikator fokus dan elemen nonteks lain setidaknya 3:1.

16.7.3 Program evaluasi

Kode 16.4 menghitung semuanya: SUS per partisipan dengan Persamaan (16.3) beserta reratanya dengan selang 95% berbasis t, UMUX-Lite dengan Persamaan (16.4), penyelesaian per tugas dengan selang pada Persamaan (16.5) dan rerata geometrik waktunya, serta kontras setiap pasangan warna dalam lembar gaya kios dengan Persamaan (16.6), termasuk cincin fokus kuning tua versi draf dan cincin biru tua yang menggantikannya.

Kode 16.4 `evaluate.py`: SUS, UMUX-Lite, penyelesaian tugas dengan selang Wald yang disesuaikan, dan kontras warna WCAG.

```
"""Evaluasi uji rintis kios: SUS, UMUX-Lite, tugas, dan kontras warna."""
import csv
import math
import sys

T95 = [12.706, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306, 2.262,
       2.228, 2.201, 2.179]              # t untuk 95%, db = 1 .. 12


def rows(path):
    with open(path, encoding="utf-8") as f:
        return list(csv.DictReader(f))


def sus(answers):
    """System Usability Scale: 10 jawaban 1..5 -> 0..100."""
    odd = sum(answers[i] - 1 for i in range(0, 10, 2))
    even = sum(5 - answers[i] for i in range(1, 10, 2))
    return 2.5 * (odd + even)


def mean_ci(xs):
    n = len(xs)
    m = sum(xs) / n
    sd = math.sqrt(sum((x - m) ** 2 for x in xs) / (n - 1))
    t = T95[n - 2] if n - 1 <= len(T95) else 1.96
    return m, sd, m - t * sd / math.sqrt(n), m + t * sd / math.sqrt(n)


def adjusted_wald(successes, n, z=1.96):
    """Interval 95% untuk laju penyelesaian, tetap baik untuk n kecil."""
    p = (successes + z * z / 2) / (n + z * z)
    half = z * math.sqrt(p * (1 - p) / (n + z * z))
    return max(0.0, p - half), min(1.0, p + half)


def luminance(hex_color):
    """Luminans relatif WCAG untuk warna seperti '#12355B'."""
    def channel(c):
        c = int(hex_color[c:c + 2], 16) / 255
        return c / 12.92 if c <= 0.04045 else ((c + 0.055) / 1.055) ** 2.4
    return 0.2126 * channel(1) + 0.7152 * channel(3) + 0.0722 * channel(5)
```

Kode 16.4 (Lanjutan) *evaluate.py: SUS, UMUX-Lite, penyelesaian tugas dengan selang Wald yang disesuaikan, dan kontras warna WCAG.*

```
def contrast(a, b):
    hi, lo = sorted((luminance(a), luminance(b)), reverse=True)
    return (hi + 0.05) / (lo + 0.05)


PAIRS = [("button text", "#FFFFFF", "#12355B", 4.5),
         ("body text", "#111827", "#FFFFFF", 4.5),
         ("tool links", "#0B5E5E", "#FFFFFF", 4.5),
         ("urgent note", "#111827", "#FDECEA", 4.5),
         ("focus, draft", "#E0A100", "#FFFFFF", 3.0),
         ("focus, fixed", "#12355B", "#FFFFFF", 3.0)]


def main(data):
    print("SUS per participant")
    scores = {}
    for r in rows(data + "/sus.csv"):
        s = sus([int(r["q%d" % i]) for i in range(1, 11)])
        scores[r["participant"]] = s
        print("  %s  %-6s %5.1f" % (r["participant"], r["age_group"], s))
    m, sd, lo, hi = mean_ci(list(scores.values()))
    print("SUS mean %.1f (SD %.1f), 95%% CI [%.1f, %.1f]" % (m, sd, lo, hi))
    lite = [(int(r["capabilities"]) - 1 + int(r["easy"]) - 1) / 12 * 100
            for r in rows(data + "/umux_lite.csv")]
    m2 = sum(lite) / len(lite)
    print("UMUX-Lite mean %.1f, SUS-comparable %.1f" % (m2, 0.65 * m2 + 22.9))
    print("\ntask  done  completion  95% CI (adj. Wald)  geo. mean time")
    tasks = {}
    for r in rows(data + "/tasks.csv"):
        tasks.setdefault(r["task"], []).append(
            (r["success"] == "1", int(r["seconds"])))
    for task, results in sorted(tasks.items()):
        ok = [sec for success, sec in results if success]
        lo, hi = adjusted_wald(len(ok), len(results))
        geo = math.exp(sum(math.log(s) for s in ok) / len(ok))
        print("%-4s  %d/%d  %9.0f%%  [%3.0f%%, %3.0f%%]  %14.1f s" % (
            task, len(ok), len(results), 100 * len(ok) / len(results),
            100 * lo, 100 * hi, geo))
    print("\ncolor pair      ratio  needed  verdict")
    for name, fg, bg, need in PAIRS:
        ratio = contrast(fg, bg)
        print("%-14s %6.2f  %6.1f  %s" % (name, ratio, need,
                                          "pass" if ratio >= need else "FAIL"))
```

Kode 16.4 (Lanjutan) *evaluate.py: SUS, UMUX-Lite, penyelesaian tugas dengan selang Wald yang disesuaikan, dan kontras warna WCAG.*

```
if __name__ == "__main__":
    main(sys.argv[1] if len(sys.argv) > 1 else "../data")
```

Kode 16.5 adalah evaluasi yang sama dalam Java.

Kode 16.5 `Evaluate.java`: evaluasi yang sama dalam Java.

```java
import java.nio.file.*;
import java.util.*;

/** Evaluasi uji rintis kios: SUS, UMUX-Lite, tugas, dan kontras warna. */
public class Evaluate {
    static final double[] T95 = {12.706, 4.303, 3.182, 2.776, 2.571, 2.447,
        2.365, 2.306, 2.262, 2.228, 2.201, 2.179};   // t untuk 95%, db 1..12

    static List<String[]> rows(String path) throws Exception {
        List<String> lines = Files.readAllLines(Path.of(path));
        List<String[]> out = new ArrayList<>();
        for (String l : lines.subList(1, lines.size())) out.add(l.split(","));
        return out;
    }

    /** System Usability Scale: 10 jawaban 1..5 -> 0..100. */
    static double sus(int[] a) {
        int sum = 0;
        for (int i = 0; i < 10; i += 2) sum += a[i] - 1;
        for (int i = 1; i < 10; i += 2) sum += 5 - a[i];
        return 2.5 * sum;
    }

    static double[] meanCi(List<Double> xs) {
        int n = xs.size();
        double m = 0, ss = 0;
        for (double x : xs) m += x;
        m /= n;
        for (double x : xs) ss += (x - m) * (x - m);
        double sd = Math.sqrt(ss / (n - 1));
        double t = n - 1 <= T95.length ? T95[n - 2] : 1.96;
        return new double[] {m, sd, m - t * sd / Math.sqrt(n),
            m + t * sd / Math.sqrt(n)};
    }

    /** Interval 95% untuk laju penyelesaian, tetap baik untuk n kecil. */
    static double[] adjustedWald(int successes, int n) {
        double z = 1.96, p = (successes + z * z / 2) / (n + z * z);
        double half = z * Math.sqrt(p * (1 - p) / (n + z * z));
        return new double[] {Math.max(0, p - half), Math.min(1, p + half)};
    }
```

Kode 16.5 (Lanjutan) *Evaluate.java: evaluasi yang sama dalam Java.*

```
    /** Luminans relatif WCAG untuk warna seperti "#12355B". */
    static double luminance(String hex) {
        double[] w = {0.2126, 0.7152, 0.0722};
        double y = 0;
        for (int k = 0; k < 3; k++) {
            double c = Integer.parseInt(hex.substring(1 + 2 * k, 3 + 2 * k), 16)
                / 255.0;
            c = c <= 0.04045 ? c / 12.92 : Math.pow((c + 0.055) / 1.055, 2.4);
            y += w[k] * c;
        }
        return y;
    }

    static double contrast(String a, String b) {
        double la = luminance(a), lb = luminance(b);
        return (Math.max(la, lb) + 0.05) / (Math.min(la, lb) + 0.05);
    }

    static final String[][] PAIRS = {
        {"button text", "#FFFFFF", "#12355B", "4.5"},
        {"body text", "#111827", "#FFFFFF", "4.5"},
        {"tool links", "#0B5E5E", "#FFFFFF", "4.5"},
        {"urgent note", "#111827", "#FDECEA", "4.5"},
        {"focus, draft", "#E0A100", "#FFFFFF", "3.0"},
        {"focus, fixed", "#12355B", "#FFFFFF", "3.0"}};

    public static void main(String[] args) throws Exception {
        String data = args.length > 0 ? args[0] : "../data";
        System.out.println("SUS per participant");
        List<Double> scores = new ArrayList<>();
        for (String[] r : rows(data + "/sus.csv")) {
            int[] a = new int[10];
            for (int i = 0; i < 10; i++) a[i] = Integer.parseInt(r[i + 2]);
            double s = sus(a);
            scores.add(s);
            System.out.printf("  %s  %-6s %5.1f%n", r[0], r[1], s);
        }
        double[] c = meanCi(scores);
        System.out.printf("SUS mean %.1f (SD %.1f), 95%% CI [%.1f, %.1f]%n",
            c[0], c[1], c[2], c[3]);
        double sum = 0;
        List<String[]> lite = rows(data + "/umux_lite.csv");
        for (String[] r : lite)
            sum += (Integer.parseInt(r[1]) - 1 + Integer.parseInt(r[2]) - 1)
                / 12.0 * 100;
        double m2 = sum / lite.size();
```

Kode 16.5 (Lanjutan) *Evaluate.java: evaluasi yang sama dalam Java.*

```
        System.out.printf("UMUX-Lite mean %.1f, SUS-comparable %.1f%n",
            m2, 0.65 * m2 + 22.9);
        System.out.println(
            "\ntask  done  completion  95% CI (adj. Wald)  geo. mean time");
        Map<String, List<int[]>> tasks = new TreeMap<>();
        for (String[] r : rows(data + "/tasks.csv"))
            tasks.computeIfAbsent(r[1], k -> new ArrayList<>()).add(
                new int[] {Integer.parseInt(r[2]), Integer.parseInt(r[3])});
        for (var e : tasks.entrySet()) {
            int ok = 0, n = e.getValue().size();
            double logs = 0;
            for (int[] t : e.getValue())
                if (t[0] == 1) { ok++; logs += Math.log(t[1]); }
            double[] ci = adjustedWald(ok, n);
            System.out.printf(
                "%-4s  %d/%d  %9.0f%%  [%3.0f%%, %3.0f%%]  %14.1f s%n",
                e.getKey(), ok, n, 100.0 * ok / n, 100 * ci[0], 100 * ci[1],
                Math.exp(logs / ok));
        }
        System.out.println("\ncolor pair      ratio  needed  verdict");
        for (String[] p : PAIRS) {
            double ratio = contrast(p[1], p[2]), need = Double.parseDouble(p[3]);
            System.out.printf("%-14s %6.2f  %6.1f  %s%n", p[0], ratio, need,
                ratio >= need ? "pass" : "FAIL");
        }
    }
}
```

16.7.4 Hasilnya

Kedua program mencetak:

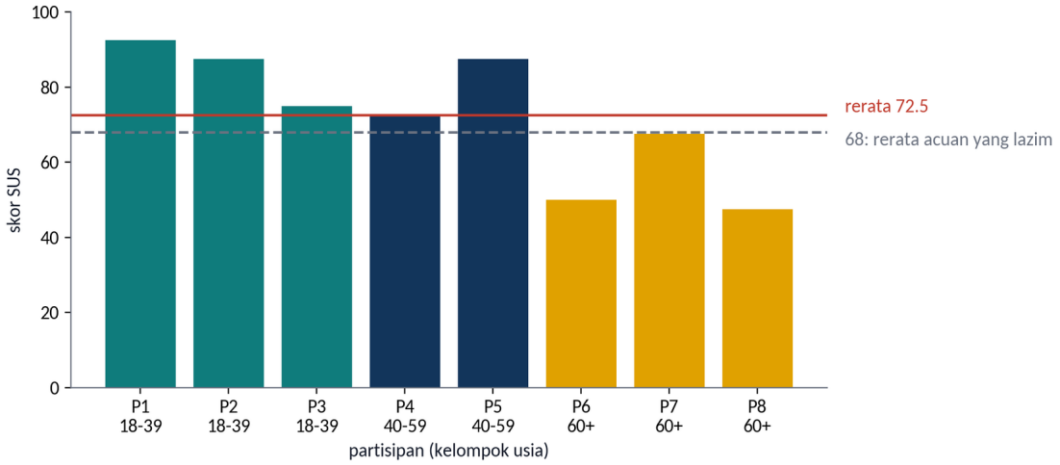
```
SUS per participant
  P1  18-39   92.5
  P2  18-39   87.5
  P3  18-39   75.0
  P4  40-59   72.5
  P5  40-59   87.5
  P6  60+     50.0
  P7  60+     67.5
  P8  60+     47.5
SUS mean 72.5 (SD 17.0), 95% CI [58.3, 86.7]
UMUX-Lite mean 79.2, SUS-comparable 74.4

task  done  completion  95% CI (adj. Wald)  geo. mean time
T1    8/8        100%  [ 63%, 100%]            12.7 s
T2    6/8         75%  [ 40%,  94%]            26.0 s
T3    7/8         88%  [ 51%, 100%]            22.8 s

color pair      ratio  needed  verdict
button text     12.46     4.5  pass
body text       17.74     4.5  pass
tool links       7.56     4.5  pass
urgent note     15.51     4.5  pass
focus, draft     2.27     3.0  FAIL
focus, fixed    12.46     3.0  pass
```

Gambar 16.4 menggambar skor SUS per partisipan dan kelompok usia. Reratanya, 72,5, di atas rerata acuan 68, dan UMUX-Lite sejalan dengan nilai setara SUS 74,4. Namun selang 95%-nya membentang dari 58,3 sampai 86,7: dengan delapan partisipan, data tidak dapat memastikan apakah kios itu biasa saja atau sangat baik. Pola di dalam data lebih jelas daripada reratanya. Ketiga partisipan berusia 60 tahun ke atas memberi skor 50,0, 67,5, dan 47,5; semua yang lebih

muda memberi 72,5 atau lebih. Bagi pengguna yang paling dipedulikan kepala puskesmas, kios ini belum cukup baik.



Gambar 16.4 Skor SUS dalam uji rintis, diwarnai menurut kelompok usia, dengan rerata (72,5) dan rerata acuan yang lazim, 68; ketiga partisipan berusia 60 tahun ke atas berada di bawah acuan.

Tugas-tugasnya menunjukkan di mana masalahnya. Semua orang menyelesaikan T1, mengambil nomor, dengan rerata geometrik 12,7 detik. T2, berganti bahasa, gagal bagi dua dari delapan orang, keduanya lansia, dan memakan waktu dua kali lipat; dalam catatan berpikir keras, keduanya berkata mereka "tidak melihat" tautan bahasa, teks kecil bergaris bawah di bagian atas. T3, huruf besar, gagal bagi satu partisipan lansia yang menemukan tautannya tetapi tidak menyadari bahwa halamannya telah berubah. Pemeriksaan kontras menemukan satu kegagalan, cincin fokus kuning tua dengan 2,27:1 terhadap tuntutan 3:1, yang sudah diperbaiki dalam kode di atas. Polanya adalah pelajaran Bab 2 dan Bab 11 sekaligus: kontrol yang kecil dan jauh dari tindakan utama tidak terlihat oleh orang yang mata dan perhatiannya sudah terentang.

16.8 Iterasi: dari Temuan ke Perubahan

Temuan-temuan itu menjadi daftar berprioritas. **Tabel 16.4** menilai masing-masing dengan skala keparahan Nielsen, dari 0 (bukan masalah) sampai 4 (bencana usabilitas) [15], dengan menggabungkan berapa banyak pengguna yang terdampak, seberapa parah, dan apakah mereka dapat mengakalinya.

Tabel 16.4 Temuan uji rintis dengan penilaian keparahan dan perubahan yang direncanakan untuk iterasi kedua.

Temuan	Bukti	Keparahan	Perubahan untuk iterasi 2
Tautan bahasa tidak terlihat oleh lansia	T2: 2 dari 8 gagal, keduanya 60+; berpikir keras	3	Dua tombol besar "Bahasa Indonesia" dan "English" di atas layanan, seukuran tombol layanan
Perubahan huruf besar tidak disadari	T3: 1 kegagalan; "tidak terjadi apa-apa"	2	Tawarkan huruf besar bagi siapa pun dengan sekali sentuh; tambahkan satu baris konfirmasi
SUS lansia di bawah rerata acuan	50,0; 67,5; 47,5	3	Ditangani dua perubahan di atas; uji ulang dengan setidaknya lima lansia
Kontras cincin fokus 2,27:1	Pemeriksaan kontras	2	Sudah diperbaiki: cincin biru tua, 12,46:1
Sebagian pasien takut menyentuh kios	Temuan riset 2	2	Relawan atau petugas di dekat kios pada bulan pertama; papan "Loket tetap buka seperti biasa"

Dua hal tentang daftar ini sama pentingnya dengan isinya. Ia mengikat setiap perubahan pada bukti, sehingga tak seorang pun mengubah kios karena selera pribadi. Dan ia diakhiri dengan *uji ulang*: iterasi kedua akan dievaluasi dengan tugas yang sama dan setidaknya lima lansia, dan keberhasilan perubahan akan dinilai dari penyelesaian T2 dan T3 bagi kelompok itu, bukan dari rerata keseluruhan. Bab 14 akan menambahkan satu hal lagi begitu kios dipakai sehari-hari: hitungan anonim dari `/status.json` per jam dan per layanan, dibandingkan dengan kertas nomor dari loket, menunjukkan apakah kios benar-benar memperpendek antrean, yang merupakan tujuan awal kepala puskesmas.

16.9 Menguji Kode

Aritmetika evaluasi harus benar sebelum hasilnya dipercaya. **Kode 16.6** memeriksa nilai ekstrem SUS, kontras hitam di atas putih (tepat 21:1), dan bahwa selang Wald yang disesuaikan tetap punya batas bawah pada 8 dari 8.

Kode 16.6 `test_evaluate.py`: uji penskoran SUS, kontras, dan selang Wald yang disesuaikan.

```
"""Pemeriksaan aritmetika evaluasi; jalankan: python -m unittest test_evaluate"""
import unittest

from evaluate import adjusted_wald, contrast, sus


class EvaluateTest(unittest.TestCase):
    def test_sus(self):
        self.assertEqual(sus([5, 1] * 5), 100)
        self.assertEqual(sus([3] * 10), 50)

    def test_contrast(self):
        self.assertAlmostEqual(contrast("#000000", "#FFFFFF"), 21)

    def test_adjusted_wald(self):
        lo, hi = adjusted_wald(8, 8)
        self.assertEqual(hi, 1.0)
        self.assertGreater(lo, 0.6)


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

Kode 16.7 adalah versi Java-nya.

Kode 16.7 `EvaluateTest.java`: pemeriksaan yang sama dalam Java.

```java
/** Pemeriksaan aritmetika evaluasi; jalankan: java EvaluateTest */
public class EvaluateTest {
    static int failed = 0;

    static void check(boolean ok, String what) {
        System.out.println((ok ? "ok   " : "FAIL ") + what);
        if (!ok) failed++;
    }

    public static void main(String[] args) {
        check(Evaluate.sus(new int[] {5, 1, 5, 1, 5, 1, 5, 1, 5, 1}) == 100,
            "best answers give SUS 100");
        check(Evaluate.sus(new int[] {3, 3, 3, 3, 3, 3, 3, 3, 3, 3}) == 50,
            "neutral answers give SUS 50");
        check(Math.abs(Evaluate.contrast("#000000", "#FFFFFF") - 21) < 1e-9,
            "black on white is 21:1");
        double[] ci = Evaluate.adjustedWald(8, 8);
        check(ci[1] == 1.0 && ci[0] > 0.6, "8 of 8 still has a lower bound");
        System.out.println(failed == 0 ? "all checks passed" : failed + " failed");
        System.exit(failed == 0 ? 0 : 1);
    }
}
```

`python -m unittest test_evaluate` melaporkan tiga uji yang lulus; versi Java mencetak:

```
ok   best answers give SUS 100
ok   neutral answers give SUS 50
ok   black on white is 21:1
ok   8 of 8 still has a lower bound
all checks passed
```

Server kios diuji dengan cara lain: dengan meminta halaman yang sama dari keduanya dan membandingkan HTML-nya bita demi bita, dengan mengirim kode layanan yang tidak sah dan memeriksa bahwa jawabannya berupa pengalihan (302 dari Flask, 303 dari Java), dan dengan skrip Playwright, yang gagal bila tombol yang diharapkannya tidak ada.

16.10 Menyajikan Keputusan Desain

Desain hanya sekuat argumen yang dapat diperiksa orang lain. Di akhir semester tim mempresentasikannya kepada kepala puskesmas, Mbak Ratna, dan dinas kesehatan kabupaten. Laporan mereka mengikuti struktur yang berlaku untuk proyek akhir apa pun:

1. **Masalah dan orang-orangnya**, dalam satu halaman: antrean di loket, empat temuan, dua persona.
2. **Kebutuhan**, masing-masing ditelusuri ke sebuah temuan dan sebuah bab, seperti di Subbab 16.4.
3. **Desainnya**, diperlihatkan dengan kios yang berjalan dan tangkapan layarnya, setiap keputusan diikat pada sebuah kebutuhan.
4. **Buktinya**: uji rintis, dengan ukuran sampel yang dinyatakan jujur, hasil di Subbab 16.7.4 termasuk yang mengecewakan, dan pemeriksaan kontras.
5. **Apa yang berubah berikutnya**, daftar berprioritas pada **Tabel 16.4**, dan cara uji berikutnya akan menilainya.
6. **Apa yang sengaja tidak dilakukan**, seperti triase AI, perkiraan waktu tunggu, dan data pribadi, beserta alasannya masing-masing.

Butir keenam sering lebih meyakinkan pendengar yang skeptis daripada butir lainnya. Ia menunjukkan bahwa tim telah menimbang pilihan-pilihan yang sedang tren dan menolaknya

dengan alasan yang terikat pada pengguna, dan itulah beda antara sebuah desain dan sebuah peragaan.

16.11 Memakainya Sungguhan

Untuk memasang kios di puskesmas sungguhan, tim mendaftar apa yang membedakan prototipe dari layanan. Tablet menjalankan peramban dalam *mode kios*, yang menyembunyikan bilah alamat dan mencegah orang meninggalkan halaman, dan dipasang pada ketinggian yang dapat dijangkau dari kursi roda. Sebuah printer termal kecil ditambahkan agar nomor dapat dibawa di atas kertas, karena nomor di layar lenyap ketika pasien berikutnya menyentuhnya; printer digerakkan oleh server ketika `/take` mengeluarkan nomor. Hitungan antrean disimpan dalam basis data kecil dan diatur ulang setiap pagi, dan aplikasi loket memanggil endpoint `/call` untuk menandai nomor yang sudah dilayani (Soal 6). Endpoint status mengisi televisi ruang tunggu, yang nomornya diukur dengan metode Bab 15. Akhirnya, server pengembangan Flask diganti server produksi, seperti diperingatkan Flask sendiri, dan kios dijalankan ulang otomatis bila berhenti.

Pendekatan yang sama melayani tempat lain tempat orang mengambil nomor: kantor desa, kantor cabang bank, kantor pos, kantor Samsat. Keempat layanan berubah, kata-katanya berubah, dan uji harus diulang bersama orang-orang di tempat itu. Prosesnya, dan aturan bahwa tak seorang pun boleh lebih dirugikan daripada sebelumnya, tidak berubah.

16.12 IMK dalam Praktik: Profitina

Profitina, platform AI Visibility (GEO) di app.profitina.com, dibangun dan dirilis dalam siklus yang sama dengan kios ini, pada skala yang lebih besar. Pertanyaan risetnya adalah bagaimana pemilik merek yang bukan insinyur dapat memahami mengapa mesin jawaban AI mengutip atau tidak mengutip situs mereknya, dan jawaban desainnya adalah dasbor yang menjelaskan setiap sinyal terukur dengan kata-kata sederhana, bukan satu angka yang buram. Kebutuhannya mencakup kepatuhan UU PDP 27/2022 yang dapat dijalankan pengguna sendiri: persetujuan eksplisit, ekspor data, penggantian surel dan kata sandi, serta penghapusan akun sepenuhnya, semuanya di dalam produk, bukan lewat surel ke bagian dukungan. AI-nya berjalan lokal, dan bila tidak ada model yang tersedia aplikasi mengatakannya, yaitu prinsip kejujuran Bab 13 dan penolakan bab ini untuk menampilkan perkiraan waktu tunggu.

Evaluasinya berlangsung terus-menerus. Rangkaian uji otomatis yang lengkap, jauh lebih dari seribu uji termasuk penjaga JavaScript untuk antarmuka, harus lulus sebelum rilis apa pun, dan rilis bertahap baru melaporkan keberhasilan setelah versi yang dipasang menjawab pemeriksaan kesehatannya. Umpan balik pengguna mengalir melalui alur yang transparan (kirim, kurasi, dataset, latih dan ukur, rilis) yang statusnya dapat dilihat setiap pengguna di dasbor. Siklus pada **Gambar 16.1**, menemukan, merumuskan, mengembangkan, menyerahkan, dan kembali lagi, sama bagi kios puskesmas buatan tiga mahasiswa dan bagi produk yang dipakai merek-merek di seluruh Indonesia.

16.13 Kesalahan yang Sering Terjadi, dan Cara Menghindarinya

- **Memulai dari solusi.** “Kita akan membangun kios” datang dari kepala puskesmas; riset bisa saja menunjukkan bahwa loket kedua pada jam sibuk lebih baik. Pisahkan tujuan (antrean lebih pendek, tak seorang pun dirugikan) dari solusi yang diusulkan sampai riset selesai.

- **Merancang untuk pengguna rata-rata.** Rerata SUS 72,5 tampak baik dan menyembunyikan skor para lansia. Laporkan hasil menurut kelompok yang penting, seperti dilakukan **Gambar 16.4**.
- **Menganggap uji kecil itu presisi.** Delapan partisipan menemukan masalah; mereka tidak mengukur rerata dengan presisi. Selalu laporkan selangnya, seperti [58.3, 86.7] untuk SUS.
- **Memeriksa aksesibilitas di akhir.** Kegagalan cincin fokus murah diperbaiki karena ditemukan lebih awal. Masukkan kriteria WCAG ke dalam kebutuhan (**Tabel 16.2**) dan periksalah dengan kode.
- **Menambah teknologi karena tersedia.** Triase AI atau pengenalan wajah akan menambah risiko dan data tanpa menolong persona mana pun. Tuliskan apa yang Anda putuskan untuk tidak dilakukan, dan mengapa.
- **Mengumpulkan data yang tidak diperlukan.** Nomor antrean tidak memerlukan nama. Setiap kolom yang tidak Anda kumpulkan adalah layar persetujuan yang tidak Anda perlukan dan kebocoran yang tidak mungkin terjadi.
- **Berhenti setelah satu iterasi.** Uji yang menemukan masalah baru berhasil bila masalahnya diperbaiki dan diuji ulang. Rencanakan iterasi kedua sebelum menyajikan yang pertama.
- **Hanya menyajikan kabar baik.** Pengambil keputusan memercayai tim yang memperlihatkan hasil yang mengecewakan dan apa yang akan dilakukan terhadapnya.

16.14 Ringkasan Bab

- Sistem interaktif yang utuh mencakup layanan di sekitar layar: tempatnya, orang yang tidak memakainya, petugas, dan aturannya. ISO 9241-210 menggambarkan desainnya sebagai siklus memahami konteks, merinci kebutuhan, merancang, dan mengevaluasi [1]; Double Diamond memberi jalur yang sama dalam empat fase (**Gambar 16.1**) [2].
- Setiap bab buku ini menyumbang satu keputusan bagi kios antrean (**Tabel 16.1**), termasuk keputusan untuk tidak memakai AI di tempat tebakan dapat mencelakakan.
- Riset di loket menghasilkan empat temuan, dua persona, dan satu skenario; kebutuhan ditulis sebagai pernyataan yang dapat diuji dan ditelusuri ke temuan.
- Hukum Hick (Persamaan (16.1)) mempertahankan pilihan pada empat layanan, dan hukum Fitts (Persamaan (16.2)) membenarkan tombol besar selebar layar [3], [4]; kriteria sukses WCAG 2.2 menjadi kebutuhan sejak awal (**Tabel 16.2**) [5], demikian pula hukum aksesibilitas Indonesia [6], [7] dan minimisasi data [9].
- Kios dibangun sebagai aplikasi Flask dan sebagai server HTTP JDK yang menyajikan HTML identik, dengan dua bahasa, mode huruf besar, tautan lompat, pesan status, dan pengalihan bantuan darurat, lalu ditangkap dengan Playwright (**Gambar 16.2**, **Gambar 16.3**).
- Uji rintis dengan delapan partisipan mengukur SUS (Persamaan (16.3)), UMUX-Lite (Persamaan (16.4)), penyelesaian tugas dengan selang Wald yang disesuaikan (Persamaan (16.5)), dan kontras warna (Persamaan (16.6)): SUS 72,5 dengan selang lebar [58.3, 86.7], semua lansia di bawah 68, berganti bahasa sebagai tugas terlemah, dan satu kegagalan kontras yang sudah diperbaiki.
- Temuan-temuan menjadi daftar berprioritas dengan tingkat keparahan (**Tabel 16.4**) dan uji ulang terencana bersama lansia; presentasinya memperlihatkan bukti, kekecewaan, dan pilihan yang sengaja ditolak.

16.15 Soal Latihan

1. Hitung skor SUS partisipan P6 dengan tangan menggunakan Persamaan (16.3), dari jawaban 3, 3, 3, 3, 3, 2, 3, 3, 3, 4. Lalu sebutkan dua butir yang paling menurunkan skornya.
2. Hitung selang Wald yang disesuaikan untuk T2 (6 berhasil dari 8) dengan tangan menggunakan Persamaan (16.5), lalu bandingkan dengan selang naif $\hat{p} \pm 1.96\sqrt{\hat{p}(1-\hat{p})/n}$ dengan $\hat{p} = 0.75$. Mengapa selang yang disesuaikan lebih disukai di sini?
3. Hitung rasio kontras warna tautan alat `#0B5E5E` di atas putih dengan Persamaan (16.6), dengan memperlihatkan kanal-kanal yang dilinearkan, dan pastikan angka 7,56 yang dicetak `evaluate.py`.
4. Seorang manajer meminta delapan layanan di layar pertama, bukan empat. Pakai Persamaan (16.1) dan Persamaan (16.2) untuk menjelaskan biayanya, lalu usulkan desain yang melayani keperluan tambahan tanpa merugikan tugas utama.
5. Rencanakan uji iterasi kedua: partisipan, tugas, ukuran, dan kriteria yang akan menentukan apakah tombol bahasa yang baru berhasil. Berapa lansia yang akan Anda rekrut, dan mengapa?
6. Tambahkan endpoint `POST /call` dengan isian formulir `service` yang menandai nomor berikutnya yang sedang menunggu pada layanan itu sebagai sudah dilayani, di `kiosk_app.py` dan `KioskServer.java`. Apa yang harus dilakukannya bila tak ada yang menunggu, dan bagaimana "Orang di depan Anda" berubah?

16.16 Penyelesaian Soal Latihan

1. Butir ganjil 1, 3, 5, 7, 9 semuanya 3, menyumbang $5 \times (3-1) = 10$. Butir genap 2, 4, 6, 8, 10 bernilai 3, 3, 2, 3, 4, menyumbang $(5-3)+(5-3)+(5-2)+(5-3)+(5-4) = 2+2+3+2+1 = 10$. SUS $= 2.5 \cdot 20 = 50$, sama dengan cetakan `evaluate.py`. Yang paling merugikan adalah butir 10 ("Saya perlu mempelajari banyak hal sebelum dapat memakai sistem ini"), dijawab 4, dan butir-butir positif yang semuanya netral: tak ada yang jelas-jelas mudah bagi partisipan ini, sejalan dengan kegagalannya pada T2.

2. $z^2 = 3.8416$. $\hat{p} = (6 + 1.9208)/(8 + 3.8416) = 7.9208/11.8416 = 0.6689$. Setengah lebar $= 1.96\sqrt{0.6689 \cdot 0.3311/11.8416} = 1.96 \cdot 0.1368 = 0.268$, jadi selangnya [0.401,0.937], tercetak sebagai [40%, 94%]. Selang naifnya $0.75 \pm 1.96\sqrt{0.75 \cdot 0.25/8} = 0.75 \pm 0.300$, yaitu [0.45,1.05], yang melewati 100% dan cakupan sesungguhnya terlalu sempit untuk sampel kecil. Selang yang disesuaikan tetap di dalam 0 sampai 1 dan cakupannya mendekati 95% yang dijanjikan bahkan dengan delapan partisipan [13].

3. Merah: $0x0B = 11$, $c = 0.0431 > 0.04045$, jadi $c_{\text{lin}} = ((0.0431 + 0.055)/1.055)^{2.4} = 0.0930^{2.4} \approx 0.00335$. Hijau dan biru: $0x5E = 94$, $c = 0.3686$, $c_{\text{lin}} = (0.4015)^{2.4} \approx 0.1119$. $L = 0.2126 \cdot 0.00335 + 0.7152 \cdot 0.1119 + 0.0722 \cdot 0.1119 = 0.0007 + 0.0800 + 0.0081 = 0.0888$. Putih memiliki $L = 1$. Rasio $= (1 + 0.05)/(0.0888 + 0.05) = 1.05/0.1388 = 7.56$, jauh di atas 4,5.

4. Hick: $\log_2 9 = 3.17$ bit alih-alih $\log_2 5 = 2.32$, sekitar 37% lebih banyak informasi yang harus diproses *setiap* pasien, padahal kebanyakan membutuhkan salah satu dari empat poli. Fitts: delapan tombol di layar yang sama harus kira-kira setengah tingginya, 32 piksel, menaikkan indeks kesulitan dari 2,51 menjadi 3,37 bit untuk tombol terjauh dan mendekati batas minimum 24 piksel WCAG 2.2. Desain yang lebih baik mempertahankan empat poli sebagai layar pertama dan menambahkan satu tombol yang lebih kecil dan berlabel jelas "Layanan lain dan pertanyaan" di bawah, yang mengeluarkan nomor loket; keperluan tambahan terlayani, dan tugas utama tidak terganggu.

5. Partisipan: setidaknya lima berusia 60 tahun ke atas, karena itulah kelompok yang masalahnya disasar perubahan, ditambah tiga partisipan lebih muda untuk memastikan tak ada yang memburuk bagi mereka; setidaknya dua lansia yang datang sendiri. Tugas: T1–T3 yang sama, agar hasilnya dapat dibandingkan. Ukuran: penyelesaian dan waktu per tugas menurut kelompok usia, SUS dan UMUX-Lite, serta catatan berpikir keras. Kriteria: tombol bahasa yang baru berhasil bila setidaknya empat dari lima lansia menyelesaikan T2 tanpa bantuan dan tak seorang pun berkata tidak melihat pilihan bahasa. Dengan lima partisipan, selangnya tetap lebar, jadi kriterianya tentang masalah yang ditemukan, bukan tentang laju yang presisi.

6. Di Python, tambahkan rute `@app.post("/call")` yang membaca `code = request.form.get("service", "")`, memeriksa `code in queues`, dan menaikkan `served[code]` hanya bila `served[code] < queues[code]`; rute itu mengembalikan JSON status agar loket melihat hitungan yang baru. Di Java, tambahkan cabang untuk jalur `/call` dengan metode POST yang melakukan hal sama dengan `served.merge(code, 1, Integer::sum)` setelah pemeriksaan yang sama. Bila tak ada yang menunggu, endpoint itu tidak boleh berbuat apa-apa dan harus mengatakannya (misalnya `{"called": null}`), dan tidak pernah membiarkan `served` melampaui `given`. Pada halaman `/take`, "Orang di depan Anda" dihitung sebagai `queues[code] - served[code] - 1`, jadi setiap panggilan menguranginya satu bagi setiap orang yang mengambil nomor sesudahnya; pasien yang sudah memegang nomor melihat perubahannya hanya di televisi ruang tunggu, yang membaca `/status.json`.

Daftar Pustaka

[1] ISO 9241-210:2019, *Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. Geneva, Switzerland: International Organization for Standardization, 2019.

[2] Design Council, "Framework for Innovation: Design Council's evolved Double Diamond," 2019. [Online]. Available: https://www.designcouncil.org.uk/our-resources/framework-for-innovation/

[3] W. E. Hick, "On the rate of gain of information," *Quarterly Journal of Experimental Psychology*, vol. 4, no. 1, pp. 11–26, 1952.

[4] I. S. MacKenzie, "Fitts' law as a research and design tool in human-computer interaction," *Human-Computer Interaction*, vol. 7, no. 1, pp. 91–139, 1992.

[5] W3C, "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, Oct. 5, 2023; disetujui sebagai ISO/IEC 40500:2025, lihat W3C WAI, "WCAG 2.2 approved as an ISO standard," Oct. 21, 2025. [Online]. Available: https://www.w3.org/TR/WCAG22/ and https://www.w3.org/WAI/news/2025-10-21/wcag22-iso

[6] Republik Indonesia, *Undang-Undang Nomor 8 Tahun 2016 tentang Penyandang Disabilitas*. Jakarta, 2016.

[7] Republik Indonesia, *Peraturan Pemerintah Nomor 42 Tahun 2020 tentang Aksesibilitas terhadap Permukiman, Pelayanan Publik, dan Pelindungan dari Bencana bagi Penyandang Disabilitas*. Jakarta, 2020.

[8] European Union, "Directive (EU) 2019/882 on the accessibility requirements for products and services (European Accessibility Act)," *Official Journal of the European Union*, L 151, Jun. 7, 2019; berlaku sejak Jun. 28, 2025.

[9] Bisnis.com, "Aturan Turunan UU PDP: Persetujuan Data Konsumen Lebih Ketat," Aug. 29, 2026, tentang Peraturan Pemerintah No. 33 Tahun 2026 sebagai pelaksana UU No. 27 Tahun 2022. [Online]. Available: https://teknologi.bisnis.com/read/20260829/84/1999753/aturan-turunan-uu-pdp-persetujuan-data-konsumen-lebih-ketat

[10] Microsoft, "Playwright for Python," dokumentasi. [Online]. Available: https://playwright.dev/python/ (diakses Sep. 2026).

[11] J. Nielsen and T. K. Landauer, "A mathematical model of the finding of usability problems," in *Proc. INTERCHI '93*, Amsterdam, The Netherlands, 1993, pp. 206–213.

[12] J. Brooke, "SUS: A 'quick and dirty' usability scale," in *Usability Evaluation in Industry*, P. W. Jordan *et al.*, Eds. London, UK: Taylor & Francis, 1996, pp. 189–194.

[13] J. Sauro and J. R. Lewis, *Quantifying the User Experience: Practical Statistics for User Research*, 2nd ed. Cambridge, MA, USA: Morgan Kaufmann, 2016.

[14] J. R. Lewis, B. S. Utesch, and D. E. Maher, "UMUX-LITE: When there's no time for the SUS," in *Proc. CHI '13*, Paris, France, 2013, pp. 2099–2102.

[15] J. Nielsen, "Severity ratings for usability problems," Nielsen Norman Group, Nov. 1, 1994. [Online]. Available: https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/