Lecture 1 — Introduction to HCI: Why Interfaces Matter

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 1

Lecturer: Irfan Subakti — yifana@gmail.com

Today we meet the discipline, see why it matters in Indonesia in 2026, set up the workbench, and turn a design argument into numbers. Book: Chapter 1, Sections 1.1–1.16.

What This Meeting Covers

1. **Why interfaces matter** — Bu Ratna's warung page, and the same story at the puskesmas, the ATM, the ojek base and the laundry.
2. **What HCI is** — usability, user experience and accessibility in the words of ISO; Norman's two gulfs.
3. **HCI in 2026** — new design languages, AI agents, CHI 2026, and the Indonesian users and laws behind the screens.
4. **The workbench** — JDK 25, Python 3.14 and VS Code on Windows, macOS and Linux.
5. **A first HCI program** — the Keystroke-Level Model predicts how many seconds a better screen saves.

Every program today exists in Java and in Python, and every output shown is the output the program really printed. Book: Chapter 1.

Where We Begin: Bu Ratna’s Warung Goes Online

Dimas builds his aunt an order page: a dropdown of 42 menu items, a quantity box, *Add*, and *Save*.

Every order is stored correctly, yet on the first Saturday she returns to her notebook by eleven o’clock.

One order means: open the dropdown, scroll, click, click the box, type, move the hand back, click *Add*, repeat.

Nothing was broken except the fit between the page and the person using it under pressure. Book: Section 1.1.

What Changed Was the Interface

Dimas watches for an hour and redraws the page as **large tiles**, most frequent items first; a click adds a portion.

The same order drops from more than twenty seconds to about ten, and Bu Ratna uses it without being asked.

The database, the network and the language did not change. Only the **interface** did.

The interface is the part of a system that a person sees, hears and touches, and through which the person tells the system what to do. Book: Section 1.1.

The Same Story, Everywhere

A **puskesmas kiosk** with twelve small labels, and an elderly patient looking for the dentist.

An **ATM** that once kept the card of everyone who walked away with the cash.

An **ojek app** whose pickup pin lands across the river; a **laundry** WhatsApp menu where nobody remembers which number means “iron only”.

The book teaches how to see such problems before users do, how to design better, and how to prove the result is better. Book: Section 1.1.

What Human-Computer Interaction Is

ACM SIGCHI: *a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use.*

Design before building, **evaluation** with real people and tasks, **implementation** in code, success measured by **human use**.

It draws on computer science, psychology, design, sociology, ergonomics and linguistics.

In Indonesian universities the course is called Interaksi Manusia dan Komputer (IMK). Book: Section 1.2.1.

Three Words with Precise Meanings

Usability (ISO 9241-11:2018): effectiveness, efficiency and satisfaction for *specified* users, goals and contexts of use.

User experience (ISO 9241-210:2019): perceptions and responses before, during and after use.

Accessibility: usable by people with the widest range of needs and capabilities, including people with disabilities.

The standards turn vague praise (“user-friendly”) into things that can be checked. Book: Section 1.2.1.

“Specified”: Usable for Whom?

Dimas’s first page is perfectly usable by Dimas at his desk.

It fails for a specified user (Bu Ratna, a slow mouse user), a specified goal (a three-item order) and a specified context (twenty hungry students).

Every method in this course starts by asking **who**, **what for** and **where**.

Change the user, the goal or the context and the verdict can change. Book: Section 1.2.1.

Measuring the Three Qualities

Effectiveness: the share of orders entered without a mistake.

Efficiency: seconds per order.

Satisfaction: a short questionnaire such as the System Usability Scale (Chapter 7).

User experience is wider still: Bu Ratna's worry on Friday night and her small pleasure when the queue is empty. Book: Section 1.2.1.

Norman's Two Gulfs

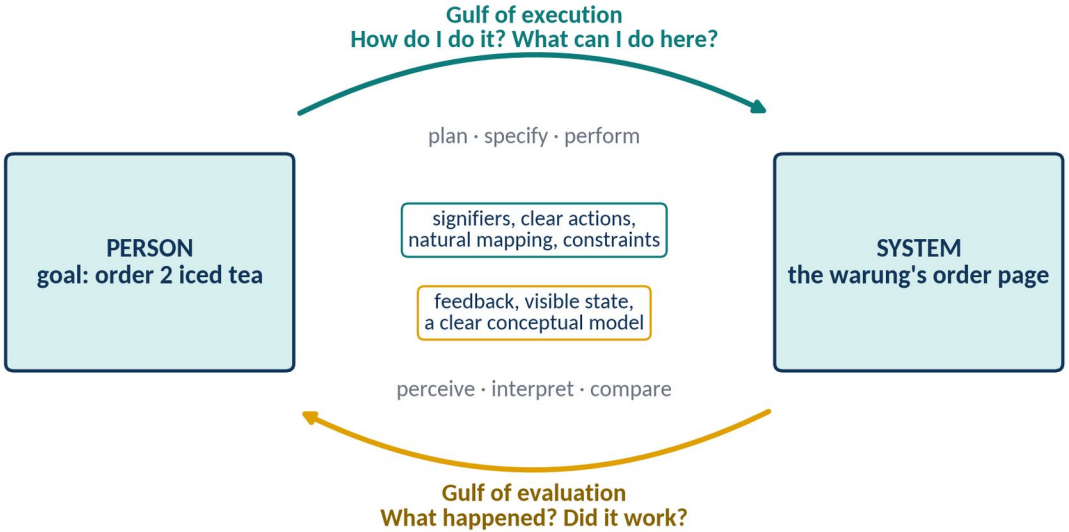


Figure 1.1 Norman's two gulfs: execution (how do I do it?) and evaluation (did it work?), and the design qualities that bridge each one.

Figure 1.1: the gulf of execution separates the goal from the actions the system offers; the gulf of evaluation separates what the system shows from what the person needs to know. Book: Section 1.2.2.

Bridging the Gulfs

Execution: signifiers that show what can be done, natural mapping, constraints that block wrong actions.

Evaluation: immediate feedback, visible state, a clear conceptual model.

Bu Ratna clicked *Add*, saw nothing change, clicked again: the kitchen made four iced teas.

Banks bridged the ATM's gulf of evaluation by returning the card before the cash. Book: Section 1.2.2.

Blame the Gulf, Not the User

When a person makes a mistake, ask **which gulf was too wide?**, not what is wrong with the person.

Blaming users changes nothing; narrowing a gulf changes the behaviour of every future user.

This one habit separates HCI from "she will get used to it". Book: Section 1.2.2.

What HCI Is Not

Not **making things pretty**: a beautiful page that hides its main action has failed.

Not **asking users what they want**: people predict their own behaviour poorly; observe them (Chapter 5).

Not **common sense**: what is obvious to the builder is exactly what the builder can no longer see.

The discipline replaces the builder's blind spot with evidence. Book: Section 1.2.3.

Who the User Was, Era by Era

Table 1.1 The eras of human-computer interaction and what each asked of its users.

Era	Typical interface	What the user had to learn
1950s–1970s	Batch jobs, command lines	Exact command syntax
1980s–1990s	Windows, icons, menus, pointer	Desktop metaphors
1990s–2000s	The web	Links, forms, search
2007–2020s	Touchscreens, mobile apps	Gestures, app conventions
2020s	Voice, generative AI, agents, spatial	What to ask, when to trust

Table 1.1: the history of HCI is the history of who the user is, from trained specialists to almost everyone. Book: Section 1.3.

2025–2026: The Design Languages Were Renewed

Liquid Glass (Apple, June 9, 2025): a translucent material across iOS 26 and its siblings.

Material 3 Expressive (Google, 2025): 46 studies, more than 18,000 participants; key elements found up to four times faster.

Fluent 2: Microsoft's shared design system for Windows and Microsoft 365.

These are applications of principles, not fashions to copy; Chapter 3 shows where they apply and where they strain the principles. Book: Section 1.3.

Beyond the Flat Screen, and AI as an Interface

Apple Vision Pro (February 2024) made **spatial computing** mainstream; watches, earbuds and cars have their own interfaces.

ChatGPT reported **900 million weekly users** in February 2026; copilots and **agents** act on users' behalf.

The gulfs return: *what should I ask?* and *is this true, and what did the agent just do?*

Amershi et al.'s guidelines for human-AI interaction (CHI 2019) remain the starting point; Chapter 13 develops them. Book: Section 1.3.

A Research Community Larger Than Ever

CHI 2026, Barcelona, April 13–17, 2026: 1,702 full papers accepted from 6,730 submissions (25.3%), 5,254 participants on site.

Figma Config 2026 (June 23–24): an AI agent inside the canvas, code layers, motion design.

Tools change fast; this course uses them for prototyping (Chapter 6) but never lets them replace watching real people. Book: Section 1.3.

Who Is Online in Indonesia

APJII 2026: 235.2 million users, **81.72%** of 287.9 million people; from 85.95% in Java to 69.74% in Maluku and Papua. Of those still offline, 34% have no device and **31.5% lack digital literacy**.

The second number is an HCI number: a third of the unconnected are held back by interfaces that ask too much of first-time users. Book: Section 1.4.1.

On What Devices? Ask What a Statistic Counts

StatCounter, August 2026: Android **77.07%**, iOS 22.88% of mobile page views in Indonesia.

The same service: 60.59% of all page views from desktops, 38.98% from phones.

It counts **page views on tracked websites**, not people; most everyday use happens inside apps it cannot see.

Always read the method before designing on a number. Book: Section 1.4.1.

The Users Are Getting Older

BPS (SUPAS 2025): **11.97%** of Indonesians are sixty or older; Indonesia has entered an ageing population.

From 17.83% in Yogyakarta to 6.71% in Central Papua.

A screen designed by twenty-year-olds for twenty-year-olds fails a growing part of the country (Chapter 2).

The elderly are among the most frequent users of public services such as the puskesmas. Book: Section 1.4.1.

The Law Now Asks for Good Interfaces

UU 8/2016 on Persons with Disabilities: a right to accessible information and communication.

PP 42/2020: accessibility of public services for persons with disabilities.

UU PDP 27/2022: explicit, valid consent for specific purposes; **PP 33/2026**, signed July 16, 2026, in force January 16, 2027.

A consent screen that people cannot understand is bad design and a weak legal basis (Chapter 12). Book: Section 1.4.2.

Abroad, and the Technical Yardstick

The **European Accessibility Act** has applied since June 28, 2025 to e-commerce, banking, e-books and ticketing sold in the EU.

WCAG 2.2 is the current W3C Recommendation; **WCAG 3.0** is still a working draft (latest review draft September 2026), years from completion.

Chapter 11 works through WCAG 2.2 criterion by criterion. Book: Section 1.4.2.

How the Book Works

Everyday situation → idea in plain words → theory with evidence → hands-on method → companion program in Java and Python.

Then *HCI in Practice: Profitina*, common mistakes, summary, exercises with complete solutions, references.

Programs make HCI measurable: KLM (Ch. 1), Fitts and Hick (Ch. 2), heuristic reports (Ch. 3), affinity boards (Ch. 4).

The programs are deliberately small; they measure designs rather than build systems. Book: Section 1.4.3.

Your Workbench

Table 1.2 The tools of the workbench and the versions used throughout the book.

Tool	Version used
JDK (Temurin)	25 LTS, update 25.0.4
Python	3.14.7
Visual Studio Code	1.139
Extension Pack for Java, Python extension	from the Extensions view

Table 1.2: every program also runs on JDK 17+ and Python 3.11+, so older installations still work. Book: Section 1.5.

How Java and Python Run a Program

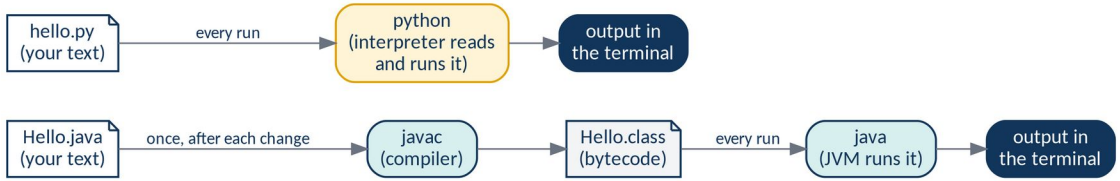


Figure 1.2 How a Java program and a Python program run: javac compiles once to bytecode for the JVM; Python reads and runs the file every time.

Figure 1.2: Java finds many mistakes while compiling; Python finds them when it reaches the faulty line. `java Hello.java` *runs a single file in one step. Book: Section 1.5.*

Installing on Windows

```
PS> winget install EclipseAdoptium.Temurin.25.JDK
PS> py install 3.14
PS> winget install Microsoft.VisualStudioCode
```

Open a **new** PowerShell after installing; check `java -version`, `javac -version` and `python --version`.

"not recognized" means the folder is not on the `PATH`: rerun the installer with **Add to PATH**.

Python on Windows now comes through the Python install manager. Book: Section 1.6.

Installing on macOS and Linux

```
you@mac ~ % brew install --cask temurin@25
$ sudo apt install openjdk-25-jdk python3 python3-venv python3-pip
```

macOS: the Python installer from python.org, then **Install Certificates.command** once; use `python3`.

Ubuntu 26.04 LTS ships OpenJDK 25 and Python 3.14; never remove the system Python on Linux.

Fedora: `sudo dnf install java-25-openjdk-devel python3 python3-pip`. *Book: Sections 1.7 and 1.8.*

A Tour of VS Code

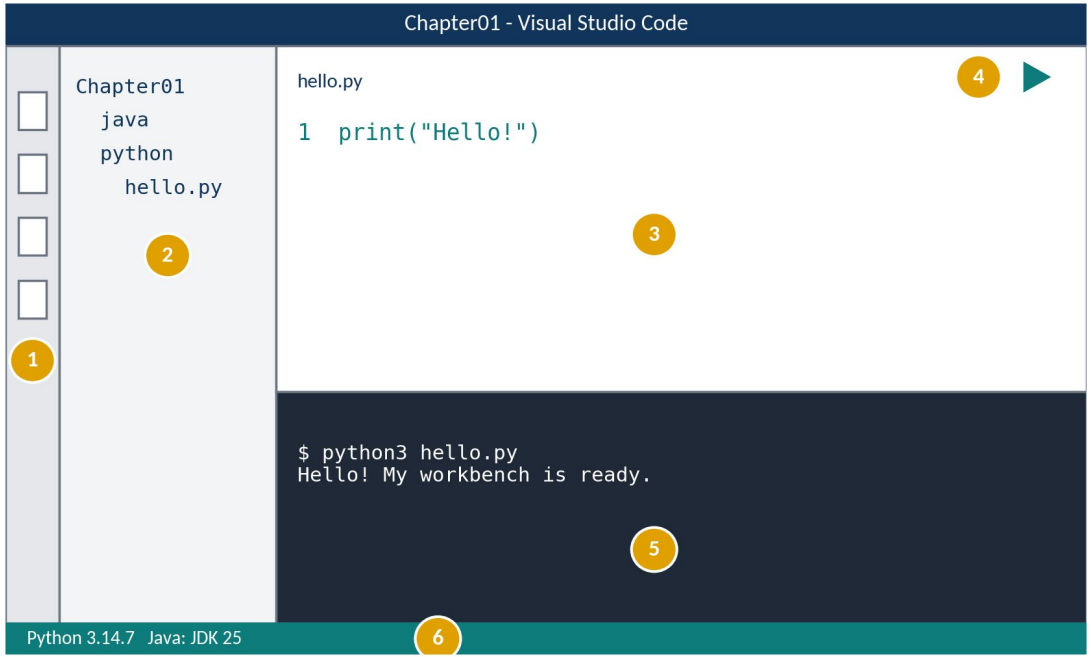


Figure 1.3 The parts of Visual Studio Code used in this book: (1) Activity Bar, (2) Explorer, (3) editor, (4) Run button, (5) integrated terminal, (6) Status Bar.

Figure 1.3: install the Extension Pack for Java and the Python extension, and always open a folder, not a single file. Book: Section 1.9.

First Programs, and a Check

```
$ 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
```

`Hello.java` and `hello.py` greet and print the versions; `CheckSetup.java` checks from the Java side. The last line is the one to read. Book: Section 1.10.

The Keystroke-Level Model

Table 1.3 The Keystroke-Level Model operators and their standard unit times.

Operator	Meaning	Time (s)
K	Press a key	0.28
P	Point with the mouse	1.10
B	Press or release the mouse button	0.10
H	Move a hand between keyboard and mouse	0.40
M	Prepare mentally	1.35

Table 1.3: Card, Moran and Newell (1980) predict an expert's time before anything is built; one M goes before each chunk that needs a decision. Book: Section 1.11.1.

Counting Steps, Adding Times

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

Count each kind of operator, multiply by its unit time, add.

Equation (1.1) is all there is to the model; the art lies in writing the task as operators. Book: Section 1.11.1.

Design A by Hand: the Dropdown Form

Per item: M, P B B (open list), P B B (choose), P B B (quantity box), H, K, H, P B B (*Add*).

Three items and *Save*: $n_M = 4$, $n_P = 13$, $n_B = 26$, $n_H = 6$, $n_K = 3$.

$T_A = 5.40 + 14.30 + 2.60 + 2.40 + 0.84 = 25.54$ s.

Pointing alone costs 14.30 seconds of the 25.54. Book: Section 1.11.2.

Design B by Hand: the Menu Tiles

Iced tea: M P B B B B; fried rice: M P B B; geprek chicken: M P B B; *Save*: M P B B.

$T_B = 4(1.35) + 4(1.10) + 10(0.10) = 10.80$ s.

Saving: $25.54 - 10.80 = 14.74$ s, or 57.7%, on every such order.

The four M operators are identical in both designs: the thinking does not change, the pointing does. Book: Section 1.11.2.

Where the Time Goes

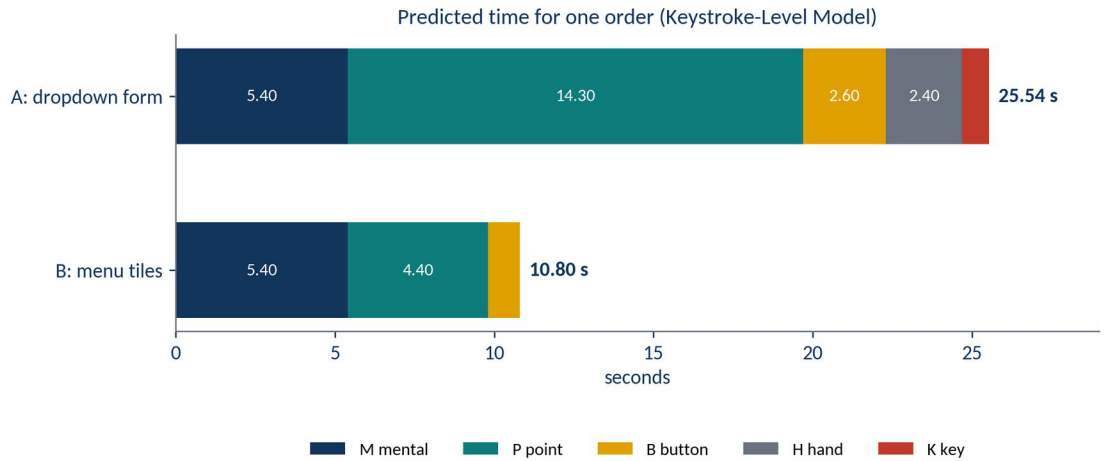


Figure 1.4 The predicted time of one order on the two pages, split by operator: pointing (P) dominates design A.

Figure 1.4 makes the argument visible to someone who does not read equations, such as Bu Ratna. Book: Section 1.11.2.

The Program Reads Designs as Text

Listing 1.1 `data/orders.txt`: the two designs written as Keystroke-Level Model operators.

```
# The order "2 iced tea, 1 fried rice, 1 geprek chicken" on two designs.
# K key, P point, B mouse button press or release, H hand moves between
# keyboard and mouse, M mental preparation. "2B" means "B B".
A: dropdown form
  M P 2B P 2B P 2B H K H P 2B     # iced tea: item list, quantity, Add
  M P 2B P 2B P 2B H K H P 2B     # fried rice
  M P 2B P 2B P 2B H K H P 2B     # geprek chicken
  M P 2B                          # Save
B: menu tiles
  M P 4B                          # iced tea tile, clicked twice
  M P 2B                          # fried rice tile
  M P 2B                          # geprek chicken tile
  M P 2B                          # Save
```

Listing 1.1: a name at the margin starts a design; indented lines add operators; `2B` means `B B`. Book: Section 1.11.3.

The Heart of `klm.py`

Listing 1.2 From `klm.py`: counting operators and applying the model.

```
def count(ops):
    """Count the operators in a text such as '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):
    """The predicted time: every operator count times its unit time."""
    return sum(counts[op] * TIMES[op] for op in ORDER)
```

Listing 1.2: an unknown letter raises an error, because a silent typo would make a design look faster than it is. `Klm.java` does the same with an int array. Book: Section 1.11.3.

Running It

```
$ 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).
```

Both languages print the same lines. At 120 orders a day, the tiles give back half an hour a day and 12.8 working hours a month. Book: Section 1.11.4.

What the Model Does Not Say

It predicts **error-free expert** time; beginners are slower and every error costs more.

It says nothing about learning, satisfaction or correct orders.

It is a cheap first estimate before code; test with real people afterwards.

Here the estimate confirmed what watching Bu Ratna showed: the design, not the user, was slow. Tests: 4 in Python, 5 checks in Java. Book: Sections 1.11.4 and 1.11.5.

HCI in Practice: Profitina

Score legibility: each AI-visibility signal explained in plain words with a weekly trend.

Consent under UU PDP: explicit consent, data export, password change and account deletion as self-service actions.

Feedback timing: posts appear at once; if sending fails, the typed text is restored.

The app advises and the user decides; when no local model is available it says so. Book: Section 1.12.

Common Mistakes

Designing for yourself; blaming the user; saying “usable” without saying for whom.

Trusting a statistic without asking what it counts; treating the KLM as the truth.

Installing from random websites; opening a single file instead of a folder in VS Code.

Each mistake has a cheap remedy in the chapter. Book: Section 1.13.

Summary

1. HCI designs, evaluates and implements interactive systems for human use; usability, UX and accessibility have ISO meanings.

2. Norman's gulfs of execution and evaluation explain most interface problems.

3. In Indonesia: 81.72% online, Android 77.07% of mobile page views, 11.97% aged sixty or older; the law now asks for good interfaces.

4. The workbench is JDK 25, Python 3.14 and VS Code.

5. The KLM (Equation (1.1)) predicts 10.80 s instead of 25.54 s for Bu Ratna's order.

Book: Section 1.14.

Next Time

Chapter 2 — Human Factors. Perception, attention and memory; Fitts's law and Hick's law; a puskesmas kiosk redesigned with numbers.

Before then: install the workbench, run `check_setup.py` and `klm.py`, and work Exercises 1.1, 1.3, 1.4 and 1.6.

Bu Siti, sixty-eight, will show us why a 20 mm button can still be too small. Book: Sections 1.15 and 1.16.

Lecture 2 — Human Factors: Perception, Cognition, Memory, Fitts's and Hick's Laws

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 2

Lecturer: Irfan Subakti — yifana@gmail.com

Today we study the body and mind that use every interface, and turn two laws of experimental psychology into a calculator for a puskesmas kiosk. Book: Chapter 2, Sections 2.1–2.16.

What This Meeting Covers

1. The person as an information processor — perceptual, cognitive and motor stages, and their limits.

2. Perception — the fovea, ageing eyes, colour vision, Gestalt grouping, what stands out.

3. Attention and memory — one thing at a time; about four chunks; recognition rather than recall.

4. Fitts's law and Hick's law — how long it takes to reach a target and to make a choice.

5. A calculator — fit both laws to logs and compare two kiosk layouts in Java and Python.

Book: Chapter 2.

Where We Begin: Bu Siti at the Kiosk

Puskesmas Kenanga's kiosk: twelve small text buttons in a grid at the top of the screen.

Bu Siti, 68, cannot read the letters with either pair of glasses; her finger covers *Elderly* and presses *Children*.

The ticket says "A-017, Children"; she notices forty minutes later, in the wrong room.

The staff say elderly patients "are not used to technology"; dr. Rina asks what the kiosk asked too much of. Book: Section 2.1.

What Did the Kiosk Ask Too Much Of?

Eyes: small text at an awkward distance.

Attention: twelve labels that all look alike.

Memory: holding the word sought while reading the others.

Hand: a 20 mm target at arm's length; and no second chance to notice the mistake.

11.97% of Indonesians were sixty or older in 2025 (BPS): the elderly clinic's patients are the kiosk's most regular users. Book: Section 2.1.

The Model Human Processor

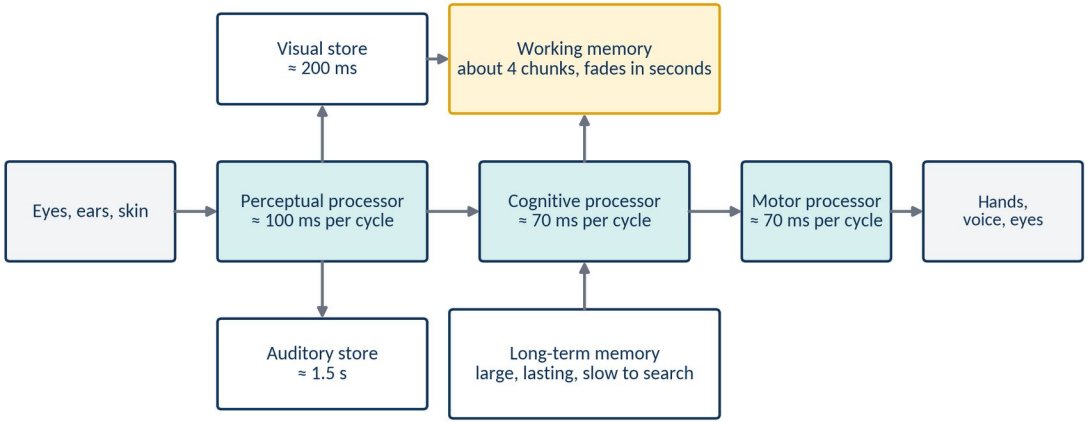


Figure 2.1 The Model Human Processor: perceptual, cognitive and motor processors with typical cycle times, the brief sensory stores and the two memories.

Figure 2.1: Card, Moran and Newell (1983). Even the simplest reaction takes a few hundred milliseconds; each stage can be overloaded while the others wait. Book: Section 2.2.

A Three-Question Checklist

Can they see it? (perception)

Can they decide? (cognition and memory)

Can they reach it? (motor)

A response within about a tenth of a second feels instantaneous; a second feels slow.

Modern engineering psychology has richer models; the three stages remain a good checklist. Book: Section 2.2.

Vision: Sharp Only in a Small Spot

The **fovea** covers about two degrees, a thumbnail at arm’s length; the rest is peripheral vision.

The eyes jump several times a second (**saccades**); text is read only where they land.

An error at the top of a long form is missed by a user at the bottom.

Put messages where the eyes are. Book: Section 2.3.1.

Designing for Older Adults

Generous text size and strong contrast; matte screens away from glare.

Large targets with space between them; no gestures that need speed or precision.

Fewer options per screen, plain words, and time to read and correct.

These rules, collected by Czaja and colleagues, help everyone; older users simply feel their absence first. Book: Section 2.3.1.

Ageing Eyes, and Colour

From the forties: **presbyopia**; the lens yellows; more light and contrast needed; more glare.

About **1 in 12 men and 1 in 200 women** have a colour vision deficiency, mostly red-green.

Never let colour be the only carrier of meaning (WCAG 2.2, 1.4.1).

Small text, low contrast and glossy screens in bright halls hit older users first. Book: Section 2.3.1.

Grouping Before Reading: Gestalt

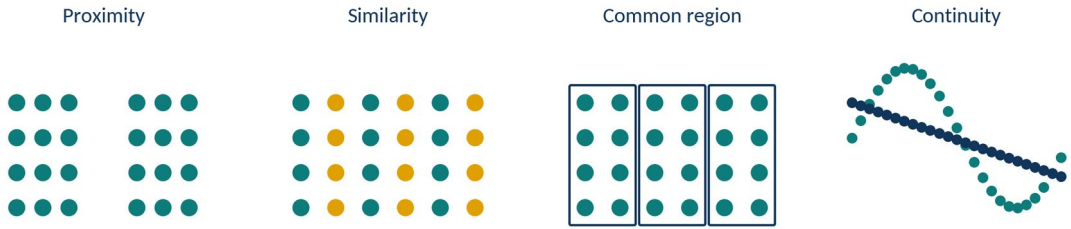


Figure 2.2 Four Gestalt principles: proximity, similarity, common region and continuity.

Figure 2.2: twelve identical buttons at identical distances say that all twelve services are equal; separating the common ones lets perception do half the search. Book: Section 2.3.2.

What Stands Out, and Hearing and Touch

A single red, large or moving item is found at once (**preattentive** features); combinations are checked one by one.

Material 3 Expressive: key elements found up to four times faster; the age gap narrowed.

A called number must be **shown and announced**; a touchscreen needs visible or felt feedback.

A highlight used everywhere highlights nothing. Book: Sections 2.3.3 and 2.3.4.

Attention: One Thing at a Time

Attention is selective and limited; switching tasks costs time and errors.

Inattentional blindness: viewers counting passes miss a gorilla crossing the scene.

Put feedback where attention already is; interrupt deliberately only when a mistake is costly.

Bu Siti, searching for a word, did not read the ticket that said Children. Book: Section 2.4.

Working Memory: About Four Chunks

Miller (1956): “seven, plus or minus two”; Cowan (2001): about **four chunks**, fading within seconds.

Chunking: 081234567890 is twelve items; 0812-3456-7890 is three.

Show account numbers, virtual accounts and one-time codes **grouped**.

Design for about four chunks and let the interface carry the rest. Book: Section 2.5.1.

Recognition, and Mental Models

Recall without a cue is slow; **recognition** of what is shown is easy.

Put knowledge **in the world**, not in the head: menus, visible options, recent items.

A visible queue (“number 17 for the Elderly clinic; 6 before you”) builds a correct **mental model**.

Keyboard codes demand recall, tiles offer recognition (Exercise 1.3). Book: Sections 2.5.2 and 2.5.3.

Fitts's Law

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

D: distance to the target; W: its width; ID: the index of difficulty in bits.

Equation (2.1): difficulty depends on the ratio D/W; doubling the distance adds a fixed amount, it does not double the time. Book: Section 2.6.1.

Why the Ratio, Not the Distance?

The hand first makes a fast, rough movement, then smaller corrections until it lands inside the target.

Each correction shrinks the remaining error by a roughly constant factor, so the number of corrections grows with $\log(D/W)$.

Twice as far and twice as wide needs the same corrections, hence the same time.

This is the intuition behind the logarithm; the law itself is an empirical result, confirmed for fingers, mice, styluses and eyes. Book: Section 2.6.1.

Movement Time and Throughput

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

a and b depend on the person, the device and the posture, and are **measured**.

Throughput $TP = 1/b$ bits per second (ISO 9241-411 uses an effective-width refinement).

Equation (2.2) is a straight line in ID, so a and b come from fitting a line to measured taps. Book: Section 2.6.1.

A Worked Example: the General Button

Old kiosk: centre $(45, 225)$ mm, hand at $(110, -40)$: $D = \sqrt{74450} = 272.9$ mm, $W = 20$ mm.

$ID = \log_2(272.9/20 + 1) = 3.872$ bits; $MT = 0.156 + 0.147 \times 3.872 = 0.725$ s.

New kiosk: 95 by 60 mm, $D = 245.7$ mm, $ID = 2.349$ bits, $MT = 0.501$ s.

W is the smaller side: a finger from below must stop within the button's height. The larger button saves a quarter second and is three times taller. Book: Section 2.6.2.

Target Sizes in Current Guidelines

Table 2.1 Minimum sizes for touch targets in current guidelines.

Source	Minimum target size
WCAG 2.2, 2.5.8 (AA)	24 by 24 CSS pixels, or enough spacing
WCAG 2.2, 2.5.5 (AAA)	44 by 44 CSS pixels
Apple Human Interface Guidelines	44 by 44 points
Android and Material Design	48 by 48 dp, about 9 mm

Table 2.1: floors, not goals. The old kiosk's 20 mm buttons exceed all of them and were still too small for Bu Siti. Book: Section 2.6.3.

Hick's Law

$$RT = a + b \log_2(n+1) \tag{2.3}$$

n equally likely options; the $+1$ is the option of not responding.

Practice constants: twelve buttons take $0.227 + 0.205 \times 3.700 = 0.986$ s; seven take 0.842 s.

Equation (2.3): Hick (1952) and Hyman (1953); decision time grows with the logarithm of the number of choices. Book: Section 2.7.1.

Unequal Choices: Entropy

$$H = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} \tag{2.4}$$

$RT = a + b\,H$: a regular patient for whom *General* is far more likely carries less uncertainty.

Equation (2.4): Exercise 2.3 finds $H = 2.806$ *bits for the kiosk's twelve shares. Book: Section 2.7.1.*

When Hick's Law Does Not Apply

It describes **practised choice** among known options.

A newcomer **searching** unfamiliar labels reads them one by one: time grows roughly linearly.

Never use it to defend a long menu; prefer broad and shallow to narrow and deep.

Fewer options, frequent ones first, related ones together. Book: Section 2.7.2.

The Proposal: Six Large Buttons and *Other*

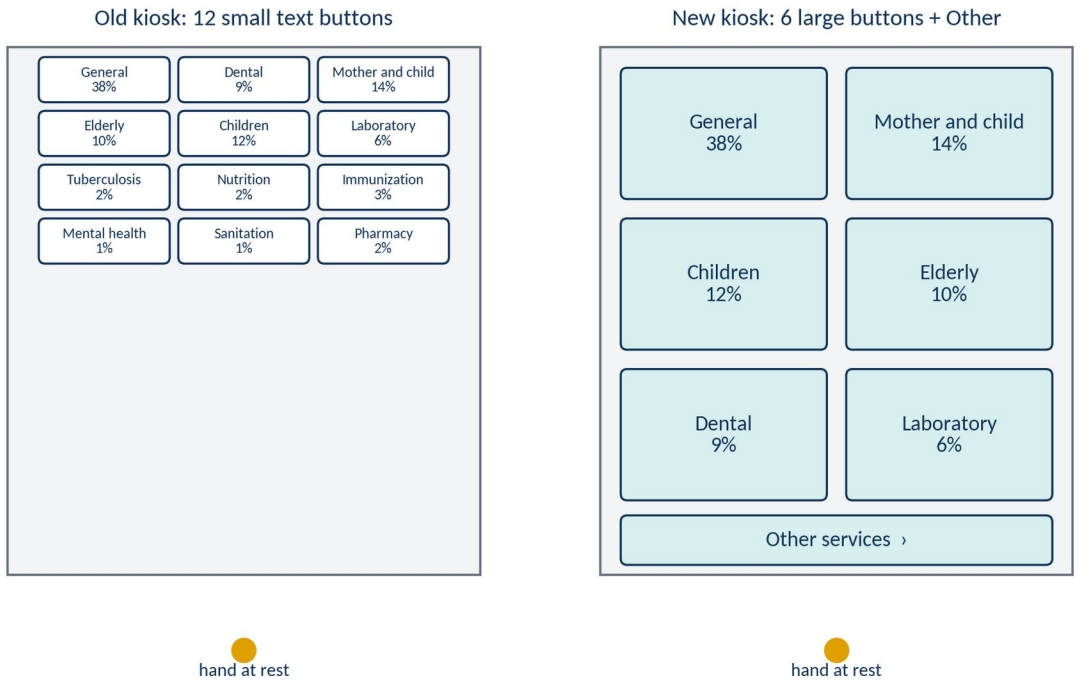


Figure 2.3 The old and the new first screen of the queue kiosk, to scale, with each service's share of patients and the resting point of the hand.

Figure 2.3: the six services on the first screen serve 89% of patients. Is it faster, for whom, and by how much? Book: Section 2.8.

Fitting a Line: Least Squares

$$b = \frac{\sum_i (x_i - \acute{x})(y_i - \acute{y})}{\sum_i (x_i - \acute{x})^2}, a = \acute{y} - b\,\acute{x} \tag{2.5}$$

Fitts: $x = ID$, $y = MT$; Hick: $x = \log_2(n+1)$, $y = RT$.

r^2 says how much of the variation the line explains.

*Equation (2.5). The logs in the chapter folder are **practice** logs generated from the laws plus noise; Exercise 2.6 replaces them with your own measurements. Book: Sections 2.8.1 and 2.8.2.*

The Expected Time of a Layout

Screen 1 service: Hick over all screen-1 buttons + Fitts from the hand to the button.

Screen 2 service: decide and move to *Other*, then decide and move from *Other* to the service.

Expected time per patient: each service's time weighted by its share.

The model ignores screen changes and first-time reading. Book: Section 2.8.3.

The Core of `humanfactors.py`

Listing 2.1 From `humanfactors.py`: the two laws and the least-squares fit.

```
def fitts_id(distance, width):
    """Index of difficulty in bits (Shannon form)."""
    return math.log2(distance / width + 1)


def hick_bits(choices):
    """Bits of a choice among n equally likely options (plus 'none')."""
    return math.log2(choices + 1)


def fit_line(xs, ys):
    """Least squares: the a and b of y = a + b*x, and r squared."""
    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
```

Listing 2.1: `HumanFactors.java` is the same, with `Math.log(x) / Math.log(2)` for the base-2 logarithm and a small CSV reader. Book: Section 2.8.4.

Running It

```
$ 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
```

The first three lines are the fitted laws; the old kiosk needs 1.70 s per patient, almost the same for every service. Book: Section 2.9.

The New Layout

```
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
```

1.44 s per patient, 15% less; the six common services take 1.22–1.34 s; the 11% on screen 2 pay about a second more. Book: Section 2.9.

The Fitted Laws

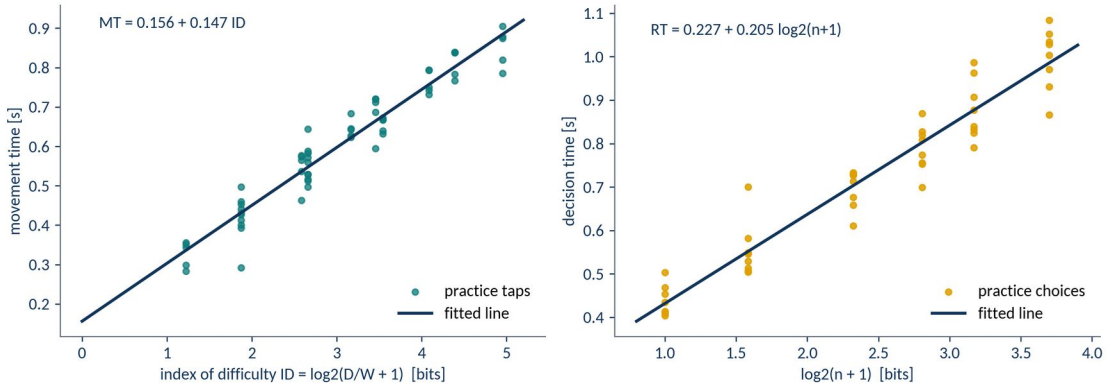


Figure 2.4 The practice logs with the fitted lines: Fitts's law (left) and Hick's law (right).

Figure 2.4: the lines recover the chosen constants closely but not exactly, as fitting noisy points should. Book: Section 2.9.

What the Numbers Do Not See

Reading unfamiliar labels: far slower than 1.7 s for a first-time user.

Errors: Bu Siti's wrong ticket came from a covered target; taller buttons prevent it.

Use the model for what it measures and usability tests (Chapter 7) for the rest.

Tests: 3 in Python, 6 checks in Java, on cases with exact answers. Book: Sections 2.9 and 2.10.

Record Your Own Logs

Five or more people from the real user group, with their consent, at the kiosk’s height.

Taps: four distances, three widths, five taps each, random order; record distance, width, time, misses.

Choices: a word and n buttons, n from 1 to 12, eight times each; fit each person and the group.

Twenty taps and twenty choices from three people of the intended age already beat any textbook constant for your kiosk (Exercise 2.6). Book: Section 2.11.

A Human-Factors Review, Stage by Stage

Perception: size, contrast, colour plus words, grouping.

Attention: one main action; feedback where the user looks.

Memory: about four chunks; recognition; grouped numbers.

Decision: rare options off the first screen. **Movement**: targets large, close, dangerous ones apart.

The checklist of the chapter turns the three questions into a review; replace the practice logs with measurements of your own users. Book: Section 2.11.

HCI in Practice: Profitina

The dashboard gathers the usual next steps on one screen, each clickable: fewer decisions, fewer movements.

The weekly score is a **line**, so perception, not memory, answers “am I improving?”.

“N new” counts only top-level posts; rare account tasks under UU PDP are separate, clearly labelled actions.

Book: Section 2.12.

Common Mistakes

Designing text and targets for young eyes and hands; colour as the only signal.

Believing “seven plus or minus two”; using Hick’s law to defend long menus.

Plugging textbook constants into Fitts’s law; taking the width in the wrong direction; reading a time model as the whole truth.

Book: Section 2.13.

Summary

1. Perceptual, cognitive and motor stages each have limits (Figure 2.1).

2. Fovea, ageing eyes, 1 in 12 men colour-deficient; Gestalt groups before we read.

3. Attention is limited; working memory holds about four chunks.

4. Fitts: $MT = a + b\,ID$ (Equation (2.2)); Hick: $RT = a + b\log_2(n+1)$ (Equation (2.3)).

5. The new kiosk: 1.44 s instead of 1.70 s, and far fewer wrong tickets.

Book: Section 2.14.

Next Time

Chapter 3 — Design Principles and Heuristics. Norman, Shneiderman, ISO 9241-110 and Nielsen; Liquid Glass and Material 3 Expressive; heuristic evaluation of a laundry's order page.

Before then: run `humanfactors.py` and work Exercises 2.1, 2.2, 2.4 and 2.7.

Four evaluators, 26 findings, 14 problems: Profitina Laundry's first order page. Book: Sections 2.15 and 2.16.

Lecture 3 — Design Principles and Heuristics

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 3

Lecturer: Irfan Subakti — yifana@gmail.com

Today: the principles that explain why designs work, the heuristics that let us check them, and a heuristic evaluation of a laundry's order page turned into a ranked report. Book: Chapter 3, Sections 3.1–3.15.

What This Meeting Covers

1. Norman’s principles — signifiers, mapping, constraints and forcing functions, feedback, conceptual models.

2. Three families of rules — Shneiderman’s golden rules, ISO 9241-110, Nielsen’s ten heuristics.

3. Principles meet 2025–2026 design languages — Liquid Glass, Material 3 Expressive, Fluent 2.

4. Heuristic evaluation — procedure, severity, and how many evaluators.

5. A report generator — merging 26 findings of four evaluators in Java and Python.

Book: Chapter 3.

Where We Begin: Profitina Laundry Leaves WhatsApp

Mbak Dewi's orders came by WhatsApp with a numbered menu: an hour every evening, and "2" confused with "3".
A new web order page works technically, yet in week one: duplicate orders, a Sunday pickup, questions about the price.
Her niece proposes a **heuristic evaluation**: four people, a short list of rules, one afternoon.

Profitina Laundry is the teaching case study of the book. Book: Section 3.1.

Before and After

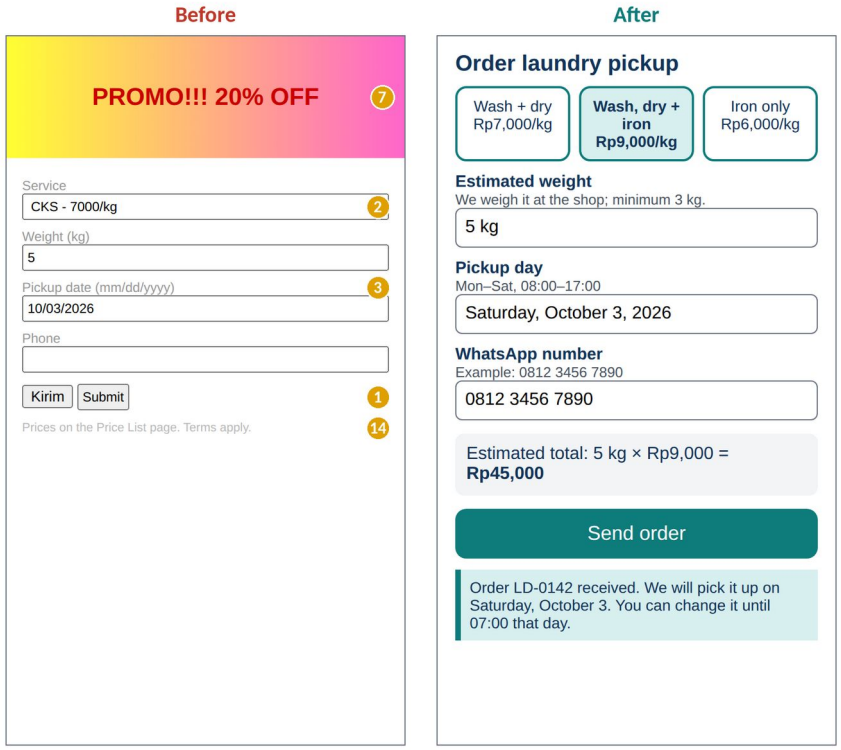


Figure 3.1 The order page of Profitina Laundry before and after a heuristic evaluation; the numbered markers point at problems P01, P02, P03, P07 and P14.

Figure 3.1: both pages are our own teaching pages rendered in Chromium. By the end of today we will know how the right-hand page was reached. Book: Section 3.1.

Principles, Guidelines, Heuristics

Principle: general, lasting, says *why* (people need feedback).

Guideline: specific to a platform, says *what* (48 by 48 dp targets).

Heuristic: a rule of thumb to *check* any screen against (visibility of system status).

Heuristic evaluation: a few evaluators inspect independently against heuristics, record location and severity, then merge. Book: Section 3.2.

Affordances and Signifiers

An **affordance** is what an object makes possible for a person; a **signifier** shows what and where.

On screens almost anything could react, so the signifier is what matters.

Kirim and *Submit* side by side are contradictory signifiers; “CKS” signifies nothing to a customer.

“Users do not see it is a button” is a complaint about missing signifiers. Book: Section 3.3.1.

Mapping

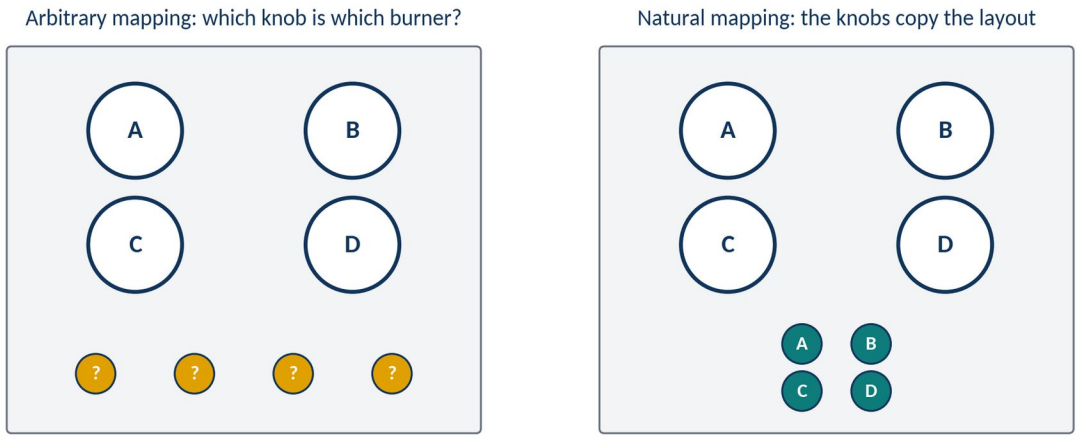


Figure 3.2 Mapping on a stove: knobs in a row give an arbitrary mapping, knobs in the same square as the burners give a natural one.

Figure 3.2: a natural mapping needs no label. The service tiles map three services to three visible choices with their prices. Book: Section 3.3.2.

Post-Completion Errors Everywhere

The campus photocopier: people take the copies and leave the original on the glass.

A forcing function: release the last copies only when the lid is lifted, or use a feeder that returns the original with the copies.

The fuel cap left open after paying: a cap tethered to the car is a constraint that prevents losing it.

Whenever the main goal comes before a final step, expect that step to be forgotten, and redesign the order (Exercise 3.4). Book: Sections 3.3.3 and 3.15.

Constraints and Forcing Functions

Constraints make wrong actions impossible: a date picker that greys out Sundays.

A **forcing function** blocks the next step until the previous one is done.

The ATM returns the card **before** the cash: the post-completion error disappears.

No training, no warning sign: a change of order removed the error. Book: Section 3.3.3.

Feedback and the Conceptual Model

Feedback: quick and clear — “Order LD-0142 received” — or users repeat actions.

Conceptual model: per-kilogram prices, an estimated weight, an order number, changeable until a stated time.

Together with signifiers, mapping and constraints they give **discoverability**.

Missing feedback is exactly why customers sent their order twice. Book: Section 3.3.4.

Shneiderman and ISO 9241-110

Eight golden rules: consistency; universal usability; informative feedback; closure; prevent errors; easy reversal; user in control; reduce memory load.

ISO 9241-110:2020: suitability for the task; self-descriptiveness; conformity with expectations; learnability; controllability; use error robustness; user engagement.

ISO's value: a tender or a contract can require compliance. Book: Section 3.4.

The Families Line Up

Informative feedback = self-descriptiveness = **H1**.

Easy reversal = controllability = **H3**; prevent errors = use error robustness = **H5**.

Reduce short-term memory load = **H6**, recognition rather than recall.

Choose one list per evaluation so that all evaluators speak the same language. Book: Section 3.4.

Nielsen’s Ten Heuristics

Table 3.1 Nielsen’s ten usability heuristics with examples from the first order page of Profitina Laundry.

Code	Heuristic	Example on the first order page
H1	Visibility of system status	No confirmation; no total price
H2	Match with the real world	Codes “CKS”, “CK”, “SL”
H3	User control and freedom	No way to change or cancel
H4	Consistency and standards	Two send buttons; month/day dates
H5	Error prevention	Past dates and Sundays accepted
H6	Recognition rather than recall	Prices on another page
H7	Flexibility and efficiency	No “repeat my last order”
H8	Aesthetic and minimalist design	Animated promotion
H9	Recover from errors	“Error 422”
H10	Help and documentation	No word on weighing or the 3 kg minimum

Table 3.1: derived in 1994 from 249 usability problems and used almost unchanged since. The three families agree far more than they differ. Book: Section 3.4.

Liquid Glass: a Principle Wins

Apple’s translucent material (June 2025) serves **H8**: content first, controls recede.

It strains **legibility**: text over busy content loses contrast, especially for older eyes.

iOS 26.1 added *Settings › Display and Brightness › Liquid Glass*: **Clear** or **Tinted** — user control (H3, H7).

When a style conflicts with a principle, the principle wins in the end. Book: Section 3.5.

What It Means in Indonesia

Most users are on Android phones, a significant minority on iPhones (Chapter 1).

Follow each platform’s current conventions (H4): users expect a screen to behave like their other apps.

Check every style choice for contrast and target size for older users.

Book: Section 3.5.

Material 3 Expressive and Fluent 2

Material 3 Expressive: stronger colour, shape, size and motion; 46 studies, 18,000+ participants, key elements found up to four times faster.

Fluent 2: consistency across Windows and Microsoft 365 (H4).

Follow the platform's conventions, and check every style choice against the principles.

A design system does not guarantee usability; the heuristics check the assembly. Book: Section 3.5.

Heuristic Evaluation in Five Steps

1. Prepare heuristics, scope and a recording template.

2. Evaluate independently, at least two passes.

3. Debrief and merge findings into distinct problems.

4. Rate severity. **5. Report and prioritize**.

Independence matters: evaluators who talk find the same problems. Book: Section 3.6.1.

Nielsen’s Severity Scale

Table 3.2 Nielsen’s severity scale for usability problems.

Rating	Meaning
0	Not a usability problem
1	Cosmetic: fix if time allows
2	Minor: low priority
3	Major: high priority
4	Catastrophe: fix before release

Table 3.2: severity combines frequency, impact and persistence. Book: Section 3.6.1.

How Many Evaluators?

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

Each evaluator finds a share λ; a problem is missed by all i with probability $(1-\lambda)^{i}$.

Nielsen and Landauer's average: $\lambda \approx 0.31$.

Equation (3.1): the expected number of problems found by i independent evaluators. Book: Section 3.6.2.

Estimating λ and the Team Size

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

F findings by n evaluators about P distinct problems; somewhat optimistic.

Equation (3.2): for the laundry page, $26/(4 \times 14) = 0.464$. *Book: Section 3.6.2.*

Evaluators Needed

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

$\lambda = 0.31$: four evaluators for 75% (3.74), seven for 90% (6.21).

Equation (3.3): both logarithms are negative, so the inequality keeps its direction. Book: Section 3.6.2.

The Evaluator Effect

Different evaluators find **different problems**: on the laundry page seven of fourteen were found by one person only.

They also **rate** the same problem differently, sometimes by two levels.

Merging and averaging independent ratings is what makes the list trustworthy.

Hertzum and Jacobsen called it a chilling fact about usability evaluation methods. Book: Section 3.6.2.

Diminishing Returns

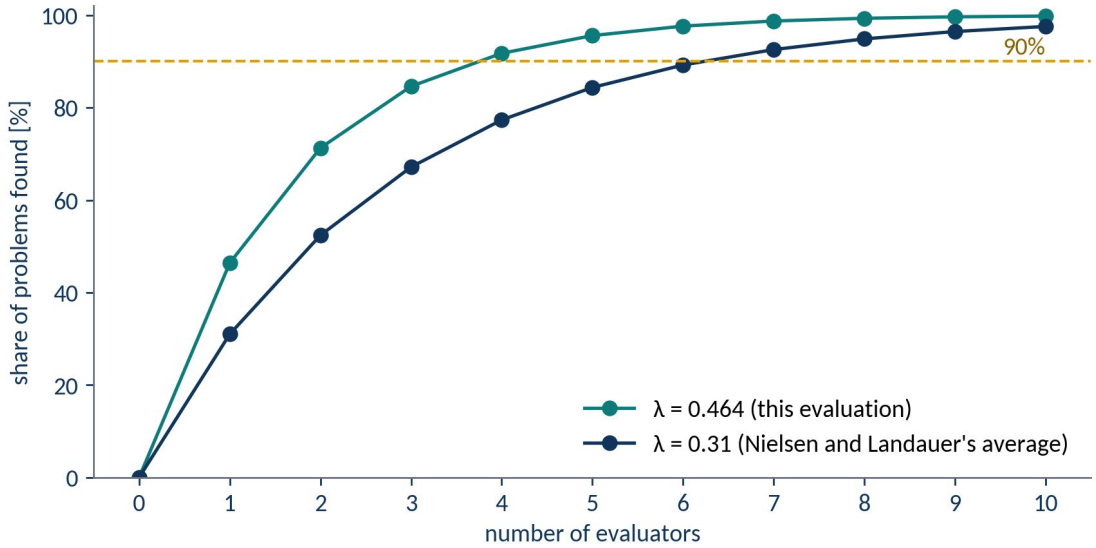


Figure 3.3 The share of problems found by i evaluators for λ = 0.464 (this evaluation) and λ = 0.31 (Nielsen and Landauer's average).

Figure 3.3: the flattening curve is why three to five evaluators is the usual advice. The evaluator effect adds that ratings differ too, so average them. Book: Section 3.6.2.

Relatives: Walkthroughs and Guideline Reviews

Cognitive walkthrough: one task, step by step; will the user try, notice, associate and see progress?

Guideline review: a long list of specific rules, such as WCAG 2.2; thorough but mechanical.

A practical order: heuristic evaluation, walkthrough of key tasks, accessibility review, then user tests.

Each costs hours, not weeks. Book: Section 3.6.3.

The Data: Problems and Findings

Listing 3.1 The first lines of `data/findings.csv`: evaluator, problem and severity.

```
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
```

Listing 3.1: `problems.csv` holds the fourteen problems agreed in the debriefing; `findings.csv` the 26 ratings. Merging stays human; counting is the program's job. Book: Section 3.7.1.

The Heart of `heuristics.py`

Listing 3.2 From `heuristics.py`: merging, ranking and the Nielsen–Landauer model.

```python
def merge(problems, findings):
    """Per problem: who found it and with which severities (0 to 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):
    """Most severe first; then found by more evaluators; then by 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 and Landauer: share of problems found by i evaluators."""
    return 1 - (1 - lam) ** i


def evaluators_needed(lam, target):
    """The smallest number of evaluators whose predicted share >= target."""
    return math.ceil(math.log(1 - target) / math.log(1 - lam))
```

Listing 3.2: ranking by a tuple (mean severity, evaluators, id). `Heuristics.java` adds a character-by-character CSV splitter for quoted commas. Book: Section 3.7.2.

Running It

```
$ 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
```

Three problems have mean severity 3 or more: no confirmation, no total price, no way to change an order — the causes of week one. Book: Section 3.8.

The Rest of the Report

```
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
```

Only one problem was found by all four evaluators and seven by just one; $\hat{\lambda} = 0.464$ predicts about 92% found by four. Book: Section 3.8.

A Fifth Evaluator

```
$ python3 heuristics.py data5
```

```
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
```

E5 found five problems, none new: $\hat{\lambda}$ falls to $31/(5\times14)=0.443$ and the list of fourteen looks close to complete (Exercise 3.3). Book: Section 3.15.

Why Not Only Ask Users?

Heuristic evaluation needs no users, no lab and little time: it is **discount** usability.

It finds what experts can foresee, not what only real users reveal.

Use it first to remove the obvious problems, then test with people (Chapter 7).

Book: Section 3.2.

From Report to Redesign

Visible service tiles with prices (P02, P05); weighing explained (P10).

A named pickup day with opening hours (P03, P13); an example phone number (P12); the total before sending (P11).

One send button (P06); a confirmation with order number and change deadline (P01, P08); no banner or grey print (P07, P14).

The right page of Figure 3.1 is the result; Chapter 7 checks it with real customers. Tests: 3 in Python, 5 checks in Java. Book: Sections 3.8 and 3.9.

The Recording Template

Table 3.3 A recording template for heuristic evaluation.

Field	Example
Evaluator	E2
Location	Order page, pickup date field
Problem	Accepts Sundays, when the shop is closed
Heuristic	H5
Severity	3
Suggestion	Offer only opening days

Table 3.3: one row per problem; after the debriefing the rows become `problems.csv` and `findings.csv`. Book: Section 3.10.

Using It for Real

Three to five evaluators, two hours each, one recording template: location, problem, heuristic, severity, suggestion.

Write problems as **observations**, not solutions; record positive findings too; evaluate again after fixes.

For AI features, add the guidelines for human-AI interaction as extra heuristics (Chapter 13).

Book: Section 3.10.

HCI in Practice: Profitina

H1: optimistic posting; a failed post restores the text with a clear message.

H5 and H9: Link Safety flags risky links and explains why.

H3: self-service export, password change and deletion under UU PDP; **H2**: plain-language signals; channel limits stated beside the button.

Book: Section 3.11.

Common Mistakes

Evaluating together from the start; using one evaluator; writing solutions instead of problems.

Rating severity alone; treating a heuristic evaluation as a user test.

Following a design system blindly; forcing every problem into a heuristic; stopping after one round.

Book: Section 3.12.

Summary

1. Principles say why, guidelines what, heuristics let us check.

2. Norman: signifiers, mapping (Figure 3.2), constraints, feedback, conceptual model.

3. Shneiderman, ISO 9241-110 and Nielsen (Table 3.1) agree.

4. Design languages apply and strain the principles; Liquid Glass gained a *Tinted* option.

5. i evaluators find $1-(1-\lambda)^i$ (Equation (3.1)); four found 14 problems on the laundry page.

Book: Section 3.13.

Next Time

Chapter 4 — User-Centered and Design-Thinking Processes. ISO 9241-210, the Double Diamond, design thinking; affinity diagrams, personas, journey maps; an ojek base that wants an app.

Before then: run `heuristics.py`, evaluate a page you use every day with the template, and work Exercises 3.1, 3.2, 3.4 and 3.6.

Book: Sections 3.14 and 3.15.

Lecture 4 — User-Centered and Design-Thinking Processes

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 4

Lecturer: Irfan Subakti — yifana@gmail.com

Today: how to organize a project so that it builds the right thing, the tools that turn interviews into requirements, and a program that does the bookkeeping of synthesis. Book: Chapter 4, Sections 4.1–4.15.

What This Meeting Covers

1. **User-centered design** — Gould and Lewis's three habits; ISO 9241-210's principles and activities.
2. **The Double Diamond and design thinking** — and Lean UX in agile teams.
3. **The working tools** — affinity diagram, persona, journey map, HMW questions, impact-effort matrix, measurable requirements.
4. **AI in the process, 2026** — what it can and cannot do; research data under UU PDP.
5. **A synthesis program** — from 37 notes to priorities in Java and Python.

Book: Chapter 4.

Where We Begin: an Ojek Base Wants an App

A *pangkalan ojek* at a market in East Java: twenty drivers, regular passengers, many of them older people who call the base.

Pak Darto asks students for “an app like the big ones, but ours”; they start drawing a map.

Their lecturer asks: **which problem of which person would that map solve?** Nobody knows.

Many regular passengers cannot read a map; drivers know the town by mosques and warungs, not street names. Book: Section 4.1.

What the Team Did Instead

Three mornings at the base; six passengers and four drivers; every observation on a note.

Findings: the worst moments are **waiting without knowing** and **agreeing on the price**; drivers worry about **fair turns**.

Proposal: a fare table, a “driver on the way” message, a share-my-ride link, later a fair queue — not a full app.

A fraction of the original cost, and it answers what people actually said. Book: Section 4.1.

The Idea in Three Habits

Focus early on users and tasks.

Measure empirically how people use simulations and prototypes.

Design iteratively: fix what the measurements reveal and measure again.

Gould and Lewis (1985); every modern process organizes these three habits. The opposite is designing from the inside out: from a technology, a competitor or the client's wish. Book: Section 4.2.

A Request Is a Hypothesis

“An app like the big ones” is a proposed **solution**, not a requirement.

Pak Darto knows his passengers better than anyone and is an essential participant.

The process begins by understanding the problem behind the request.

User-centered design, in ISO terms human-centred design, makes systems usable and useful by focusing on users, their needs and requirements. Book: Section 4.2.

ISO 9241-210: Six Principles

Explicit understanding of users, tasks and environments; users involved throughout.

Design driven and refined by user-centred evaluation; an iterative process.

The whole user experience; a multidisciplinary team (a driver and a passenger in the design sessions).

The principles say what must be true of a project; the activities say what the team does. Book: Section 4.3.

ISO 9241-210: Four Activities in a Cycle

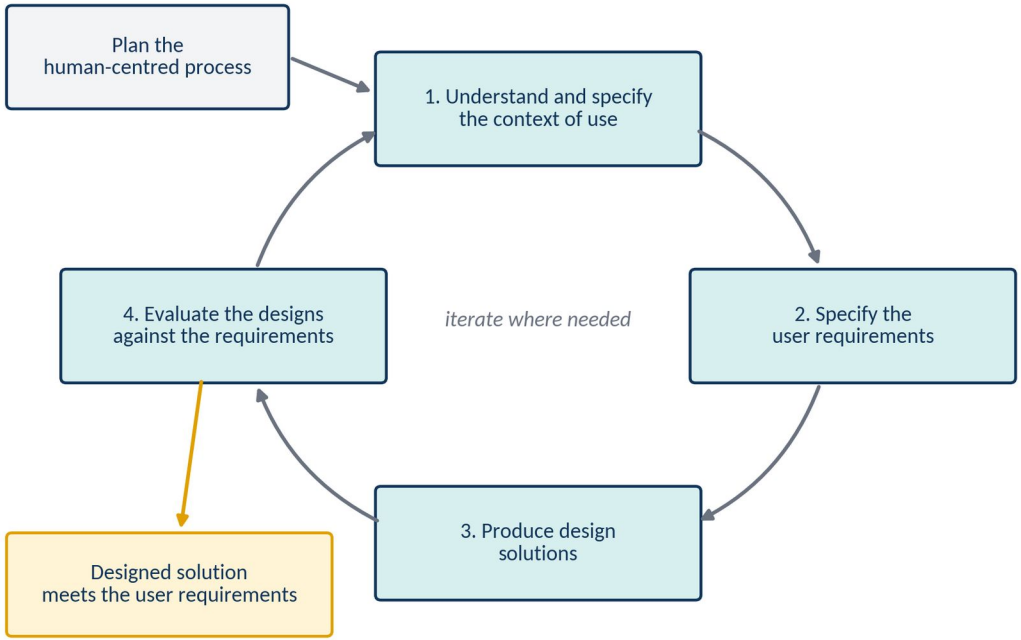


Figure 4.1 The human-centred design activities of ISO 9241-210: plan, then cycle through context, requirements, design and evaluation until the solution meets the user requirements.

Figure 4.1: an evaluation can send the team back to any activity, even the first, when it shows the context was misunderstood. Book: Section 4.3.

Context of Use and User Requirements

Context of use: users, goals, tasks, resources and environments — the sun at the gate, cheap phones with little data, the market's noise.

User requirements: what users must be able to do, and how well, written so that it can be **tested**.

The "specified" context of the usability definition in Chapter 1. Book: Section 4.3.

The Double Diamond

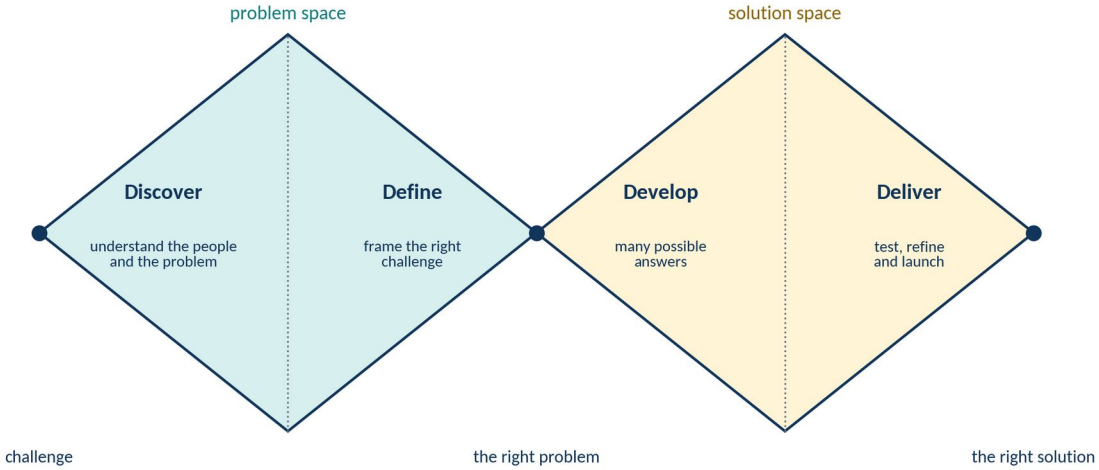


Figure 4.2 The Double Diamond: discover and define in the problem space, develop and deliver in the solution space; each diamond first widens, then narrows.

Figure 4.2: UK Design Council, 2004; Framework for Innovation 2019; Systemic Design Framework 2021. The most common mistake is skipping the first diamond. Book: Section 4.4.1.

Design Thinking and Lean UX

d.school's five modes: empathize, define, ideate, prototype, test — not a one-way sequence.

Lean UX: every design decision is a hypothesis with a measure of success.

Dual-track: discovery runs one or two cycles ahead of delivery.

"We believe showing the fare before booking reduces haggling; we will know when fewer passengers ask the price by phone." Book: Sections 4.4.2 and 4.4.3.

One Cycle, Three Vocabularies

Table 4.1 The same cycle in three vocabularies, with typical methods for each step.

ISO 9241-210	Double Diamond	Design thinking	Methods
Context of use	Discover	Empathize	Interviews, observation (Ch. 5)
User requirements	Define	Define	Affinity, personas, journey map
Design solutions	Develop	Ideate, prototype	Sketches, prototypes (Ch. 6)
Evaluation	Deliver	Test	Heuristics (Ch. 3), tests (Ch. 7)

Table 4.1: choose one vocabulary so that the team speaks one language; the others are translations. Book: Section 4.4.3.

The Affinity Diagram

One observation per note, as a **fact in the participant's words**, not an interpretation.

Group notes bottom-up without preset categories; name the groups: the **themes**.

37 notes formed eight themes, from "Unknown waiting time" to "Fair turns".

Moving notes is interpretation and stays human. Book: Section 4.5.1.

Count People, Not Notes

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

P_t: participants with a note in theme t; P: all participants.

Ten notes from one talkative person are one view; five notes from five people are a pattern.

Equation (4.1): the reach of a theme. Book: Section 4.5.1.

A Persona Built from Notes

Table 4.2 A persona built from research notes, with the design implications it suggests.

Field	Content
Name and context	Mbah Sri, 72, sells vegetables at the market
Goal	At the market by 5:30, home at noon, without depending on her son
Frustrations	Standing in the sun not knowing how long; cannot read a map
Design implication	No map; driver's name and arrival time; landmarks, not addresses

Table 4.2: "Would Mbah Sri use this?" has a concrete answer; "would users use this?" does not. A persona without evidence is a stereotype. Book: Section 4.5.2.

The Journey Map

Stages: **Find** a driver, **Agree** on place and price, **Wait**, **Ride**, **Pay**.

For each stage: actions, thoughts, feelings, touchpoints, and pain from 1 (mild) to 5 (severe).

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

Equation (4.2): the mean pain of stage s over its notes N_s. Book: Section 4.5.3.

Empathy Maps and Jobs to Be Done

Empathy map: says, thinks, does, feels — Mbah Sri *says* she does not mind waiting but *does* stand at the gate twenty minutes early.

Jobs to be done: nobody wants an ojek app; people want “to get to the market before the good vegetables are gone”.

Framing a need as a job keeps it separate from any one solution. Book: Section 4.5.4.

Problem Statements and “How Might We”

Problem statement: user, need, insight — older regular passengers need to know when the driver arrives, because waiting blind makes them leave too early.

HMW: “How might we show passengers when their driver will arrive?”

Too narrow: “build an app”; too broad: “improve transport”.

A good HMW question is broad enough for many answers and narrow enough to give direction. Book: Section 4.5.5.

The Impact-Effort Matrix

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

Quick win: high impact, low effort. **Big bet**: high impact, high effort.

Fill-in: low impact, low effort. **Avoid for now**: low impact, high effort.

Equation (4.3): value for effort orders ideas within a quadrant; votes break ties. The scores are judgments that the matrix makes visible. Book: Section 4.5.6.

Measurable User Requirements

“A regular passenger aged 60+ books from the saved link in under 30 seconds without help, in 4 of 5 attempts.”

“After booking, 9 of 10 passengers can say the driver’s name and expected arrival time.”

“In a month, calls asking where the driver is fall by half.”

Each can be checked in a usability test (Chapter 7) or with data (Chapter 14). Book: Section 4.5.7.

AI in the Process, 2026

Develop: Figma's Config 2026 agent, layout variants, draft copy, clickable prototypes, transcript summaries.

Discover: "synthetic users" reflect what is typical in training data, not what is true of these people.

AI may help you **work with** what people said; it may not **invent** what they would have said.

No language model would have found the drivers' worry about fair turns. Book: Section 4.6.

Research Data Are Personal Data

Notes, recordings and photos of participants fall under **UU PDP 27/2022**; **PP 33/2026** takes effect in January 2027.

Explain the purpose, ask consent to note or record, use codes (P1, D1), delete recordings after analysis.

Pasting raw transcripts with names into a cloud AI tool is a transfer of personal data.

Book: Section 4.6.

The Data: Notes

Listing 4.1 The first lines of `data/notes.csv`: participant, role, stage, theme, pain and the note.

```
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
```

Listing 4.1: 37 notes; `themes.csv` holds labels and HMW phrases, `ideas.csv` impact, effort and votes. Book: Section 4.7.1.

The Heart of `synthesis.py`

Listing 4.2 From `synthesis.py`: themes with their people, the journey, the quadrant and the ranking of ideas.

```
def affinity(notes):
    """Group notes by theme: note count, distinct people, roles."""
    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):
    """Mean pain (1 mild to 5 severe) and note count per journey stage."""
    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):
    """The impact-effort matrix: high impact is 4 or 5, low effort 1 or 2."""
    if impact >= 4:
        return "Quick win" if effort <= 2 else "Big bet"
    return "Fill-in" if effort <= 2 else "Avoid for now"
```

```
def prioritise(ideas):
    """Quadrant first, then value for effort, then votes."""
    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]))
```

Listing 4.2: a set counts different people. `Synthesis.java` uses records and TreeSets, and a CSV reader for quoted commas. Book: Section 4.7.2.

Running It: the Affinity Board

```
$ 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
```

Waiting without knowing reaches seven of ten people, $r = 70\ \%$; Fair turns comes only from drivers and would be missed without them. Book: Section 4.8.

The Journey and the Questions

```
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)
```

Agree and Wait average 3.6: for Agree, $36/10 = 3.6$*. The pain is before the ride, not during it. Book: Section 4.8.*

The Emotional Line

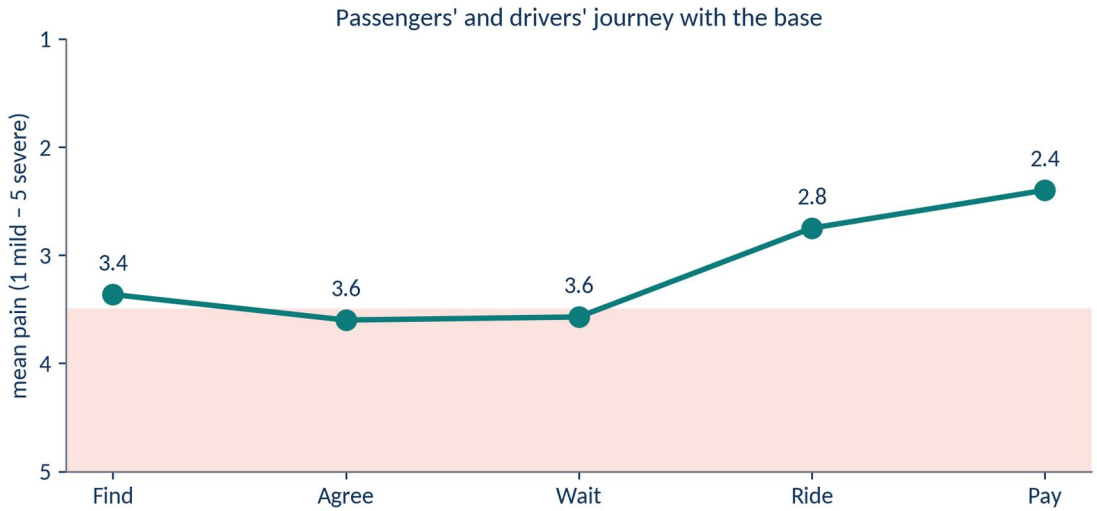


Figure 4.3 The emotional line of the journey map: mean pain per stage from the 37 notes; the axis is reversed and the shaded band marks severe pain.

Figure 4.3: a finding to state plainly to Pak Darto — the ride the drivers are proud of is fine; what hurts is everything before it. Book: Section 4.8.

The Ideas, Prioritized

```
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
```

Three quick wins, three big bets; the original “own full ride-hailing app” ends last with two votes. Book: Section 4.8.

A Driver's Idea in the Second Session

"A printed fare card at the base and in the shops": impact 4, effort 1, five votes.

```
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
```

It lands on top with value for effort $4/1 = 4.00$*, and it reaches passengers without smartphones, which the fare table shown before booking does not (Exercise 4.3). Book: Section 4.15.*

The Matrix

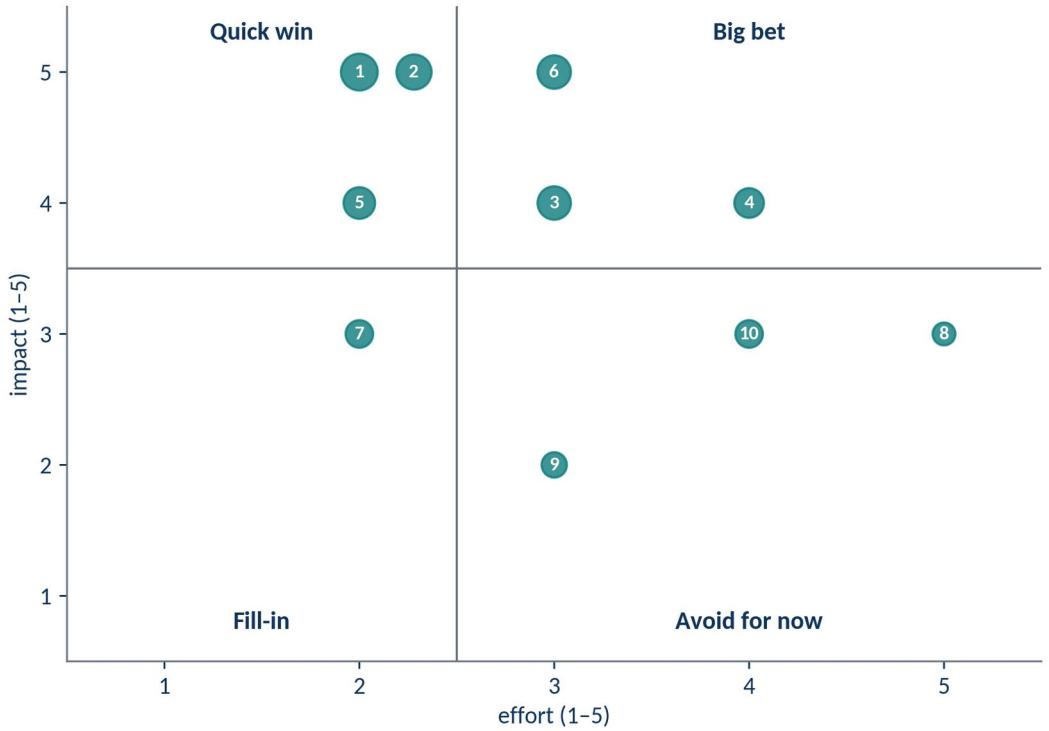


Figure 4.4 The ten ideas in the impact-effort matrix, numbered in the order of the ideas file; circle size shows votes.

Figure 4.4: quick wins upper left, big bets upper right. Book: Section 4.8.

The Recommendation

Start with the three quick wins: fare table, driver-on-the-way message, share-my-ride link.

Plan the fair queue with the drivers; prototype landmark-photo pickup with older passengers.

Do not build a full ride-hailing app; attach measurable requirements to each step.

The next cycle of Figure 4.1 checks whether it worked. Tests: 3 in Python, 5 checks in Java. Book: Sections 4.8 and 4.9.

A One-Day Synthesis Workshop

Morning: read and group notes; name themes; tag stages and pain; run the program; write personas and HMW questions.

Afternoon: ideate without judging; score impact and effort; collect participants' votes; run again; choose prototypes; write requirements.

Replace the stages and roles with your own: a laundry, a puskesmas, a campus service. Book: Section 4.10.

Two Cautions

The numbers summarize **judgments**: discuss disagreements rather than averaging them away.

A theme with low reach may still be **critical**, such as a safety issue raised by one night-shift nurse.

Numbers guide attention; they do not replace it. Book: Section 4.10.

HCI in Practice: Profitina

The **transparent feedback pipeline**: submit, curate, dataset, train and measure the local Qwen2.5, release — a visible user-centered loop.

Simplification from evidence: the Agency tier removed in August 2026, one 30-day Pro subscription.

“The app advises, the user decides” as a requirement: every AI suggestion can be evaluated and rejected.

Book: Section 4.11.

Common Mistakes

Starting from the solution; treating the client's request as the requirement; interviewing only one kind of user.

Counting notes instead of people; inventing personas or synthetic users.

Untestable requirements; doing the process once; mishandling participants' data.

Book: Section 4.12.

Summary

1. Early focus on users, empirical measurement, iterative design.

2. ISO 9241-210: six principles, four activities in a cycle (Figure 4.1).

3. Double Diamond and design thinking describe the same cycle (Table 4.1).

4. Reach (Equation (4.1)), journey pain (Equation (4.2)), value for effort (Equation (4.3)).

5. For the ojek base: waiting (70% reach) and price hurt most; three quick wins instead of a full app.

Book: Section 4.13.

Next Time

Chapter 5 — User Research Methods. Interviews, observation, surveys and diary studies: how the notes of this chapter are collected, and when to stop.

Before then: run `synthesis.py`, write a persona from P1's notes, and work Exercises 4.2, 4.3, 4.4 and 4.6.

Book: Sections 4.14 and 4.15.

Lecture 5 — User Research Methods

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 5

Lecturer: Irfan Subakti — yifana@gmail.com

Today we learn to replace guesses about users with evidence: choosing a method from the question, planning a small study with honest consent, interviewing without leading, surveying with known precision, and deciding when we have interviewed enough. Book: Chapter 5, Sections 5.1–5.16.

What This Meeting Covers

1. **The problem** — a laundry loses repeat customers and everyone has a different guess.
2. **The idea** — what people say versus what they do; why versus how many; the four questions that choose a method.
3. **Planning** — the one-page research plan, recruiting the people who left, and consent under UU PDP 27/2022 and PP 33/2026.
4. **Interviews and contextual inquiry** — asking about the last specific occasion, and the master-apprentice stance.
5. **Surveys** — good items, margin of error, confidence intervals, top-2-box, with Java and Python.
6. **Qualitative analysis** — coding, affinity diagrams and a stated rule for saturation.
7. **AI in research, 2026** — what helps, and why synthetic users are not evidence.

Every program today exists in Java and Python, and every output shown is what the program printed. Book: Chapter 5.

Bu Rina's Missing Customers

Laundry Bersih, near a campus in Surabaya: orders by WhatsApp, pickup by motorbike, clothes back in two days.

This year: four months of falling orders, about a third fewer repeat customers.

Four guesses, four investments: an app (the nephew), a lower price (a neighbor), a second motorbike (the courier), "they went home for the holiday" (the owner).

If the real reason is something else, every investment is wasted and customers keep leaving.

User research: a disciplined way of *finding out* before spending.

The case follows us through Chapters 6 and 7 as well: the messages we prototype and test come from this study. Book: Section 5.1.

Why It Matters in Indonesia, 2026

APJII (May 2026): internet penetration **81.7%**, about 235 million of 287 million people; 84.75% in cities, 78.18% in villages.

StatCounter (August 2026): of the *page views* it counts from Indonesia, 60.59% desktop, 38.98% mobile.

Not a contradiction: APJII asks **people**; StatCounter counts **page views** on sites carrying its code.

Lesson for the whole chapter: every number answers its own question. Know which one before you quote it.

A researcher who quotes "most Indonesians browse on desktop" from page-view data has confused two different measurements. Book: Section 5.1.

What User Research Is

> **DEFINITION.** *User research* is the systematic study of the people who use, or might use, a product: their behavior, needs, goals, context and attitudes. A *research question* is what the team needs to know; a *method* is how it finds out; a *finding* is what the study showed, stated with its evidence.

ISO 9241-210:2019: understand and specify the **context of use** before designing.

A study starts with a question and a decision, never with a method. “Let’s do a survey” is not a plan.

This is the first diamond of the design-thinking cycle in Chapter 4: widen the view of the problem before narrowing on a definition. Book: Section 5.2.1.

Say versus Do

People are poor witnesses of their own behavior: “every week” may be every ten days with a gap in July.

People predict their future badly: “would you use an app with tracking?” gets a polite yes.

Two axes: **attitudinal** (what people say) versus **behavioral** (what they do); **qualitative** (why, how) versus **quantitative** (how many).

Triangulation: when an interview, a survey and the order book agree, act with confidence.

Neither answer in the first bullet is a lie; memory keeps the typical case, not the actual one. Book: Section 5.2.2.

The Landscape of Methods

Figure 5.1 places the common methods on the two axes.

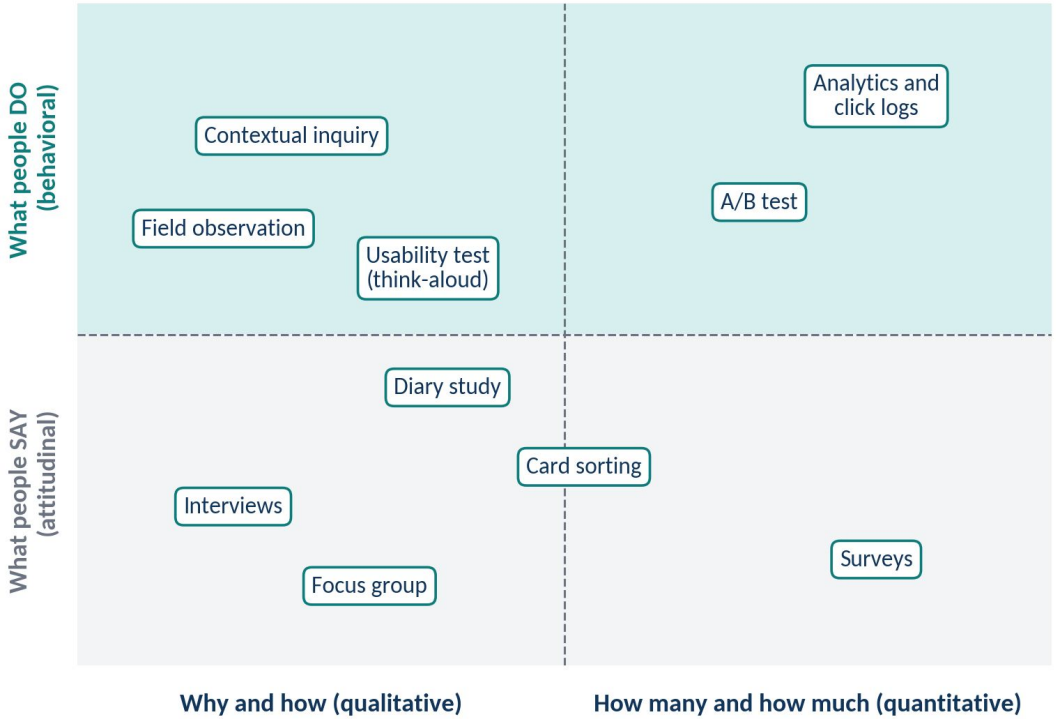


Figure 5.1 The landscape of user research methods: behavioral methods (top) watch what people do, attitudinal methods (bottom) record what people say; qualitative methods (left) explain why, quantitative methods (right) count how many.

No corner is better than another; a good study combines two. Book: Section 5.2.2 and Figure 5.1.

Four Questions That Choose the Method

1. What decision will this inform? Bu Rina: where to spend money in the next three months.

2. What do we already know, and how? The order book: fewer repeat customers, but not why.

3. Why or how many? “Why do customers leave?” is qualitative.

4. What can we afford? A few evenings, a customer list, small vouchers: a dozen interviews and a short survey.

Typical shape of a small study: **qualitative first to discover, quantitative second to measure.**

Research that cannot change the decision is not worth doing. Book: Section 5.2.3 and Table 5.1.

The One-Page Research Plan

Decision — price, pickup capacity, communication, or an app.

Questions — why do repeat customers stop? What happens between the message and the clothes coming back?

Methods — 12 semi-structured interviews, one morning of contextual inquiry, then a 6-item survey.

Participants — people who stopped *and* regulars; students, workers, families, boarding-house owners; at least two aged 60+.

Ethics and data — consent before recording; codes instead of names; recordings deleted after checking.

Output — coded notes, affinity diagram, survey summary, five insights, one-page recommendation.

The value is not the document but the conversation it forces before anyone talks to a customer. Book: Section 5.3.1 and Table 5.2.

Recruiting the Right People

Customers who left are the hardest to reach and the most important to hear.

A **screener** of four questions: last order, frequency before, occupation, willingness to talk for 30 minutes.

Include the edges: older customers, customers with disabilities (UU 8/2016 guarantees accessible services and information).

How many interviews? For one fairly homogeneous group, most themes appear within 6–12 interviews (Guest, Namey and Chen, 2020).

A study that leaves people out designs them out. Book: Section 5.3.2.

Consent under UU PDP and PP 33/2026

UU PDP 27/2022: a lawful basis for processing personal data; research normally uses explicit consent.

PP 33/2026: consent **free, conscious, specific, unambiguous**; information *before* consent; no bundling with terms and conditions; no deceptive methods.

Withdrawal: processing stops within **72 hours** of a verified request; refusing may not reduce a service that does not need the data.

Detailed obligations apply from January 16, 2027, but a study run today should already work this way.

Treating participants this way is the law soon and simply decent now. Book: Section 5.3.3.

The Six-Part Consent Script

1. Who is doing the study, and why.

2. What will happen, and for how long.

3. What is recorded, who sees it, when it is deleted.

4. Voluntary; stop at any time; no effect on the service.

5. The thank-you, whether or not the participant finishes.

6. Two separate questions: “Do you agree to take part?” and “Do you agree that we record the audio?”

> **WARNING.** Do not collect what you do not need: no address, family details or income for a study of ordering. Use participant codes from the start.

Taking part and being recorded are two different consents; PP 33/2026’s “specific” means asking them separately. Book: Section 5.3.3.

The Semi-Structured Interview

A guided conversation, not a questionnaire read aloud: topics and opening questions, then follow the participant.

Four parts: **warm-up**, **the story of the last time**, **specific topics** (pickup, price, waiting, paying, problems), **close**.

The heart: a *specific recent occasion*. “Usually it’s fine” → “tell me about the last time it wasn’t.”

The researcher makes the participant the expert and listens far more than talks (Portigal).

Generalities produce generalities; a particular Tuesday produces details. Book: Section 5.4.1.

Asking Without Leading

Table 5.1 Interview traps and better questions for the laundry study. ::: {custom-style="SlideBullet"} Table 5.1 shows the traps. Three habits: **silence** (count to five), **echo** (repeat the last words), **ask for the artifact** ("could you show me that chat?"). :::

Trap	Poor	Better
Leading	"Pickup times are too short, aren't they?"	"Tell me about arranging the pickup last time."
Prediction	"Would you use an app with tracking?"	"What did you do while waiting last time?"
Design	"What features should our app have?"	"What was annoying about the last order?"
Double	"Was the price clear and fair?"	"How did you find out the price?" then "What did you think of it?"

The WhatsApp history on the participant's own phone shows exact times and words, far better than memory. Book: Section 5.4.2 and Table 5.3.

Contextual Inquiry: Master and Apprentice

Interview where the work happens, while it happens (Holtzblatt and Beyer).

Principles: **context**, **partnership**, **interpretation**, **focus**.

A morning on Dimas's motorbike: at three boarding houses the bag was left with a guard who could not say when it would return.

An afternoon at the counter: the price message came 3–4 hours after pickup; the "ready" message sometimes after 9 p.m.

Nobody had complained in words, but chats were full of "Bu, sudah selesai belum?"

Observation found what interviews could not: the problem people had stopped noticing. Book: Section 5.4.3.

Field Notes and Diary Studies

One observation per line, each marked **quote**, **observation** or **interpretation**.

Keeping the three apart is the single most important habit: interpretations are questioned later; quotes and observations are the evidence.

Diary studies record entries each time something happens, over days or weeks.

Boarding-house owners ordered every Monday but decided on Sunday evening: a Sunday reminder might be welcome.

A one-hour interview compresses the ordinary rhythm; a diary study restores it. Book: Section 5.4.4.

Surveys: When and How

After the interviews, to measure *how widespread* each candidate problem is.

Items written in the customers' own words from the interviews.

Rules: one idea per item; concrete and recent; neutral wording (avoid acquiescence bias); same scale and direction; few items; pilot with three people.

Laundry survey: six Likert items (1–5), plus age group and channel.

A survey written before any interview measures the team's assumptions. Book: Sections 5.5.1 and 5.5.2.

How Precise Is a Survey?

For a proportion p from n answers, the 95% margin of error (Equation (5.1)):

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

Worst case $p = 0.5$. For $n = 60$: $1.96 \times \sqrt{0.25/60} = 0.127$, **±12.7 points**. For $n = 400$: ±4.9 points.

Needed sample: $n = z^2\, p(1-p)/E^2$; for $E = 0.10$, 97 answers.

Quadrupling the sample halves the margin, because n sits under a square root. Book: Section 5.5.3.

Confidence Interval of a Mean

For a Likert item with mean $\acute{x}$, standard deviation s and n answers (Equation (5.2)):

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

A program needs t without a table; a series around z is accurate from 5 degrees of freedom (Equation (5.3)):

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

For $v = 59$: $1.960 + 0.040 + 0.001 = 2.001$, as in the printed table.

Top-2-box: share of answers 4 or 5 — easy to explain, and a proportion, so Equation (5.1) gives its margin.

The sample standard deviation divides by $n - 1$ because sixty answers stand for a larger population. Book: Section 5.5.3.

The Survey Summarizer

`survey_stats.py` / `SurveyStats.java`: reads `survey.csv` and `items.csv`; prints mean, CI, top-2-box, the Q2 breakdown by age, and the margin of error.

Java sets `Locale.US` so both programs print a decimal point everywhere.

Output:

```
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
```

The item texts are read from a file, so the same program summarizes the next survey unchanged. Book: Sections 5.6.2–5.6.4.

Picturing Likert Data

Figure 5.2: a diverging stacked bar chart — disagreement to the left, agreement to the right.

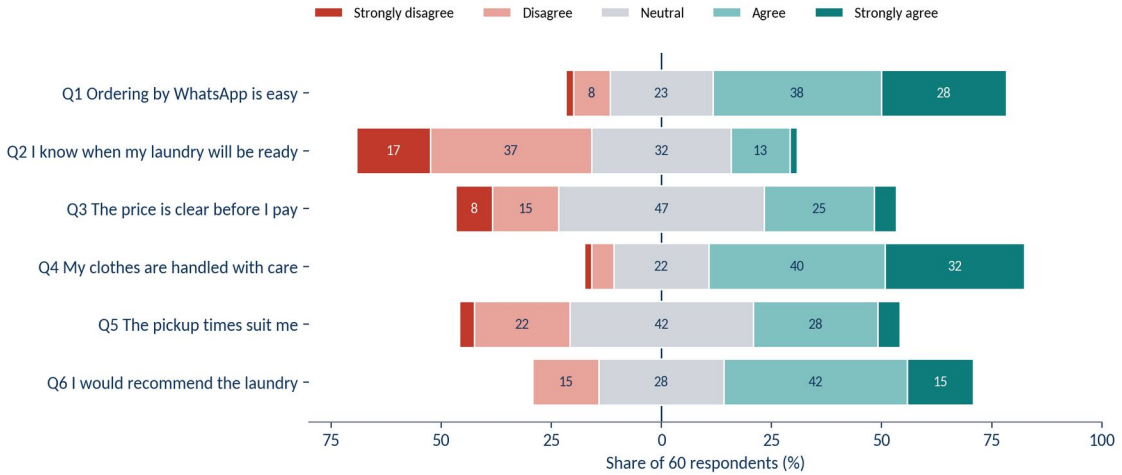


Figure 5.2 The six survey items as a diverging stacked bar chart (60 respondents); item Q2 leans furthest to the disagreeing side.

This is the clearest single picture of Likert data. Book: Section 5.6.4 and Figure 5.2.

Reading the Survey

Channel and core service are fine: Q1 3.83 (two thirds agree), Q4 3.95 (72%).

The weakest item by far: **Q2 “I know when my laundry will be ready”**, 2.47, only 15% agree; CI [2.21, 2.72] lies wholly below 3.

Age breakdown: 45+ at 2.14 ± 0.63, 18–29 at 2.68 ± 0.42 — intervals overlap: a **hint**, not proof.

±12.7 points at $n = 60$: “15% agree” means “a few percent to about a quarter”.

> **KEY INSIGHT.** Interviews told *what* and *why*; the survey told *how widespread*. Neither alone would have been enough.

Report hints as hints; the next interviews can test them. Book: Section 5.6.5.

From Notes to Codes and Themes

Coding: short labels on passages, e.g. `ready-time-unknown`, `reply-slow-evening`; a codebook grows as analysis proceeds.

Thematic analysis (Braun and Clarke): familiarize, code, search for themes, review, define and name, write up with quotes.

Affinity diagram: one idea per sticky note, group silently, name each group as a sentence from the customer's view.

"Communication problems" is a label; "I never know when my clothes will be back" is a finding someone can design for.

A theme supported by only one participant is an opinion, not a finding. Book: Sections 5.7.1 and 5.7.2.

When Have We Interviewed Enough?

Guest, Namey and Chen (2020): base size $b = 4$, run length $r = 2$, threshold $\tau = 5\ \%$.

B = distinct codes in the base; N_{run} = codes first appearing in a run; saturation at the first run with (Equation (5.4)):

$$\frac{N_{\text{run}}}{B} \leq \tau \tag{5.4}$$

Laundry study: base 10 codes; run 5–6 adds 1 (10%); run 6–7 adds 0 → **saturation at interview 7**.

The rule makes a vague idea precise and reportable. Book: Section 5.7.3.

The Saturation Program and Its Output

`saturation.py` / `Saturation.java`: code frequencies, new codes per interview, and the rule.

```
Base: 10 codes in interviews 1-4. Saturation reached at interview 7 (run of 2 with <= 5% new codes).
```

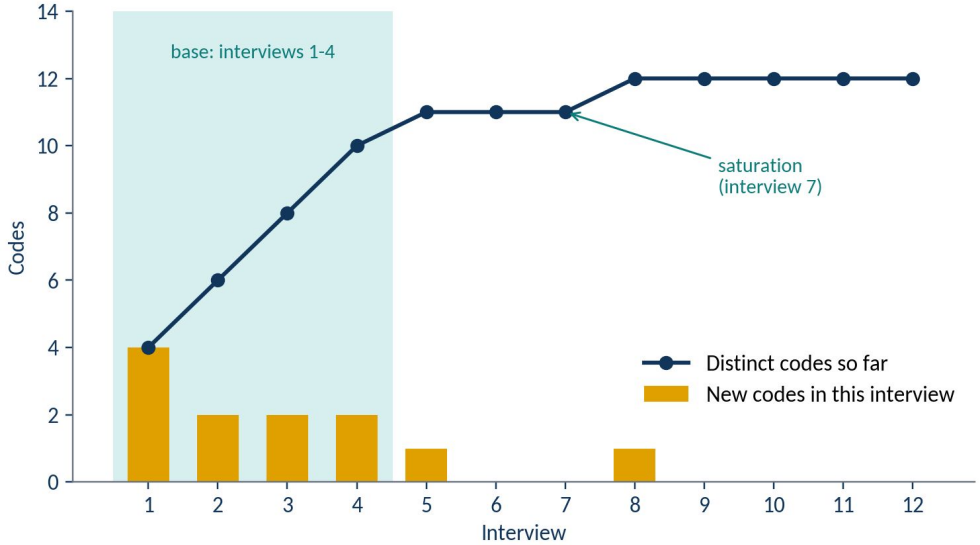


Figure 5.3 New codes per interview (bars) and distinct codes found so far (line); the rule declares saturation at interview 7.

Figure 5.3: the curve climbs in the base and flattens. Interview 8's lost sock arrives *after* saturation.

Saturation means the common themes are found, not that every story has been heard. Book: Section 5.7.4 and Figure 5.3.

Counts in Qualitative Data

`ready-time-unknown` in 11 of 12 interviews: strong evidence the problem is common *among the kinds of people interviewed*.

It is **not** a percentage of all customers; that is what the survey is for.

Rare problems need more participants and a different stopping rule (for example, safety studies).

Keep the two kinds of evidence in their own lanes. Book: Section 5.7.4.

Testing Analysis Programs

A wrong formula produces numbers that look perfectly plausible — so test.

`test_research.py` / `ResearchTest.java`: $t(59) \approx 2.001$, $t(9) \approx 2.262$; a hand-computed mean and top-2-box; margin at $n = 400$ is 4.9; saturation at 7 with base 10; never saturated when every interview is new.

```
7 checks passed
```

The Java tests use no library, so they run with nothing but the JDK. Book: Section 5.8.

Triangulation and Insights

Order book (behavioral, quantitative) + interviews and observation (qualitative) + survey (attitudinal, quantitative) all point at the **unknown ready time**.

The guesses: the competitor came up twice; the holiday explained part of the student drop; late visits were a symptom; **nobody asked for an app**.

Insight form: *[who] needs [what] because [why], but today [obstacle]*.

Example: customers need to know when laundry will be back *at handover*, because they plan their week's clothes, but today they learn it only when Dimas is about to leave.

Each insight becomes a "How might we…?" question for Chapter 4's ideation. Book: Sections 5.9.1 and 5.9.2.

What Bu Rina Did

Not an app — four changes to the WhatsApp service:

a pickup message with an estimated price range, a promised ready time and an item count;

a “ready” message by 4 p.m. with two return slots, one after 6 p.m.;

a photo of the folded laundry with the count;

a large-type price list sent as text, with voice notes welcome.

Research did not tell her what to build; it told her which problem to solve. Chapters 6 and 7 prototype and test these messages. Book: Section 5.9.3.

AI in User Research, 2026

Helps: transcription (check names and numbers), first-pass coding, clustering free-text answers — a person decides.

CHI 2026 study of a locally deployed open-source model for coding: valued for speed and surface extraction, trusted less for latent meaning; users asked for links back to source passages and lockable codebooks.

Harms: "synthetic users". NN/g (2024): useful for desk research or drafting a guide, never to replace real participants or validate designs — too agreeable, no priorities, no real experience.

Privacy: uploading recordings to a cloud AI is another processor under UU PDP and PP 33/2026; say so in the consent, prefer local tools.

> **WARNING.** An AI summary is an interpretation, not a quote. Never put it in quotation marks.

Ask a synthetic customer about an app with tracking and she says yes — exactly the answer that would have misled Bu Rina. Book: Section 5.10.

Using It for Real

Puskesmas queue: observe the 7 a.m. waiting room; ask nothing about illness (health data is *specific* personal data); refer symptoms to the nurse or doctor.

Warung ordering: lunchtime observation, five interviews, counts from the chat history with the owner's agreement.

Ojek base: interview during waiting time, 15 minutes, pay for the time.

Checklist: decision and questions written; method follows question; people who left and people at the edges; separate consents; last-occasion questions; survey after interviews; stated stopping rule and margin; every finding points to evidence.

The same study shape fits most small services. Book: Section 5.11.

HCI in Practice: Profitina

A score that must be legible: ask brand owners “tell me about the last time you looked at your score; what did you do next?”, not “do you like the dashboard?”.

Consent that is understood: Profitina implements UU PDP self-service rights; a short post-sign-up interview (“what did you just agree to?”) tests understanding better than click counts.

Feedback pipeline as continuous research: submit → curate → dataset → train and measure the local Qwen2.5 → release; code the feedback like interview notes.

Local AI processing avoids sending that data to third parties.

Profitina advises and the user decides; research is how the team learns whether the advice is understood. Book: Section 5.12.

Common Mistakes

Starting with a method instead of a question.

Asking people to predict or design.

Talking only to happy customers.

Leading questions; mixing quotes, observations and interpretations.

Writing the survey before the interviews.

A percentage without its margin; interview counts read as population percentages.

Saturation without a rule; synthetic users as evidence; one consent for everything.

Each mistake has a one-line fix in the chapter. Book: Section 5.13.

Summary

1. Research starts from a decision and questions; methods follow (Figure 5.1).

2. Say versus do, why versus how many; triangulate.

3. Plan on one page; recruit the people who left and the edges; consent informed, specific, unbundled, revocable (UU PDP, PP 33/2026).

4. Interview about the last specific occasion; observe in context.

5. Surveys: Equation (5.1), Equation (5.2), Equation (5.3); Q2 at 2.47 with 15% agreement.

6. Code, theme, and stop at saturation by a stated rule (Equation (5.4)).

7. AI assists; people decide; synthetic users are not evidence.

The laundry's real problem was the unknown ready time; nobody asked for an app. Book: Section 5.14.

Next Time

Chapter 6 — Prototyping: from paper to interactive and AI-assisted prototypes.

A puskesmas queue kiosk: paper sessions, a flow checker in Java and Python, and a coded prototype served by Flask and the JDK HTTP server.

Before then: work Exercises 5.2, 5.3, 5.4 and 5.6, and run `survey_stats` and `saturation` in both languages.

The laundry messages recommended today are prototyped as a Wizard of Oz in Section 6.9. Book: Sections 5.15 and 5.16.

Lecture 6 — Prototyping: From Paper to Interactive (and AI-Assisted) Prototypes

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 6

Lecturer: Irfan Subakti — yifana@gmail.com

Today a prototype becomes a question we can hold in our hands: we choose its fidelity from the question, test paper screens with a human "computer", check a flow with a program, build a clickable kiosk in Java and Python, and see what AI tools of 2026 add and hide. Book: Chapter 6, Sections 6.1–6.15.

What This Meeting Covers

1. **The problem** — a puskesmas queue at dawn and a vendor's kiosk brochure.
2. **The idea** — a prototype answers a question: role, look and feel, or implementation; fidelity; horizontal and vertical.
3. **Paper prototyping and Wizard of Oz** — the facilitator, the silent computer, the hidden wizard.
4. **Checking a flow** — breadth-first search finds unreachable screens, dead ends and the fewest taps.
5. **A coded prototype** — six screens, real queue numbers, a log, served by Flask and by the JDK HTTP server; Playwright screenshots.
6. **AI-assisted prototyping** — Figma Make, Google Stitch, and six rules.

The laundry messages from Chapter 5 are prototyped too, without a single screen. Book: Chapter 6.

The Queue at Dawn

A puskesmas in East Java: patients line up from a quarter past six; older patients first, because general-clinic numbers run out by mid-morning.

dr. Wulan's plan: a touch-screen kiosk; the vendor's brochure shows NIK login, a BPJS question, eleven services in small print, a confirmation.

A student team asks for one week and uses paper, scissors, a marker and a cardboard box.

They find three problems the brochure could not show: "BPJS atau Umum?" is misread as a clinic question; Help has no way back; clinic hours are unreachable.

All three are easy to fix once seen and expensive after installation. Book: Section 6.1.

A Prototype Is a Question

> **DEFINITION.** A *prototype* is any representation of a design, made before the finished product, that lets people experience or evaluate some aspect of it. *Fidelity* is how closely it resembles the finished product in appearance, interaction and content.

Houde and Hill: ask *what the prototype prototypes*.

Role — what it does in someone's life (does a kiosk shorten the dawn wait?).

Look and feel — what it is like to see and touch (are buttons big enough for a trembling finger?).

Implementation — can it be built (can a browser drive a thermal printer?).

A cardboard kiosk answers role and much of look and feel, nothing about implementation. Book: Section 6.2.1.

Sketch or Prototype? Right Thing or Thing Right?

Buxton: **sketches** are quick, cheap, many, disposable — *are we designing the right thing?*

Prototypes are refined, fewer, costlier — *are we designing the thing right?*

People comment on colors when shown polish, and on the idea when shown a pencil sketch.

> **KEY INSIGHT.** Match fidelity to the question, not to your tools. Early questions deserve rough prototypes; late questions deserve refined ones.

Jumping straight to polish answers the second question before the first. Book: Section 6.2.2.

The Fidelity Spectrum

Figure 6.1 arranges prototypes from minutes to days of work.

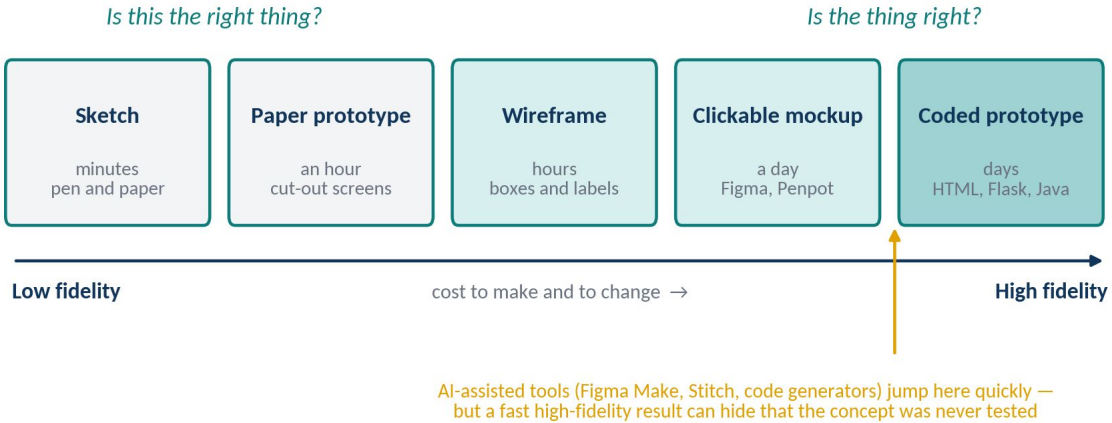


Figure 6.1 The fidelity spectrum: from sketches that ask whether this is the right thing to coded prototypes that ask whether the thing is right; AI-assisted tools reach high fidelity quickly, which makes it easy to skip the left-hand questions.

Cost to make and to change rises from left to right. Book: Section 6.2.2 and Figure 6.1.

Horizontal, Vertical and T-Shaped

Horizontal: many features, each only a surface (every kiosk screen exists; “Print” shows a picture).

Vertical: one path in depth (the dental path really issues numbers and estimates the wait).

T-shape: broad and shallow everywhere, deep along the one path that matters most.

The coded kiosk of this meeting is T-shaped.

Decide which paths deserve depth before writing any code. Book: Section 6.2.3.

Paper Prototyping: Materials and Roles

Materials: A5 sheets or cards, sticky notes for pop-ups, a marker, scissors, removable tape, a device frame (a cardboard kiosk front).

Facilitator: welcomes, explains that the design is tested, not the person; reads tasks; asks for thinking aloud.

Computer: never speaks; swaps screens exactly as the design says, including for “wrong” touches; a blank card with “?” for a missing screen.

Observers: one observation per line, as in Chapter 5.

The blank card is itself a finding: the participant expected something the design lacks. Book: Section 6.3.1.

A Session at the Puskesmas

Tasks: get a general-clinic number; find today’s dental hours; recover from pressing Help by mistake.

Eight participants over two mornings: five older patients, two mothers, one worker; consent as in Chapter 5.

The three big problems appeared within four sessions.

Smaller findings: headings touched as if they were buttons; lists that did not scroll; “how long will I wait?” asked before anything else.

None of this needed a line of code. Book: Section 6.3.2.

Wizard of Oz

A hidden person produces the system's responses by hand; the participant believes, or agrees to pretend, that the system does.

Kelley (1984) designed a natural-language office application this way before the language understanding existed.

At the puskesmas: a patient says "saya mau ke poli gigi"; a student behind the box pushes the confirmation screen into place.

Finding: the confirmation must be *spoken*, because a patient who cannot read the screen cannot read the confirmation.

Wizard of Oz is also how to prototype an AI assistant before building one (Chapter 13). Book: Section 6.3.3.

A Flow Is a Graph

Screens are nodes, buttons are arrows.

Unreachable: no way in. **Dead end:** no way out (unless it is a goal). **Taps to a goal:** length of the shortest path.

Format: `screen -> next : label`, plus `start` and `goal` lines (Listing 6.1).

Listing 6.1 An excerpt of `flow_v1.txt`: the kiosk flow of the paper test, one line per button.

```
start welcome
goal ticket
welcome -> service : Ambil antrean / Take a number
welcome -> help : Bantuan / Help
service -> bpjs : Umum / General
bpjs -> nik : Pakai BPJS / Use BPJS
bpjs -> confirm : Bayar sendiri / Pay myself
nik -> confirm : Lanjut / Next
confirm -> ticket : Cetak / Print
schedule -> service : Kembali / Back
```

Nobody draws graphs perfectly on paper; write the flow down so a program can check it. Book: Section 6.4.1.

Breadth-First Search

Visit screens in order of distance: the start, then everything one tap away, then two taps, using a queue.

The first time a screen is reached, a shortest path to it has been found.

After the search: never reached = unreachable; no outgoing button and not a goal = dead end.

`flow_check.py` uses `collections.deque`; `FlowCheck.java` uses `ArrayDeque` and a `TreeMap` so both print in the same order.

Programs check structure; people check meaning. Both are needed. Book: Section 6.4.2.

Running the Checker

Version 1 (paper test):

```
../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
```

Version 2 (BPJS moved to the desk, Help and hours reachable with Back, KIA and Lansia added):

```
../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
```

The checker found two of the three problems; the misread BPJS question is a problem of meaning that only people find. Book: Section 6.4.3.

Before and After

Figure 6.2 draws both flows.

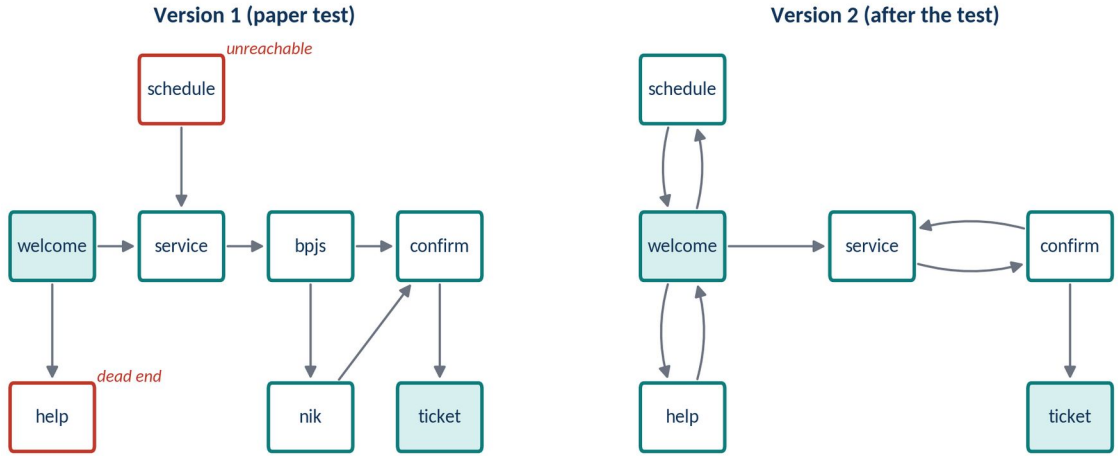


Figure 6.2 The kiosk flow before and after the paper test: version 1 has an unreachable screen and a dead end and needs four or five taps to a ticket; version 2 needs three, and every screen has a way back.

Four or five taps became three, and every screen has a way on or back. Book: Section 6.4.3 and Figure 6.2.

Wireframes and Clickable Mockups

Wireframes: structure without style — one question per screen, full-width primary button ≥ 72 px, service letter and name on every button, a yellow “PROTOTYPE” bar.

Clickable mockups in 2026: Figma (Config 2025: Figma Make, Sites, Draw, Buzz, Grid; Config 2026: Figma Motion, code layers, agent updates).

Penpot: open source, browser-based, self-hostable — useful for public institutions.

A mockup shows the flow, not that numbers are issued correctly or that two patients never get the same number.

For behavior and implementation, build a coded prototype. Book: Section 6.5.

What the Coded Prototype Must Do

Six screens of `flow_v2.txt` in Indonesian or English (`?lang=id|en`).

Real, increasing queue numbers per service; wait estimate = people ahead × minutes per patient.

Buttons ≥ 72 px tall (WCAG 2.2 SC 2.5.8 minimum is 24 × 24 CSS px), dark text on white, visible focus outline.

A “PROTOTYPE” bar on every screen; every screen view logged with a timestamp.

It does **not** print, touch patient records or store anything about the person pressing.

The waiting counts are made-up starting values; pretending otherwise would deceive participants. Book: Section 6.6.1.

The Screens: One Function

`render(screen, lang, code, number)` returns a complete HTML page; `kiosk_pages.py` and `KioskPages.java` produce byte-identical pages.

Texts of both languages in one dictionary: a translation is a change of data, not code.

Labels pass through `escape`; the page declares `<html lang="...">` for screen readers.

Tests can check pages without starting a server.

Keeping screens apart from the server makes both testable and comparable. Book: Section 6.6.2.

Two Servers

Flask (`kiosk_proto.py`, port 4060): one small `show` function per route; `lambda s=s` captures each screen name.

JDK HTTP server (`KioskProto.java`, port 4061): one handler for every path, 404 for unknown paths, query parsing.

The number counter and the log are `synchronized` in Java: two patients must never receive the same number.

```
 * Running on http://127.0.0.1:4060
```

`com.sun.net.httpserver` has been part of the JDK since Java 6 and needs no library. Book: Sections 6.6.3 and 6.6.4.

Screenshots with Playwright

`shoot.py` drives a real Chromium: open the welcome screen, take a number for the dental clinic, save a screenshot at each step.

```
title: Health Center Queue
ticket: B-004
saved 3 screenshots in ../screens/
```

Against the Java server the same three lines appear.

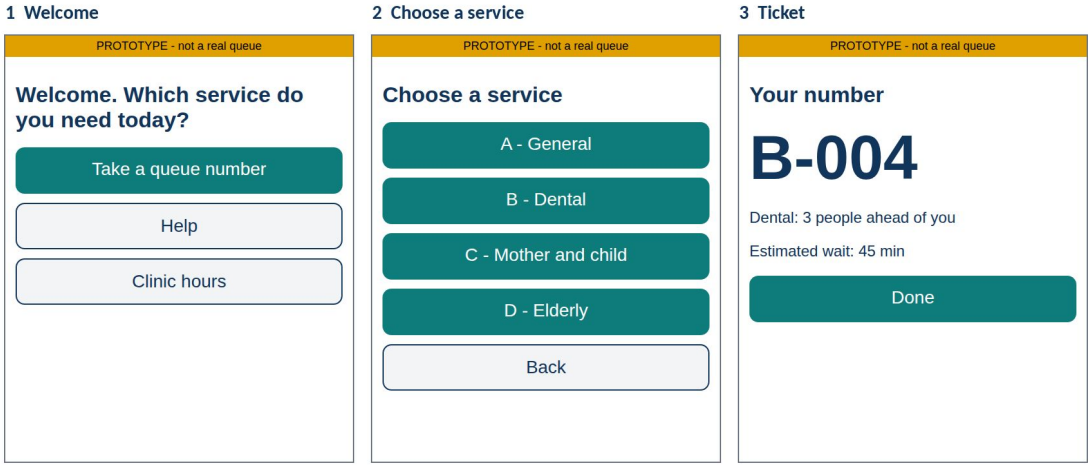


Figure 6.3 Three screens of the running kiosk prototype, captured by `shoot.py`: the welcome screen, the service choice and the ticket with the number of people ahead and the estimated wait.

Dental had three people waiting, so the first ticket is B-004 and the wait is 3 × 15 = 45 minutes. Book: Section 6.6.5 and Figure 6.3.

The Log and the Tests

Each view appends `time,lang,screen` to `data/events.csv`; differences of times are time on screen (Chapter 7 uses such logs).

`test_proto.py`: both flows; `lang` attribute and five buttons on the service page; through Flask's test client, B-004 then B-005 and two log lines (log redirected to a temporary file).

`ProtoTest.java`: the same checks without the server.

```
8 checks passed
```

A disposable prototype must still not mislead. Book: Section 6.7.

AI-Assisted Prototyping in 2026

Figma Make (May 2025): prompt or design → working prototype or app.

Google Stitch (May 2025): text and images, including sketches, → UI designs and front-end code, exportable to Figma, using Gemini models.

The team's prompt produced four attractive screens in under a minute — with NIK login, a BPJS question, a carousel and a tiny Help link: every problem the paper test had removed.

A tool can only follow the knowledge it is given; the findings lived in the observers' notes. Book: Section 6.8.1.

Six Rules for AI Prototyping

1. Sketch and test first; bring AI in when the flow is settled.

2. Prompt with your research findings.

3. Check generated screens against the flow file and an accessibility checklist.

4. Treat generated code as a prototype: it needs review and tests.

5. Mind what you upload: no real patient data; prefer local or self-hosted tools.

6. Test with people, as with the paper version.

> **WARNING.** A polished prototype made in a minute looks finished and invites polite approval. Show early AI prototypes in grayscale or on paper.

Speed at the right-hand end of the spectrum makes the left-hand end easy to skip. Book: Section 6.8.2.

Prototyping Messages: the Laundry

Chapter 5's recommendations are *messages*, so the prototype is message cards played over WhatsApp: a Wizard of Oz of a messaging service.

Findings: the price range needed a line on what makes it higher; the two return slots needed numbers ("reply 1 or 2").

Two weeks by hand from WhatsApp Business templates; automate only after the words settle.

Prototype the words, then the behavior, then the automation. Book: Section 6.9.

Using It for Real

A card: the question; the lowest fidelity that answers it; a flow file checked each round; 3–5 tasks with clear ends; 5–8 participants including the edges; honesty (label, consent, no real data); iterate.

ATM: cardboard front, paper screens and a real keypad for the order of withdrawal steps.

Ojek app: a clickable mockup tested at a driver base: is the pickup point seen before accepting?

Warung WhatsApp ordering: message cards played with regulars.

Two or three short rounds beat one long one. Book: Section 6.10.

HCI in Practice: Profitina

Instant, optimistic posting: the composer clears at once and the post appears; if sending fails, the text is restored. Timing can only be prototyped in code, with artificial delays and failures.

Quality gate: the Content Engine serves an AI draft (local Qwen2.5) only when it beats a built-in template — generated is not a synonym for better.

Widget over SSE: whether it keeps up on a slow phone connection is an implementation question for a small coded prototype.

Profitina advises and the user decides; prototypes find out before committing. Book: Section 6.11.

Common Mistakes

Starting at high fidelity; prototyping without a question.

Defending the prototype during a session.

Unreachable screens and dead ends — run `flow_check`.

Forgetting the “PROTOTYPE” label; real personal data in prototypes.

Trusting AI output as research; treating prototype code as product code.

Automating messages before the words work.

Each has a one-line remedy in the chapter. Book: Section 6.12.

Worked Problem: Shortest Is Not Typical

Exercise 6.3: in `flow_v1.txt`, a BPJS patient goes `welcome > service > bpjs > nik > confirm > ticket` — **five** taps.

`flow_check` reports **four**, because breadth-first search returns the *shortest* path, through “pay myself”.

At this puskesmas most early patients use BPJS, so the typical path is the long one.

Exercise 6.4 adds a depth-first search for the *longest* simple path; on `flow_v1.txt` it returns the five-tap BPJS path.

The search is exponential in the worst case: fine for ten screens, not for a website with thousands of pages.

Report the path that the most common users take, not only the lower bound. Book: Sections 6.14 and 6.15, Exercises 6.3 and 6.4.

Worked Problem: Adding a Language Screen

Exercise 6.2: add `welcome -> language : English / Bahasa Indonesia`.

The checker now reports `dead ends (no way on or back): language` — the new screen has no way out.

Add `language -> welcome : Kembali / Back` and the report returns to `none`.

Screens rise to 7, buttons to 12 then 13; the path to the ticket stays at three taps.

Lesson: every new screen needs a way on or back, and the checker catches the omission in a second.

Run the checker after every change to the flow, not only before a test round. Book: Section 6.15, Exercise 6.2.

Summary

1. A prototype answers a question: role, look and feel, implementation.

2. Fidelity matches the question (Figure 6.1).

3. Paper prototypes with a silent computer; Wizard of Oz for hard behaviors.

4. A flow is a graph; breadth-first search found the problems and cut the path from four or five taps to three (Figure 6.2).

5. The coded kiosk issues real numbers, logs views, uses large buttons, in Flask and the JDK server (Figure 6.3).

6. AI tools reach high fidelity in minutes; prompt with findings, check, and test with people.

Book: Section 6.13.

Next Time

Chapter 7 — Usability Evaluation and Testing: heuristic evaluation, think-aloud tests, SUS and UMUX-Lite, and A/B tests, with Java and Python.

Warung Bu Tini's pre-order page, eight customers, four tasks.

Before then: work Exercises 6.2, 6.3 and 6.5, and run both kiosk servers and `shoot.py`.

The event log of today's prototype becomes task times in Chapter 7. Book: Sections 6.14 and 6.15.

Lecture 7 — Usability Evaluation and Testing

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 7

Lecturer: Irfan Subakti — yifana@gmail.com

Today we check the claim that people can use a design: inspection by heuristics, think-aloud tests with real users, measures of success, time and satisfaction with honest intervals, how many participants are enough, and A/B tests on hundreds of real visitors. Book: Chapter 7, Sections 7.1–7.17.

What This Meeting Covers

1. The case — Warung Bu Tini's pre-order page, built by her son Arif.

2. The idea — effectiveness, efficiency, satisfaction; inspection versus empirical; formative versus summative.

3. Heuristic evaluation — independent evaluators and severity ratings.

4. Usability testing — tasks, think-aloud, what happened.

5. Measures — adjusted-Wald intervals, geometric mean times, errors, SUS, UMUX-Lite.

6. How many participants — the problem-discovery model, and its limits.

7. A/B testing — the two-proportion z-test and the sample size fixed in advance.

Every formula is worked by hand first and then computed by Java and Python programs with identical output. Book: Chapter 7.

Pre-Orders at Warung Bu Tini

Nasi campur near an office block and a campus in Surabaya; the lunchtime queue reaches the street.

Arif's page: choose dishes and quantity, choose a pickup time, pay by QRIS or at the counter, get a confirmation.

He tested it himself; friends said it was nice.

The question: *how do you know her customers can use it?* A retired teacher buying lunch for her husband; an office worker on a small phone between meetings.

"It works for me" is not evidence about anyone else. Book: Section 7.1.

What “Usable” Means

> **DEFINITION.** *Usability* is the extent to which a system can be used by specified users to achieve specified goals with *effectiveness*, *efficiency* and *satisfaction* in a specified context of use (ISO 9241-11:2018).

Not one number: effective but slow, or fast but frustrating, need separate measures.

Only for *specified* users, goals and context: every evaluation says whom, doing what, where.

Inspection (experts, principles) versus **empirical** (real users, real tasks).

Formative (what is wrong?) versus **summative** (how good is it?).

Arif’s test is formative; the A/B test at the end is summative. Book: Section 7.2.

Heuristic Evaluation

Nielsen and Molich (1990): a few evaluators judge compliance with a small set of principles, the heuristics of Chapter 3.

Procedure: **brief** the evaluators on users and context; **inspect independently**, twice; **merge**; **rate severity** independently and average.

One evaluator finds a modest fraction; several find far more — hence three to five.

Arif's classmates found six problems in an afternoon, including an unlabeled menu icon hiding "change order" and the error text "Invalid input".

They missed three that users found, such as a confirmation that did not say when the food would be ready. Book: Section 7.3.1.

Severity Ratings

Table 7.1 The severity scale for usability problems, with examples from the ordering page. ::: {custom-style="SlideBullet"} Table 7.1: severity combines **frequency**, **impact** and **persistence**. :::

Rating	Meaning	Example
0	Not a usability problem	A color the evaluator dislikes
1	Cosmetic	Menu photos load slowly
2	Minor	Price updates only after leaving the field
3	Major	QRIS code too small to scan
4	Catastrophe	Back button empties the cart

A problem few users meet every time and cannot overcome may outrank one everybody meets once. Book: Section 7.3.2 and Table 7.1.

Planning the Test

Research questions: can customers order, change an order and pay without help? Equally for older customers?

Participants: eight regulars — four office workers, two students, two retired residents over 60 (P3, P6).

Setting: their own phones, in the warung after the rush.

Tasks: **T1** find today's menu and price; **T2** order two portions for 12:00; **T3** change to three portions; **T4** pay by QRIS (test code).

Measures: success, time, errors, SUS and UMUX-Lite, problems observed.

Tasks are goals in the customer's words, never instructions in the interface's words. Book: Section 7.4.1.

Thinking Aloud

Ericsson and Simon: people can *verbalize* working memory while working without much changing it — ask them to say, not to explain.

The facilitator mostly stays quiet; neutral prompt: “what are you thinking now?”

Turn questions back: “where would you expect to find it?”; help only when stuck, and count the task as a failure.

Moments: P3 ordered again instead of changing (double charge); P6 scrolled through times from 07:00; P1’s back button emptied the cart; P5 found everything “fine”.

Confident users show that a design works for them; they cannot show that it works for everyone. Book: Sections 7.4.2 and 7.4.3.

Task Success with an Honest Interval

With small n, the ordinary interval fails ("8 of 8" gives 100% to 100%).

Adjusted Wald (Sauro and Lewis) adds $z^2/2$ successes and z^2 trials (Equation (7.1)):

$$\hat{p}_{\text{adj}} = \frac{x + z^2/2}{n + z^2}, \hat{p}_{\text{adj}} \pm z \sqrt{\frac{\hat{p}_{\text{adj}}(1 - \hat{p}_{\text{adj}})}{n + z^2}} \tag{7.1}$$

T1, 8 of 8: $9.921/11.842 = 0.838$, half-width 0.210 → **62.8% to 100%**.

T3, 4 of 8: 0.500 ± 0.285 → **21.5% to 78.5%**.

Wide intervals are an honest statement of how much eight people can tell you. Book: Section 7.5.1.

Time on Task and Errors

Task times are skewed; for n below about 25 use the geometric mean (Equation (7.2)):

$$\acute{t}_{\text{geo}} = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\ln t_i\right) \tag{7.2}$$

T4's seven times: log-mean 4.021 → **55.8 s** (arithmetic mean 58.4 s).

Failed attempts are left out of times; they are counted in completion.

Errors: actions that do not lead toward the goal — they show *where* a design misleads.

One very slow participant can dominate an arithmetic mean of eight. Book: Sections 7.5.2 and 7.5.3.

The Task Results

```
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, changing an order, is the problem: half failed, the rest were slowest with most errors.

Failures: P1 (back button), P3 (ordered again), P6 (gave up), P8 (never found the link).

The table comes from `usability_metrics.py` and `UsabilityMetrics.java`. Book: Section 7.5.4.

The System Usability Scale

Brooke (1996): ten items, 1–5, alternating positive (odd) and negative (even) wording.

Score (Equation (7.3)):

$$\text{SUS} = 2.5\left(\sum_{i \in \{1,3,5,7,9\}} (a_i - 1) + \sum_{i \in \{2,4,6,8,10\}} (5 - a_i)\right) \tag{7.3}$$

P1 answered 4, 2, 4, 1, 3, 2, 4, 2, 4, 2: odd 14, even 16, total 30 → **75**.

Not a percentage: the average product scores **68** (Sauro); curved grades, e.g. 65.0–71.0 is C, 80.8–84.0 is A, 84.1+ is A+.

Translate “system” into the product’s name and check the translation by back-translation. Book: Section 7.6.1.

UMUX-Lite

Lewis, Utesch and Maher (2013): two items on a 7-point scale — “capabilities meet my requirements”, “easy to use”. Raw score and its regression onto the SUS scale (Equation (7.4)):

$$\mathrm{UMUX}_{\mathrm{raw}} = \frac{(c-1)+(e-1)}{12} \times 100\,, \quad \widehat{\mathrm{SUS}} = 0.65\,\mathrm{UMUX}_{\mathrm{raw}} + 22.9 \tag{7.4}$$

P1: $c = 6$, $e = 5$ → raw 75.0 → 71.65 on the SUS scale.

Use it where ten items would be skipped, such as a pop-up after an order. Book: Section 7.6.2.

Satisfaction Results: Look at Everyone

```
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
```

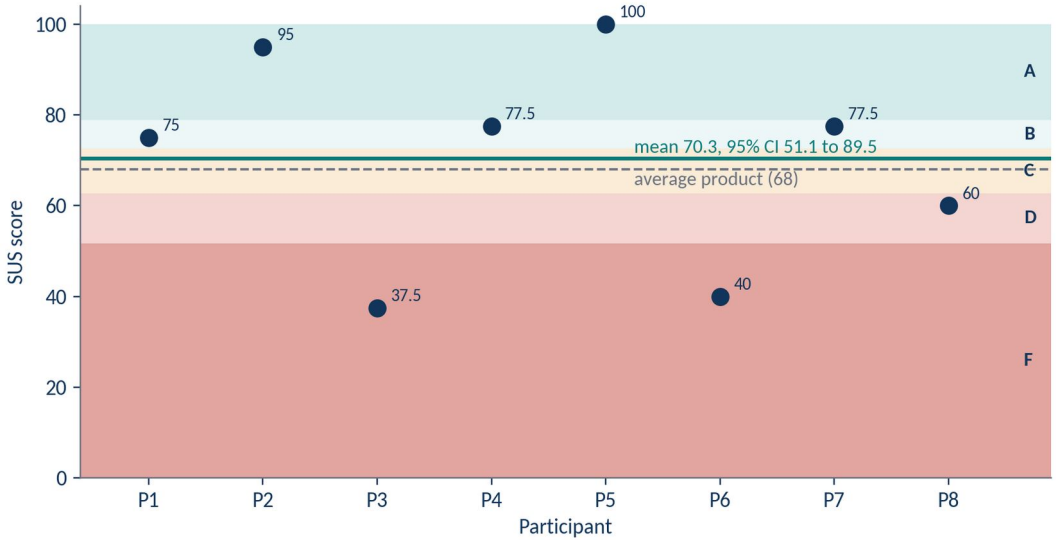


Figure 7.1 SUS scores of the eight participants against the curved grading bands; the mean (70.3) is just above the average product (68), but the two older participants score in the F band.

Figure 7.1: six scores from 60 to 100; **P3 37.5 and P6 40** — the older participants, the same two who failed T3.

KEY INSIGHT. Report distributions, not only averages. An average can hide exactly the people at the edges.

Book: Section 7.6.3 and Figure 7.1.

The Metrics Program

One function per formula: `adjusted_wald`, `geometric_mean`, `sus_score`, `umux_lite`, `discovery`.

The grading scale is a list of lower bounds; `grade` returns the first one reached.

`sus_score` uses list positions: even positions are the odd-numbered, positive items.

Java reads CSV with `split(",")`, enough because no field contains a comma; `Locale.US` keeps decimal points.

Both versions print identical reports. Book: Section 7.7.

How Many Participants?

Nielsen and Landauer: if each participant meets a problem with probability p, n participants find it with (Equation (7.5)):

$$P(\text{found}) = 1 - (1 - p)^n \tag{7.5}$$

With $p = 0.31$: five participants find $1 - 0.69^5 = 0.844$, about **85%**.

Nielsen's advice: test about five, fix, and test again.

The best return comes from several small rounds, not one large one. Book: Section 7.8.1.

Estimating p from the Study

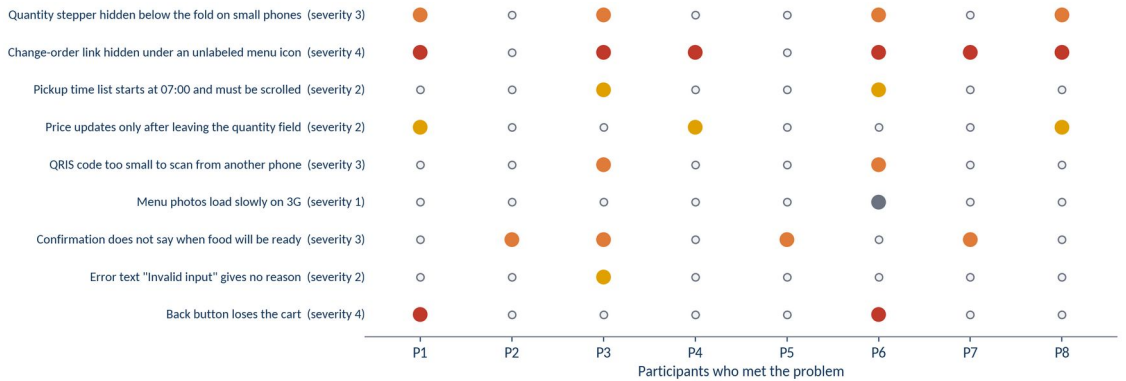


Figure 7.2 The nine problems of the warung test and the participants who met each; P3 and P6 meet the most problems.

Figure 7.2: 9 problems × 8 participants = 72 cells, 25 sightings → $\hat{p} = 0.347$.

The matrix already shows that participants are not alike. Book: Section 7.8.2 and Figure 7.2.

Three Warnings About “Five Users”

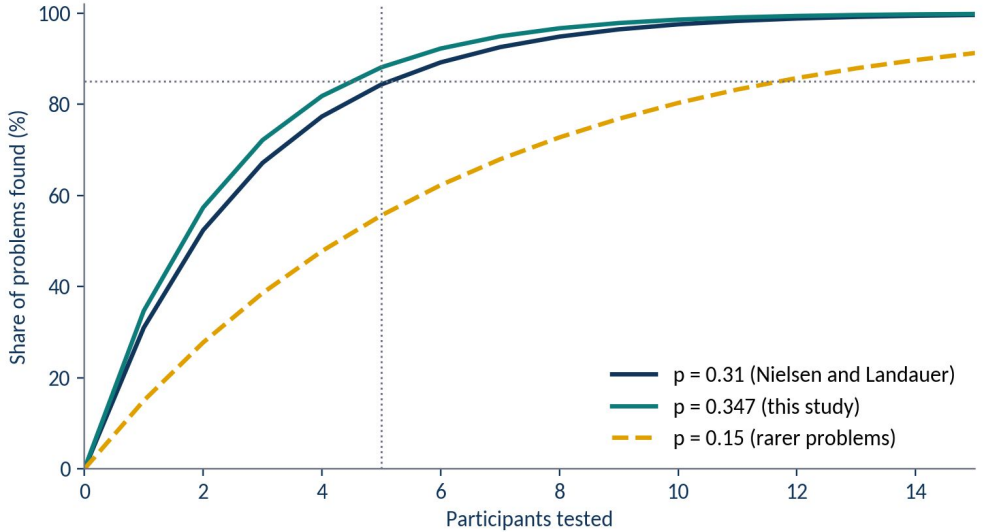


Figure 7.3 Expected share of problems found as participants are added: with p around 0.3, five participants find about 85%; with rarer problems (p = 0.15), about 56%.

Figure 7.3. **1.** $\hat{p}$ from a small study is biased upward (Lewis, 2001). **2.** The model assumes alike participants — five confident office workers would have hidden the older customers’ problems. **3.** Rarer problems need more: with $p = 0.15$, five find only 56%.

Five users find most problems of the kind of user tested. Test each distinct group. Book: Section 7.8.2 and Figure 7.3.

Testing the Metrics

Known answers: SUS 100 and 0 at the extremes, 50 for all threes; grade boundaries; UMUX-Lite top (100, 87.9); adjusted Wald 8/8 from 62.8%; geometric mean of 10 and 1000 is 100; discovery 84% for five; $\Phi(1.96)=0.975$; the A/B test.

```
11 checks passed
```

Formulas that look plausible when wrong deserve tests. Book: Section 7.9.

From Eight People to Hundreds: A/B Testing

After fixes (cart kept, labeled “Change order”, times from 11:30, large QRIS, “Ready at 12:00”), a second round with five new participants found no T3 failure.

Remaining question: which button label makes more visitors order? A *Kirim* versus B *Pesan sekarang – ambil jam 12.00*.

> **DEFINITION.** An *A/B test* randomly assigns visitors to version A or B and compares a pre-chosen metric; random assignment lets a difference larger than chance be attributed to the change (Kohavi, Tang and Xu).

Rules: choose the metric first; fix the sample size or duration in advance; check that group sizes match the plan.

Small differences can only be measured on many people, and real users are the right ones. Book: Section 7.10.1.

The Two-Proportion z-Test

A: 157 orders of 1,210 visitors (12.98%); B: 198 of 1,187 (16.68%).

Pooled $\acute{p} = 355/2397 = 0.1481$; the statistic (Equation (7.6)):

$$z = \frac{\hat{p}_B - \hat{p}_A}{\sqrt{\acute{p}(1-\acute{p})\left(\frac{1}{n_A} + \frac{1}{n_B}\right)}} \tag{7.6}$$

Standard error 0.01451, $z = 2.55$, two-sided $p \approx 0.011$.

The 95% interval of the difference (unpooled): about +0.9 to +6.6 points.

If the labels made no difference, a gap this large would appear about once in a hundred tests. Book: Section 7.10.2.

Sample Size Fixed in Advance

Visitors per version to detect $p_1 \to p_2$ at the 5% level with 80% power (Equation (7.7)):

$$n = \frac{\left(z_{\alpha/2}\sqrt{2\,\acute{p}(1-\acute{p})} + z_{\beta}\sqrt{p_1(1-p_1)+p_2(1-p_2)}\right)^2}{(p_2-p_1)^2} \tag{7.7}$$

Halving the difference to detect quadruples the sample.

Φ is computed with Abramowitz–Stegun 7.1.26 (error below 1.5×10^{-7}) because Java has no built-in error function.

Same approximation in both languages, identical output. Book: Sections 7.10.3 and 7.10.4.

The A/B Result, Honestly

```
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
```

B very likely better; but 2,162 per version were needed for a 3-point lift and the warung had about 1,200 — **underpowered, wide interval**.

A 1-point lift needs 18,330 per version: months of traffic — judge small refinements by usability tests instead.

> **WARNING.** Do not peek and stop. Checking daily and stopping at the first $p<0.05$ declares far more false winners than 5%.

Report "B is very likely better; by how much we do not know precisely." Book: Section 7.10.4.

Evaluation in 2026

Remote tests reach people outside the city; **unmoderated** tests reach many cheaply; both lose the right question at the right moment.

AI features: transcripts, hesitation highlights, summaries, even AI moderators.

Use AI to find moments to watch, not to replace watching; check summaries against recordings; recordings are personal data under UU PDP.

An AI summary "users found it easy" would have hidden P3 and P6 exactly as the mean did. Book: Section 7.11.

Using It for Real

The one-page plan: purpose; inspection first; 5–8 per round and per group; realistic tasks; measures with intervals; plot every score; iterate; A/B only for high-traffic questions with Equation (7.7) first.

Puskesmas kiosk: same tasks on the real kiosk, older patients over-represented; the screen log gives times.

Ojek app: drivers, own phones, outdoors, one hand.

ATM redesign: heuristic review against accessibility guidelines, tests with blind and older customers using the audio jack, completion per transaction from machine logs.

Book: Section 7.12.

HCI in Practice: Profitina

A score that must be understood: tasks like “find out why your score dropped this week” show whether owners move from number to action.

Release gate: the full automated suite, well over a thousand tests including JavaScript UI guards, must be green; the staged release reports success only after the health check answers.

Automated tests keep yesterday’s usability fixes from breaking; usability tests find tomorrow’s problems.

AI answers measured with users: each local-model release is compared with the current one before it is served.

Profitina advises and the user decides; evaluation shows whether the advice helps. Book: Section 7.13.

Common Mistakes

Testing with friends; tasks that give away the answer; helping during the test.

“8 out of 8” reported as 100%; arithmetic means of skewed times.

SUS read as a percentage; negative items not reversed.

“Five users are always enough.”

Peeking at an A/B test; treating significance as precision; AI summaries instead of watching.

Book: Section 7.14.

Worked Problems: SUS and Five Users

Exercise 7.1 — P3 answered 3, 4, 2, 4, 3, 3, 3, 4, 2, 3.

Odd items (3, 2, 3, 3, 2) contribute $2+1+2+2+1=8$; even items (4, 4, 3, 4, 3) contribute $1+1+2+1+2=7$; $2.5\times 15=37.5$, as printed.

Exercise 7.4 — participants for 90% of problems: $n \geq \ln 0.10 / \ln(1-p)$.

$p=0.31$: $-2.3026/-0.3711=6.2$, so **7**; $p=0.15$: $-2.3026/-0.1625=14.2$, so **15**.

The second answer is why “five users” cannot be a fixed rule: rarer problems double the need.

Book: Section 7.17, Exercises 7.1 and 7.4.

Worked Problem: A Test That Is Not Significant

Exercise 7.6 — two menu layouts: A 96 of 800 (12.00%), B 108 of 790 (13.67%).

Difference +1.67 points, 95% interval −1.62 to +4.96, $z = 0.996$, $p = 0.32$: not significant.

Honest conclusion: “no evidence of a difference *with this sample*”, not “the layouts perform the same”.

Next step: run until the planned sample is reached, or judge the layout with a usability test.

Absence of evidence is not evidence of absence, especially in an underpowered test. Book: Section 7.17, Exercise 7.6.

Summary

1. Usability: effectiveness, efficiency, satisfaction for specified users, goals, context.

2. Heuristic evaluation with severity ratings (Table 7.1).

3. Think-aloud tests with goals in users' words; the facilitator stays quiet.

4. Completion with Equation (7.1), times with Equation (7.2); T3 succeeded for 4 of 8.

5. SUS (Equation (7.3)) and UMUX-Lite (Equation (7.4)): mean 70.3 (C), but P3 and P6 in F (Figure 7.1).

6. Equation (7.5): five to eight per round per group (Figure 7.3).

7. A/B with Equation (7.6) and Equation (7.7): 12.98% → 16.68%, $p = 0.011$, interval +0.9 to +6.6.

Book: Section 7.15.

Next Time

Chapter 8 — Visual Design, Design Systems and Information Architecture: Gestalt, type and grids, WCAG 2.2 contrast from hex codes, design tokens, Material 3 Expressive, Liquid Glass, Fluent 2, and card sorting with clustering.

Before then: work Exercises 7.1, 7.2, 7.4 and 7.6; run `usability_metrics` and `ab_test` in both languages.

The bank app of Chapter 8 will be checked with the same eye for people at the edges. Book: Sections 7.16 and 7.17.

Lecture 8 — Visual Design, Design Systems and Information Architecture

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 8

Lecturer: Irfan Subakti — yifana@gmail.com

Today we make structure visible and keep it that way: Gestalt grouping and hierarchy, type and grids, contrast computed from hex codes, design tokens and the platform design languages of 2026, and a menu built from how customers group things. Book: Chapter 8, Sections 8.1–8.16.

What This Meeting Covers

1. The case — a rural bank's app that customers cannot read or navigate.

2. Seeing groups — proximity, similarity, common region, closure; visual hierarchy and the squint test.

3. Type and layout — size, line length, a modular scale, spacing and grids.

4. Color and contrast — WCAG 2.2, relative luminance, the contrast ratio, a checker that reads W3C design tokens.

5. Design systems — tokens, components, guidelines; Material 3 Expressive, Liquid Glass, Fluent 2.

6. Information architecture — organization, labels, navigation, search; card sorting and clustering; tree tests.

The programs are `contrast` and `cardsort`, in Java and Python with identical output. Book: Chapter 8.

A Rural Bank’s App

BPR Tani Makmur, East Java: farmers, market traders, teachers, retired civil servants; many over fifty.

Complaints every week: “the letters are too small”, “I can’t read the white on the green”, “where do I pay the installment?”

The home screen: fifteen equal icons in fifteen bright colors, mostly unlabeled; the balance in small grey text.

Nothing is wrong in isolation; together, every item shouts equally, so nothing is heard.

Three steps: make structure visible and check contrast; write it down as a small design system; rebuild the menu from a card sort.

The card sort will show that customers file the loan installment under bills. Book: Section 8.1.

Three Ideas

> **DEFINITION.** *Visual design* arranges type, color, space and imagery so that structure and priorities are visible at a glance. *Information architecture* organizes, labels, navigates and searches content and functions. A *design system* is a documented, shared set of decisions: tokens, components and rules.

Visual design: *what do I see first, and what goes with what?*

IA: *where is it, and what is it called?*

Design system: *how do we make the same decision the same way every time?*

A beautiful screen with the wrong structure fails; so does a sound structure nobody can read. Book: Section 8.2.

Gestalt: How People See Groups

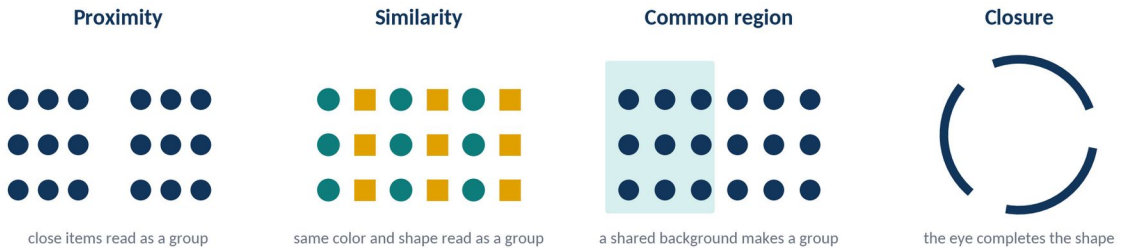


Figure 8.1 Four Gestalt principles used in interface design: proximity, similarity, common region and closure.

Figure 8.1: the visual system groups elements before we consciously look (a century of research, reviewed by Wagemans et al., 2012).

Book: Section 8.3.1 and Figure 8.1.

Using the Four Principles

Proximity: gaps between groups clearly larger than gaps inside them.

Similarity: all tappable things look alike; random icon colors destroy grouping.

Common region: a card's background groups its contents more strongly than spacing alone.

Closure: a list cut off at the bottom says "there is more below".

The bank's new home: balance plus the two everyday actions in one card; rarer functions in a second card; settings and help in a separate tab; only tappable things are teal.

Similarity cuts both ways: nothing that is not tappable may look tappable. Book: Section 8.3.1.

Visual Hierarchy and the Squint Test

Hierarchy = the order in which the eye takes in a screen: size, weight, color, position, isolation.

The visual order should match the importance for the most common task: balance first, then transfer and pay.

Squint test: blur the screen. Old: fifteen equal blobs. New: a large number, two buttons, a quiet list.

If the blurred screen does not show the order of importance, the sharp one does not either. Book: Section 8.3.2.

Type That Can Be Read

Size: body text at least 16 CSS px on phones; respect Dynamic Type and Android font scale.

Line length: about 45–75 characters.

Emphasis: weight, not long runs of capitals.

Numbers: *Rp1.250.000*, the format customers use — not *1250000* or *IDR 1,250,000*.

For a bank, numbers are the most important text on the screen. Book: Section 8.4.1.

A Modular Type Scale

Choose a base b and ratio r; use only (Equation (8.1)):

$$s_k = b \cdot r^k, k = \ldots, -1, 0, 1, 2, \ldots \tag{8.1}$$

$b = 16$, $r = 1.25$: 12.8, 16, 20, 25, 31.25, 39.06 px → the bank uses 13, 16, 20, 25, 39.

Each step is 25% larger: seen as a different level, never as a mistake.

Sizes that differ by one or two pixels look like errors, not levels. Book: Section 8.4.2.

Space, Grids and Targets

A spacing scale in multiples of 4 or 8 px: 8 px inside a group, 24 px between groups, 16 px at the edges.

A phone grid: four columns, 16 px margins and gutters.

Targets: WCAG 2.2 SC 2.5.8 requires at least 24 × 24 CSS px (or enough spacing); platforms recommend about 44–48 for primary controls.

Space is how proximity is expressed. Book: Section 8.4.3.

Why Contrast Is Measured

WCAG 2.2 (W3C Recommendation since October 2023): **4.5:1** normal text, **3:1** large text (SC 1.4.3); **3:1** for control boundaries, icons and focus indicators (SC 1.4.11).

October 2025: WCAG 2.2 approved as **ISO/IEC 40500:2025**.

WCAG 3.0 remains a **Working Draft** (new draft for review, September 2026); today's laws point to WCAG 2.x.

SC 1.4.1: color is never the *only* signal — the bank adds "Berhasil"/"Gagal" and an icon to green and red.

Red-green color vision deficiency affects several percent of men in many populations. Book: Section 8.5.1.

From Hex to Luminance

Linearize each sRGB channel (Equation (8.2)):

$$c = \frac{c_8}{255}, \qquad C = \begin{cases} c/12.92 & \text{if } c \le 0.04045 \\ \left(\frac{c+0.055}{1.055}\right)^{2.4} & \text{otherwise} \end{cases} \tag{8.2}$$

Weight the channels, green most (Equation (8.3)):

$$L = 0.2126\,R + 0.7152\,G + 0.0722\,B \tag{8.3}$$

The sRGB encoding stores brightness non-linearly; luminance needs linear light. Book: Section 8.5.2.

The Contrast Ratio, Worked

With L_1 lighter and L_2 darker (Equation (8.4)):

$$\mathrm{CR} = \frac{L_1 + 0.05}{L_2 + 0.05} \tag{8.4}$$

Navy `#12355B` = (18, 53, 91): linear 0.00605, 0.0356, 0.1046 → $L = 0.0343$; on white \$1.05/0.0843 = \$ **12.46:1**.

White on amber `#E0A100`: $L = 0.4134$ → \$1.05/0.4634 = \$ **2.27:1** — fails even for large text.

Scale from 1:1 to 21:1; 0.05 models room light reflected by the screen.

Amber is for highlights and icons on navy, not a background for white text. Book: Section 8.5.2.

Colors as W3C Design Tokens

Design Tokens Format Module **2025.10**, first stable version, October 28, 2025.

JSON; each token has `$value` and `$type`; a color value names its color space and components, with an optional `hex`.

```
"navy": {
  "$value": {
    "colorSpace": "srgb",
    "components": [0.0706, 0.2078, 0.3569],
    "hex": "#12355B"
  }
}
```

`pairs.csv` lists every pair actually used on screen, with its use: `text`, `large`, `ui`.

One token file can feed the checker, the CSS and the Android resources. Book: Section 8.6.1.

The Contrast Checker

`contrast.py` / `Contrast.java`: compute Equation (8.2), Equation (8.3), Equation (8.4) for each pair; compare with the minimum for its use.

For a failing pair: darken the *darker* color in 1% steps until the minimum is reached — the difference in luminance always grows.

Java has no JSON parser: a regular expression reads this flat file; use a JSON library for anything more general.

The suggestion shows how far a color must move; a designer may still choose another hue. Book: Sections 8.6.2 and 8.6.3.

Checker Output

```
grey       grey-light  text    4.39:1  FAIL (needs 4.5:1)
                               try foreground #69707D instead of #6B7280: 4.53:1
white      green-old   text    2.78:1  FAIL (needs 4.5:1)
                               try background #3A853D instead of #4CAF50: 4.56:1
teal       teal-light  text    4.18:1  FAIL (needs 4.5:1)
                               try foreground #0E7676 instead of #0F7C7C: 4.52:1
6 pair(s) fail WCAG 2.2 AA
```

The old white-on-green buttons: 2.78:1 — “I can’t read the white on the green”.

Grey passes on white (4.83) but fails on the light-grey card (4.39).

Teal on light teal fails by a hair; `#0E7676` fixes it invisibly.

Book: Section 8.6.4.

The Pairs as Swatches

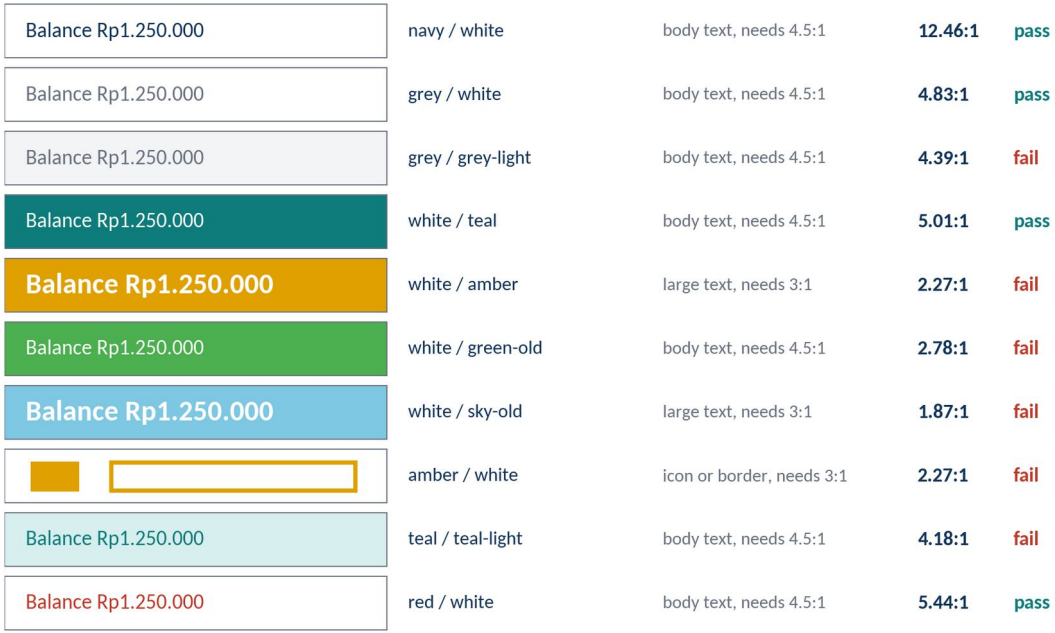


Figure 8.2 The ten color pairs of the bank app as swatches, with their use, contrast ratio and WCAG 2.2 AA result.

Figure 8.2: six of ten pairs fail.

> **WARNING.** Check the pairs actually used — text on cards, images, disabled and hover states — not only main text on the main background.

Book: Section 8.6.4 and Figure 8.2.

Design Systems and Tokens

Three layers: **tokens** (values), **components** (with states: normal, pressed, disabled, focused, error), **guidelines**.

Primitive tokens name values; **semantic** tokens name purposes and point to primitives; components use semantic tokens only.

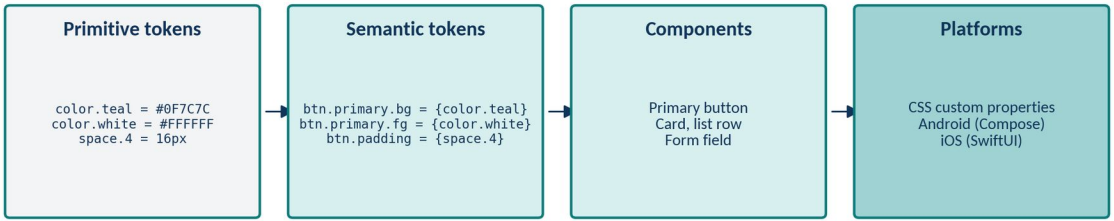


Figure 8.3 Design tokens in a design system: primitive tokens hold values, semantic tokens name purposes, components use semantic tokens, and build tools turn one token file into code for each platform.

Figure 8.3: darken one primitive teal and every component follows; a high-contrast theme repoints the semantic tokens.

Reference implementations of the 2025.10 format: Style Dictionary, Tokens Studio, Terrazzo; adopters include Figma and Penpot. Book: Section 8.7.1 and Figure 8.3.

The Design Languages of 2026

Material 3 Expressive (May 2025): varied shapes, bolder type, richer color, physics-based motion; Google reports 46 studies, 18,000+ participants, key elements spotted up to 4× faster, older users as fast as younger — Google's findings about its own system.

Liquid Glass (June 9, 2025; iOS 26 and siblings): a translucent material that refracts and adapts to content; iOS 26.1 added *Clear* versus *Tinted* (more opacity and contrast) besides Reduce Transparency.

Fluent 2 (Microsoft): open tokens, components and accessibility guidance across Windows, web, iOS and Android.

On a changing background, guarantee contrast with a backing layer or tint, and test with real content. Book: Section 8.7.2.

What a Small Team Should Do

A *small* system: about twenty colors, a type scale, a spacing scale, three radii, a dozen components with states, one page of rules.

Build on the platform's conventions so customers find things where their phone taught them to look.

The bank's first rule: "every text and icon pair passes `contrast.py` before it ships".

Give the system an owner.

A system without an owner decays into a folder of old files. Book: Section 8.7.3.

Information Architecture

Organization: by task, topic, audience, alphabet, time — confusion comes from mixing schemes unnoticed.

Labeling: a label is a promise; “Layanan” promises nothing, “Bayar tagihan” something recognizable.

Navigation: bottom tabs for three to five main destinations, lists and cards inside, a clear way back.

Search: fails when the search words are the designers’, not the users’.

The old app: one flat grid, internal labels (“Mutasi”, “Inquiry”, “Setor Angsuran”), no search.

Rosenfeld, Morville and Arango’s four systems. Book: Section 8.8.1.

Card Sorting and Tree Testing

Open card sort: participants group cards and name the piles — discovers how people think and the words they use.

Closed card sort: cards into given categories — checks a proposed structure.

15–30 participants of one audience usually give a stable picture (Spencer).

Tree test: only the text of a menu hierarchy; “where would you pay your loan installment?”

The bank: 20 customers at two branches, 15 cards; `cardsort.csv` records participant, card and pile name.

Card sorts propose; tree tests check. Book: Section 8.8.2.

Similarity and Average Linkage

Co-occurrence similarity of cards i and j (Equation (8.5)):

$$S_{ij} = \frac{n_{ij}}{N} \tag{8.5}$$

Start with every card alone; repeatedly merge the two most similar clusters, using average linkage (Equation (8.6)):

$$S(A, B) = \frac{1}{|A||B|} \sum_{i \in A} \sum_{j \in B} S_{ij} \tag{8.6}$$

Average linkage avoids single linkage's chaining and complete linkage's strictness.

Pile names do not enter the similarity; only whether two cards share a pile. Book: Sections 8.9.1 and 8.9.2.

The Card-Sort Program

`cardsort.py` / `CardSort.java`: similarities, merges named G1, G2, …, and groups at a cut.

```
  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
```

Written for clarity, not speed: fine for dozens of cards; a tolerance of `1e-12` makes Java and Python break ties identically.

Book: Sections 8.9.3 and 8.9.4.

The Dendrogram

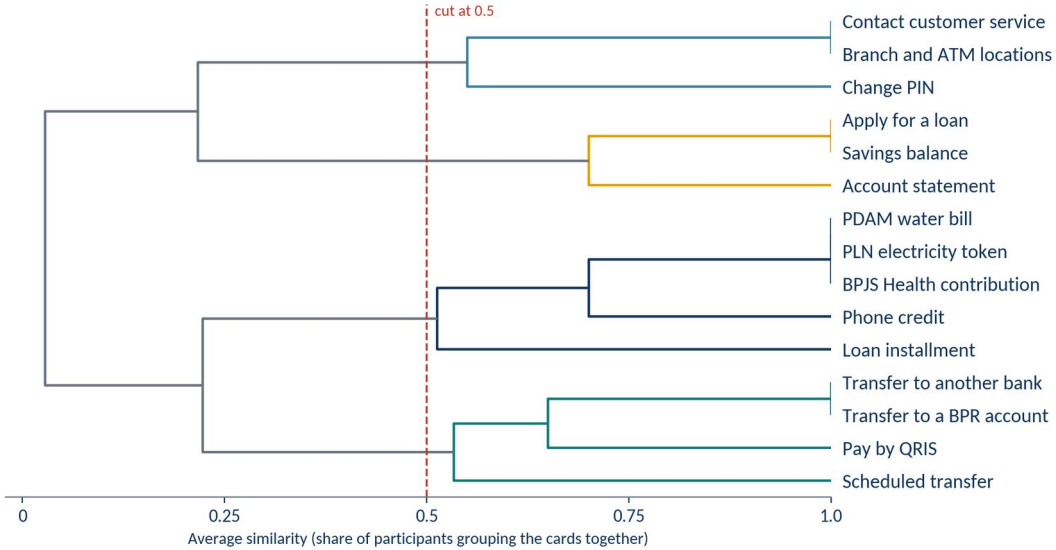


Figure 8.4 Dendrogram of the open card sort (20 participants, 15 cards, average linkage); cutting at a similarity of 0.5 gives four groups, and the loan installment joins the bills.

Figure 8.4: the further right two cards join, the more participants agreed.

Book: Section 8.9.4 and Figure 8.4.

Reading the Card Sort

Four groups at 0.5: transfers (with QRIS and scheduled transfers); savings and loans; help; **bills — with the loan installment** (0.51).

The installment was grouped with the water bill by 55% and with the savings balance by 45%: an installment is something you *pay*.

Decisions: four tabs in customers' words — *Kirim uang*, *Bayar*, *Tabungan & pinjaman*, *Bantuan*; the installment under *Bayar* **and** linked from the loan page; "Mutasi" renamed "Riwayat transaksi", kept as a search keyword.

Tree test: 18 of 20 found the installment on the first try.

> **KEY INSIGHT.** A card sort shows how customers think: clusters propose groups, pile names propose labels, split items call for cross-links. Designers decide; tree tests check.

Book: Section 8.9.4.

Testing the Programs

Contrast: 21:1 for black on white in either order, 1:1 for identical colors; the token reader; navy on white 12.46; the fix reaches its target.

Card sort: fourteen merges for fifteen cards; four groups at 0.5 with the installment among the bills; two groups at 0.2.

```
10 checks passed
```

Book: Section 8.10.

Worked Problems

Exercise 8.3 — amber icons on a navy bar: $(0.4134+0.05)/(0.0343+0.05)=5.50$ → passes. Contrast belongs to the *pair*, never to one color.

Exercise 8.5 — cut at 0.6: seven groups (scheduled transfer, installment and Change PIN leave); at 0.7: eight (QRIS leaves too). These four cards are candidates for cross-links and clearer labels.

Exercise 8.6 — single linkage merges the transfer and bill clusters at 0.55 instead of 0.22: *chaining* through the scheduled transfer.

Book: Section 8.16, Exercises 8.3, 8.5 and 8.6.

Using It for Real

Visual: three most important things per screen, in visual order; proximity and common region; one color for “tappable”; modular scale; spacing scale; squint.

Color: tokens; every used pair; `contrast.py` on every change; never color alone; test translucent and image backgrounds with real content.

System: small, platform-based, owned.

IA: open card sort with 15–30 users; Equation (8.5) and Equation (8.6); labels from pile names; cross-list split items; tree test before screens.

Examples: a puskesmas website (“imunisasi”, “posyandu”, “KIA” together), a warung menu board, ojek settings tested with drivers.

Book: Section 8.11.

HCI in Practice: Profitina

Dashboard as a cockpit: one request gathers notifications, followed channels, rank and activity, each clickable to its section — organized by *what needs me now*.

Counters that mean something: "N new" counts only top-level posts; counting replies would inflate the promise.

Architecture for machines too: a bilingual site that publishes `/llms.txt` for AI answer engines, alongside schema.org structured data.

Clear organization and honest labels serve people and the AI systems that read on their behalf. Book: Section 8.12.

Common Mistakes

Everything equally loud; random icon colors; gaps that do not group; too many font sizes.

Contrast judged by eye; color as the only signal; platform style without its accessibility settings.

A design system without tokens or an owner.

Organizing by the org chart; internal labels; forcing split items into one place.

Book: Section 8.13.

Summary

1. Gestalt grouping and hierarchy make structure visible (Figure 8.1).

2. Type from a modular scale (Equation (8.1)); space from a 4- or 8-px scale and a grid.

3. WCAG 2.2 (ISO/IEC 40500:2025): 4.5:1 text, 3:1 large text and UI; contrast from Equation (8.2), Equation (8.3), Equation (8.4); WCAG 3.0 still a draft.

4. The checker read W3C tokens: six of ten pairs failed (Figure 8.2).

5. Tokens, components, guidelines (Figure 8.3); Material 3 Expressive, Liquid Glass, Fluent 2.

6. Card sort with Equation (8.5) and Equation (8.6): four groups, the installment under bills (Figure 8.4).

Book: Section 8.14.

Next Time

Chapter 9 — Interaction Styles: GUI, Touch, Voice, Gesture.

Before then: work Exercises 8.1, 8.3, 8.5 and 8.7; run `contrast` on your own app's colors and a small card sort with five friends.

The four-tab bank app returns when we design for touch and thumbs. Book: Sections 8.15 and 8.16.

Lecture 9 — Interaction Styles: GUI, Touch, Voice, Gesture

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 9

Lecturer: Irfan Subakti — yifana@gmail.com

Today we look at the ways a person tells a computer what to do — commands, menus, forms, direct manipulation, natural language — and at the channels that carry them: pointer, touch, gesture, voice and gaze. We build one warung order core with three front ends and measure a kiosk in a real browser. Book: Chapter 9, Sections 9.1–9.14.

What This Meeting Covers

1. **The warung counter** — five ways to order one glass of tea, and what each asks of the customer.
2. **The theory** — two gulfs, direct manipulation, events and the speed of feedback.
3. **Touch** — Fitts's law, FFitts law, target sizes in WCAG 2.2 and on Apple and Android.
4. **Gestures and voice** — invisible commands, uncertain listening, and when to ask.
5. **Building and measuring** — one order core, a command line, a typed voice line, a touch kiosk in Flask and Java, and a target-size audit with Playwright.

Every program exists in Java and Python, and every output shown is real. Book: Chapter 9.

Bu Tini’s Counter

A warung near a campus in Surabaya: five dishes, a noon crowd of students, drivers and office workers.

“*Es teh dua, Bu!*” from the door — voice, and Bu Tini’s “*Dua, ya?*” is a spoken confirmation.

A newcomer points at the laminated card — menu selection by gesture; the office writes an order slip — a form.

Her nephew Dimas wants a tablet kiosk. How big must buttons be? What should it do when it does not understand? What about older customers?

81.72% of Indonesians — about 235 million — used the internet in 2026 (APJII), and every one of them meets many interaction styles each day. Book: Section 9.1.

What an Interaction Style Is

How do I say it? Typing a word, choosing from a list, filling blanks, acting on objects, speaking.

How do I see the result? Text, a changed picture, a sound, a vibration.

Five primary styles (Shneiderman): **command language, menu selection, form fill-in, direct manipulation, natural language**.

Touch, gesture, voice and gaze are **modalities** — channels that carry the styles.

A phone app is mostly direct manipulation and menus through touch; a voice assistant is natural language through speech; Vision Pro is direct manipulation through eyes and a pinch. Book: Section 9.2.1.

A Map of the Styles

Table 9.1 The five primary interaction styles and where each is strong and weak.

Style	The user…	Strong when	Weak when
Command language	types fixed syntax	experts, repetition	novices, syntax errors
Menu selection	picks from a list	occasional users	many options
Form fill-in	fills labeled blanks	structured data	unclear fields
Direct manipulation	acts on visible objects	seeing and undoing	abstract actions
Natural language	speaks or writes freely	busy hands and eyes	ambiguity, noise

Table 9.1: the deciding question is what the user must remember and what the user must see. Recognition (menus) is easier than recall (commands); recall is faster for experts. Book: Section 9.2.2 and Table 9.1.

One Glass of Tea, Five Ways

Command: `add esteh 2` — 13 keystrokes; must know the code, the verb and the order.

Menu: press 1, 1, 2 — three presses; read, but remember nothing.

Form: write 2 on the “Iced tea” line — one field, and the whole menu visible.

Direct manipulation: tap “Iced tea” twice, watch “2 x Iced tea” appear; minus undoes.

Natural language: say “*es teh dua*” — three words, but now the system must *understand*.

Styles that ask less of the user’s memory ask more of the screen; the style that asks least of the user asks most of the system. Book: Section 9.2.3.

Two Gulfs, and Direct Manipulation

Gulf of execution: from “I want two teas” to what the system lets me do.

Gulf of evaluation: from what the system shows to “did it take my order?”

Direct manipulation narrows both: objects stay visible; actions are physical, rapid, incremental and reversible; effects appear at once.

Its price: screen space, and abstract actions (“every Friday, ten teas”) are awkward.

Norman’s gulfs from Chapter 1 explain why commands feel hard to novices and why spreadsheets and ride-hailing maps feel easy. Book: Section 9.3.1.

Events, State and the Speed of Feedback

Every GUI, touch or voice interface waits for **events**, runs a **handler**, changes **state**, redraws.

The kiosk: state = the cart; events = add, remove, pay. The voice line adds one state: an order *waiting for confirmation*.

0.1 s feels instantaneous; **1 s** keeps the flow of thought; **10 s** is the limit of attention.

Past 1 s show activity; past 10 s show progress and let the user do something else.

The three limits go back to Miller (1968) and are summarized by Nielsen; they explain why good apps react at once and do slow work afterwards. Book: Section 9.3.2.

Fitts's Law

$$MT = a + b \log_2\left(\frac{D}{W} + 1\right) \tag{9.1}$$

MT: time to hit a target of width W at distance D; a, b measured per device and person.

The logarithm is the **index of difficulty** ID, in bits.

Halving the width costs about as much as doubling the distance: **small targets are slow targets**.

Equation (9.1) is the Shannon formulation used in HCI; Chapter 2 derived it for the mouse. Book: Section 9.3.3.

A Finger Is Not a Mouse Pointer: FFitts Law

$$I D_f = \log_2\left(\frac{D}{\sqrt{2\pi e\left(\sigma^2 - \sigma_a^2\right)}} + 1\right) \tag{9.2}$$

A finger touches an area several millimeters wide and hides its own target.

Bi, Li and Zhai: touch has an **absolute imprecision** σ_a that care cannot remove.

Below a certain size, a smaller target does not just slow the user — it causes **errors**.

Equation (9.2) keeps only the speed-related spread in the index; the finger's own spread is why every touch platform sets a minimum target size. Book: Section 9.3.3.

Worked Example: the Language Link

The kiosk's first version drew its language link 13 px high; menu buttons are 64 px. Distance about 300 px.

Link: $ID = \log_2(300/13 + 1) = \log_2(24.08) = 4.59$ bits.

Button: $ID = \log_2(300/64 + 1) = \log_2(5.69) = 2.51$ bits.

The link is almost **two bits harder**, and close to the finger's own scatter.

Worked by hand with Equation (9.1); the audit later finds the same link automatically. Book: Section 9.3.3, Example.

Minimum Target Sizes, September 2026

Table 9.2 Minimum sizes for touch and pointer targets as of September 2026.

Source	Minimum target	Level
WCAG 2.2, SC 2.5.8	24 × 24 CSS px, or spacing exception	AA
WCAG 2.2, SC 2.5.5	44 × 44 CSS px	AAA
Apple HIG	44 × 44 pt	guideline
Material Design 3	48 × 48 dp (≈ 9 mm)	guideline

Table 9.2: WCAG 2.2 became ISO/IEC 40500:2025 in October 2025. The 2.5.8 spacing exception lets a small target pass if a 24 px circle on it touches no other target. Book: Section 9.3.3 and Table 9.2.

Gestures: Powerful and Invisible

Swipe to delete, pinch to zoom, long press for a menu — fast for those who know, invisible for those who do not.

No **signifier** says “you can swipe here”.

Rules: a gesture is a **shortcut, not the only way**; follow platform conventions; do not exclude people.

WCAG 2.2: **2.5.1 Pointer Gestures** (single-pointer alternative), **2.5.7 Dragging Movements** (new: tap alternative to dragging).

A pinch-only map or drag-only list fails; plus/minus and up/down buttons fix it. Book: Section 9.3.4.

Voice: Natural, and Ambiguous

Suits busy hands and eyes, and many older and low-vision users.

Since September 4, 2026 Gemini gradually replaces Google Assistant on Android phones, watches and Android Auto.

Three problems: **no visible menu**, **uncertainty** (“*es teh dua*”: two teas, or tea for table two?), **fleeting** speech.

Answers: confirm in proportion to risk and doubt; say what can be said; keep a way back.

CHI 2026 research explores voice assistants that help adults with intellectual disabilities act independently. Book: Section 9.3.5.

Implicit or Explicit Confirmation

$$c = \frac{k}{m} \tag{9.3}$$

Implicit: repeat and go on — “Two iced teas — anything else?”

Explicit: stop and ask — “Did you mean one coffee? Say yes to confirm.”

Our confidence c: understood words k over words heard m; below 0.6, ask.

Costly actions (paying, deleting): **always** explicit.

Equation (9.3) is an explainable stand-in for a recognizer’s own confidence. A voice interface that never asks is guessing; one that always asks is tiring. Book: Section 9.3.5.

Today’s Design Languages and Space

Liquid Glass (iOS 26, September 15, 2025): translucent controls that refract the content behind them.

Material 3 Expressive (May 2025): 46 studies, 18,000+ participants; key elements spotted up to 4× faster.

Fluent 2: Microsoft’s shared system for Windows and Microsoft 365.

visionOS: look to target, pinch to act — Fitts’s law with the eyes as pointer.

Follow the platform’s controls and gestures; check glass and color for legibility (Chapter 11 measures contrast). Book: Section 9.3.6.

A Worksheet for Choosing a Style

1 Users · 2 Context · 3 Task shape — who, where, how many options, concrete or abstract.

4 Primary style · 5 Second path — the style that fits, and the alternative for those it fails.

6 Feedback · 7 Errors — what is seen within 0.1/1/10 s; how mistakes are seen and undone.

8 Check — measure target sizes, gesture alternatives and contrast.

Filled for the kiosk: touch kiosk with big buttons, typed or spoken orders with confirmation, and the helper at the counter as a second path. Book: Section 9.4 and Table 9.3.

One Core, Three Front Ends

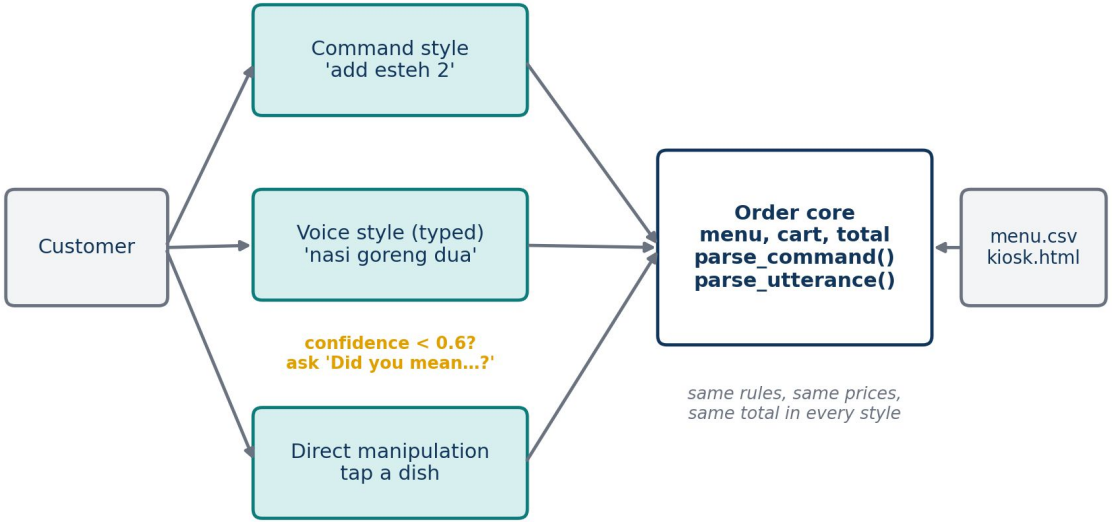


Figure 9.1 One order core with three front ends that share the menu, the cart and the total.

Figure 9.1: the front ends only translate between a person and the core; prices, rules and totals are the same in every style, and new styles can be added without touching the core. Book: Section 9.5.

The Voice Parser in Three Lines of Thought

Scan words left to right; at each position take the **longest alias** (“nasi goreng” beats a shorter one).

A number word just before or just after the dish is its quantity.

Intent words (“*batal*”, cancel) switch to remove; fillers (“*tolong*”, “*ya*”) are understood but empty.

Listing 9.1 From `order_core.py`: the confidence of Equation (9.3) is the share of understood words.

```
    conf = sum(known) / len(known) if known else 0.0
    return intent, items, round(conf, 2)
```

Listing 9.1 ends the parser; the Java version in `OrderCore.java` rounds the same way. Book: Section 9.5.1.

Asking Before Acting

Listing 9.2 From `cli.py`: below the threshold the front end stores the order and asks.

```
            if conf < CONFIRM_BELOW:
                # Low confidence: ask before acting (explicit confirm).
                pending = (intent, items)
                print(f"  -> Did you mean {heard}? Say 'yes' to confirm.")
                continue
```

The stored order waits in `pending`; "*ya*", "*iya*" or "yes" completes it; anything else discards it.

No dish at all → "Sorry, which menu item? Please say its name."

Listing 9.2 turns Bu Tini's "Dua, ya?" into code. Book: Section 9.5.2.

Running the Session

```
> 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
```

Commands, a syntax error, a fully understood utterance (confidence 1.0), one with no dish (asked for a name), and "kopi item yang kentel banget" at $c = 2/5 = 0.4$*, confirmed with "ya". Java prints the same lines. Book: Section 9.6.*

The Kiosk and Its Audit

Plain HTML forms, post/redirect/get; works on the cheapest phone, by keyboard and with a screen reader.

Flask on port 4090, the JDK's `HttpServer` on 4091 — **byte-identical pages**.

`audit_targets.py` opens the kiosk in Chromium at 360 px, taps like a customer, measures every target.

```
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
```

The 99 × 13 px language link is below all three sizes; it may pass 2.5.8 by spacing, but it fails 44/48 and is the target older customers miss. Book: Sections 9.5.3, 9.5.4 and 9.6.

The Fix, Measured

One CSS rule: `.lang { display: inline-block; min-height: 48px; padding: 12px 16px; }`.

```
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
```

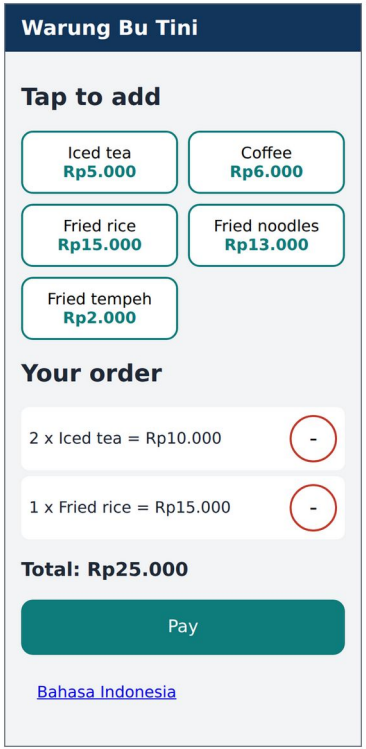


Figure 9.2 The final kiosk at 360 px after three taps, with 48 px touch areas everywhere.

Figure 9.2 is the screenshot the audit took of our own running kiosk. Book: Section 9.8.1.

Tests

`pytest`: totals, over-removal, numbers before and after dishes, the cancel intent, low confidence, bad quantity — 4 tests pass.

`OrderCoreTest.java`: the same seven checks, “all passed”.

Two checks by hand: both servers give the same MD5 for the empty page; the audit prints the same nine lines against both.

Book: Section 9.7.

Where the Parser Still Fails

"*es teh manis dua gelas*" → one iced tea at confidence 0.6: acted on, and **wrong**.

"*berapa totalnya*" → the suffix *-nya* hides the word *total*.

Confidence measures how much was understood, not whether it was right.

Indonesian customers mix affixes, Javanese and English — collect real utterances.

Exercises 9.4 and 9.5 fix both; paying must always be confirmed explicitly. Book: Section 9.8.2.

Using It for Real

Replace `menu.csv` with your dishes **and the words your customers say**.

A WhatsApp or Telegram bot is one more front end to `parse_utterance`.

Keep the helper at the counter: a kiosk should shorten the queue, not replace the person.

Audit at 320 and 360 px on the cheapest phone, in sunlight, with a thumb.

Book: Section 9.9.

HCI in Practice: Profitina

Dashboard as a cockpit: one request gathers notifications, channels, rank and activity; each item is one tap to its section — direct manipulation on a stream of events.

Optimistic posting: the composer clears at once and the post appears; if sending fails, the text is restored — feedback within 0.1 s, failures reversible.

Share to X is manual, per item, with a queue and daily quota — public actions stay under explicit control.

The content engine proposes drafts (natural language) and the user edits and decides (direct manipulation). Book: Section 9.10.

Common Mistakes

One style for everyone; gestures without alternatives.

Measuring targets by eye; confusing the spacing exception with comfort.

Guessing instead of asking; treating confidence as correctness.

Logic duplicated in the front ends; feedback slower than 0.1 s.

Book: Section 9.11.

Summary

Five primary styles, carried by touch, gesture, voice and gaze (Table 9.1).

Direct manipulation narrows both gulfs; feedback limits 0.1 / 1 / 10 s.

Fitts (Equation (9.1)) and FFitts (Equation (9.2)); targets 24 px AA, 44 AAA/Apple, 48 Material (Table 9.2).

Voice: confirm when unsure (Equation (9.3)) or when costly.

Book: Section 9.12.

Exercises to Try

9.1 Index of difficulty at 24, 44 and 48 px for $D = 300$ px.

9.3 Implicit or explicit: adding a tea, paying Rp61.000, cancelling everything.

9.4 Make "*es teh manis dua gelas*" give two teas.

9.7 Which WCAG 2.2 criteria does a pinch-only, drag-only app fail?

Full solutions are in the book. Book: Sections 9.13 and 9.14.

Next Time

Chapter 10 — Mobile and Responsive UX. An ojek base's booking page, designed on a desktop and nearly unusable on a phone.

Viewport and CSS pixels, mobile-first layouts, thumbs, phone forms, Core Web Vitals and a performance budget.

Before then: run the kiosk, audit it at 320 px, and try Exercises 9.1 and 9.4. Book: Chapter 10.

Lecture 10 — Mobile and Responsive UX

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 10

Lecturer: Irfan Subakti — yifana@gmail.com

Today we design for one hand, a narrow screen and an uncertain network. We read Indonesia's mobile statistics carefully, learn why a desktop page shrinks to a third on a phone, and rebuild an ojek base's booking page mobile-first, measuring it at four widths and on three networks. Book: Chapter 10, Sections 10.1–10.14.

What This Meeting Covers

1. Mas Yoga's phone — a booking page that worked on a monitor and failed under a tree.

2. Reading the numbers — APJII, StatCounter and Ookla measure different things.

3. Pixels and the viewport — why a page without one line becomes 37% of its size.

4. Layout, thumbs and forms — fluid grids, content-chosen breakpoints, thumb reach, input types and autocomplete.

5. Speed — Core Web Vitals, round trips, bandwidth and a performance budget, measured by our own programs.

Book: Chapter 10.

Mas Yoga’s Phone

Pak Darto’s ojek base on Jalan Kenari: a bench, a tree, four drivers; customers want to book ahead.

The nephew’s page looked tidy on a desktop: a driver table, a four-field form, Call and WhatsApp links.

On Mas Yoga’s phone: a tiny grey postcard, sideways scrolling, the letter keyboard for a phone number, a “Call” link too small for a thumb.

Bu Ani gave up and phoned — the page had made the phone call the only reliable interface.

Nothing was wrong in desktop terms; the page assumed a wide screen, a mouse, a desk and a fast wire. Book: Section 10.1.

Responsive and Mobile-First

Responsive: one page that adapts — fluid grids, flexible images, media queries (Marcotte, 2010).

Mobile-first: design the smallest, weakest case first, then *add* layout for larger screens (Wroblewski).

A phone is more than a small screen: one thumb, interruptions, sunlight, a variable network, a slow processor.

A page that is clear at 360 px is clear at 1280 px.

Book: Section 10.2.

Three Sources, Three Different Counts

APJII (May 2026): 81.72% of people, about 235 million, use the internet — it counts **people**.

StatCounter (August 2026): 38.98% of page views mobile, 60.59% desktop, 0.43% tablet; Android 77.07%, iOS 22.88% of mobile — it counts **page views on sites with its tracker**.

Ookla via Bloomberg Technoz (February 2026): median mobile download 59.18 Mbit/s, upload 17.97, latency 21 ms — a **median**, not the tail.

Always ask: *what* was counted, *where*, *how*?

Desktop page views can dominate while most people are on phones in apps; the median hides the slowest tenth. Book: Section 10.3.1.

CSS Pixels and Device Pixels

$$w_{css} = \frac{w_{dev}}{r} \tag{10.1}$$

r is the device pixel ratio (2, 3 or more on today's phones).

A phone 1080 device pixels wide with $r = 3$ gives 360 CSS pixels.

Designers work in CSS pixels — "a 360-pixel phone".

Equation (10.1): the browser maps one CSS pixel onto $r \times r$ device pixels. Book: Section 10.3.2.

The Viewport, and the Shrinking Page

$$s = \frac{w_{vp}}{w_{layout}} \tag{10.2}$$

$$h_{eff} = s \cdot h_{css} \tag{10.3}$$

Without `<meta name="viewport" content="width=device-width, initial-scale=1">` a phone lays the page out at **980 px** and zooms out.

Every size is multiplied by s of Equation (10.2); what the user meets is h_{eff} of Equation (10.3).

One line in the head decides whether the rest of the mobile work is visible at all. Book: Section 10.3.2.

Worked Example: Pak Darto’s First Page

360 px phone, layout 980 px: $s = 360/980 = 0.367$.

11 px small print → $0.367 \times 11 = 4.04$ px on the screen.

12 px “Call” link → $0.367 \times 12 = 4.4$ px as a target.

320 px phone: $s = 0.327$, and the print shrinks to 3.6 px.

About the height of a comma. Book: Section 10.3.2, Example.

Fluid Layouts and Breakpoints

Start with one column, relative units (`%`, `rem`, `fr`), `max-width` for long lines.

Media queries add layout when there is room: `@media (min-width: 700px) { … }`.

Container queries let a component adapt to its container — a card in a sidebar and in a main column.

Choose breakpoints where **the content** needs them, not at last year's phone widths.

For the booking page, one column reads well up to about 700 px; above that, cards side by side waste less space. Book: Section 10.3.3.

Hands and Thumbs

Hoober observed 1,333 people: **49%** one hand and thumb, **36%** cradled with the other hand's finger, **15%** two hands.

People change grip often, even within one task.

Frequent, important actions **within thumb reach** — lower and middle screen; rare or destructive ones further away.

The responsive page puts Call and WhatsApp in a bottom bar, each half the width, 52 px high.

Chapter 9's target sizes apply unchanged: at least 24 CSS px (AA), 44–48 for comfort. Book: Section 10.3.4.

Forms on Phones

Input type chooses the keyboard: `type="tel"` → number pad; `type="email"` → @ key.

`autocomplete` says what a field means: `name`, `tel`, `street-address` → one-tap filling.

`inputmode` picks a keyboard without changing the field, e.g. `numeric`.

Real `<label>` above each field, connected with `for`/`id`; placeholders vanish while typing.

WCAG 2.2 SC 1.3.5 Identify Input Purpose (AA) requires the purpose of personal-data fields to be machine-readable. Book: Section 10.3.5.

Core Web Vitals

LCP (Largest Contentful Paint) ≤ **2.5 s**.

INP (Interaction to Next Paint) ≤ **200 ms** — it replaced First Input Delay in March 2024.

CLS (Cumulative Layout Shift) ≤ **0.1**.

Judged “good” at the **75th percentile** of real visits.

Book: Section 10.3.6.

Round Trips, Bytes and a Budget

$$t \approx k \cdot RTT + \frac{8\,S}{B} \tag{10.4}$$

k round trips (DNS, TCP, TLS, request) of RTT seconds, then S kB at B kbit/s.

Lighthouse's mobile profile: RTT 150 ms, 1,638.4 kbit/s, CPU 4× slower — much slower than the median, deliberately.

Performance budget: decide the maximum weight so Equation (10.4) stays within 2.5 s on the chosen profile.

Two things a designer controls: how many round trips and how many bytes. Book: Section 10.3.6.

WCAG Criteria That Bite on Phones

1.4.10 Reflow (AA): usable at 320 CSS px without two-way scrolling.

1.4.4 Resize Text (AA): text to 200% without loss.

1.3.4 Orientation (AA): no lock to portrait or landscape unless essential.

1.3.5 Identify Input Purpose (AA) and **2.5.8 Target Size (AA)**.

320 px equals a 1280 px desktop window at 400% zoom, so reflow also serves low-vision desktop users. Book: Section 10.3.7.

The Mobile-First Checklist

Table 10.1 The mobile-first checklist with the first booking page's results.

Check	First booking page
Viewport line	Missing: page zoomed to 37%
Reflow at 320 px	Zoomed out instead
Body text ≥ 16 px	3.6–4.0 px effective
Targets ≥ 24 px (44–48)	3.9–4.4 px effective
Main actions in thumb reach	Tiny links at the top
Keyboards and autocomplete	Phone field is plain text
Weight within budget	2,492 kB: 12.8 s on the test profile

Table 10.1: most rows can be automated; thumb reach and sense-making need people (Chapters 5 and 7). Book: Section 10.4 and Table 10.1.

Two Templates, One Set of Words

`naive.html`: `body { width: 960px }`, 13 px font, a table, loose text before inputs, `type` text for the phone, 11 px links, no viewport line.

`responsive.html`: mobile-first base styles, 16 px, driver cards, labels above 48 px inputs, a fixed bottom bar, one media query at 700 px.

`strings.csv` holds English and Indonesian words; `pages.py` / `Pages.java` fill either template.

Flask on port 4100, the JDK's server on 4101 — identical bytes.

Book: Sections 10.5.1 and 10.5.2.

A Browser That Pretends to Be a Phone

`check_responsive.py` opens each page at 320, 360, 768 and 1280 px; below 700 px as a **mobile** context with touch. It reads `innerWidth`, `scrollWidth`, the smallest font, the smallest target and the phone field's type.

Listing 10.1 From `check_responsive.py`: sizes are scaled by Equation (10.2) before they are reported.

```
            scale = w / vw            # < 1 when the browser zooms the page out
            over = "yes" if sw > vw else "no"
```

Listing 10.1: the report shows what the eye and the thumb meet, as in Equation (10.3). Book: Section 10.5.3.

What the Check Found

```
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
```

The first page is laid out at 980 px on phones (text 3.6–4.0 px, targets 3.9–4.4 px), overflows sideways at 768 px, and has a plain-text phone field. The responsive page keeps 16 px text and 48 px targets at every width. Book: Section 10.6.

The Two Pages on a Phone

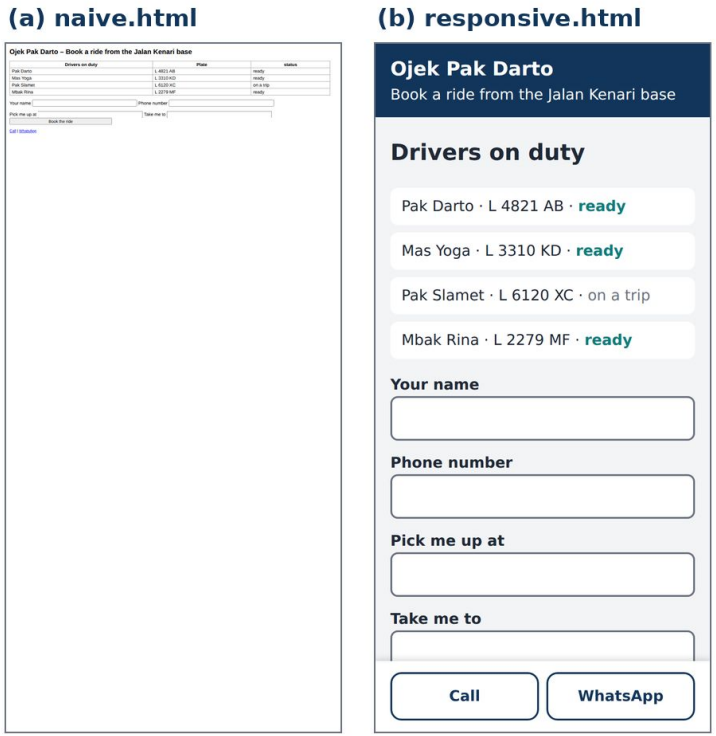


Figure 10.1 The two booking pages on a 360-pixel phone: the shrunken desktop page and the mobile-first page.

Figure 10.1: (a) laid out at 980 px and shrunk; (b) readable text, labeled 48 px fields, and the Call/WhatsApp bar in thumb reach. Book: Section 10.6.

The Load-Time Estimate

Listing 10.2 `budget.py`: Equation (10.4) with $k = 4$ round trips.

```python
def seconds(kb, mbps, rtt_ms):
    return ROUND_TRIPS * rtt_ms / 1000 + kb * 8 / (mbps * 1000)
```

```
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
```

Listing 10.2. Check by hand: $4 \times 0.150 + 8 \times 2492/1638.4 = 0.6 + 12.17 = 12.77$ *s. On the median the heavy page arrives in 0.42 s — why it seemed fine in the city. Book: Section 10.6.*

Six Estimates on One Chart

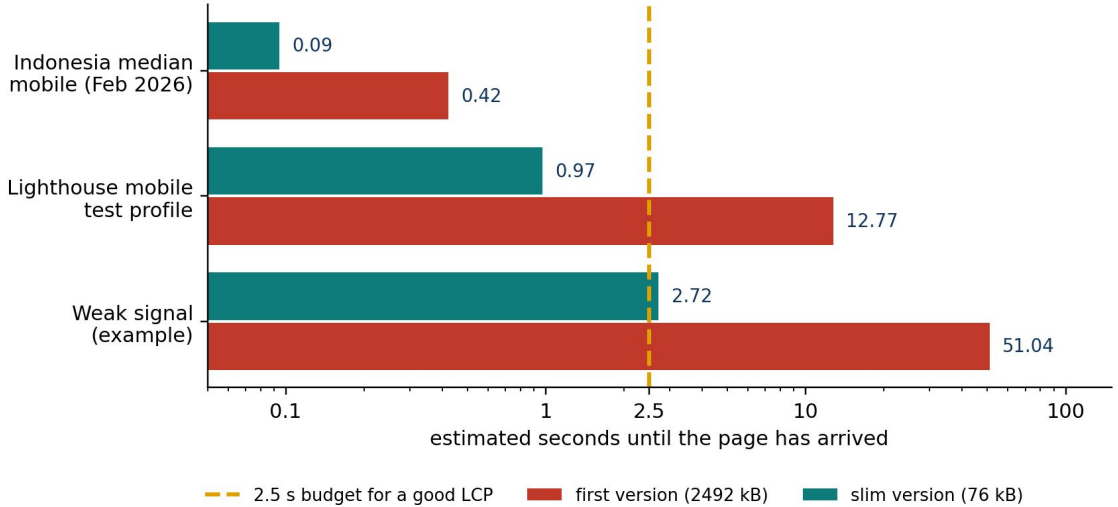


Figure 10.2 Estimated arrival time of the 2,492 kB and the 76 kB page on three networks; the dashed line is the 2.5 s budget.

Figure 10.2 uses a logarithmic time axis. Test on the slowest phone your users own, outdoors, on a weak connection. Book: Section 10.6.

Reading the Output Line by Line

Layout 980 at 320 and 360: the missing viewport line — the browser pretends to be a desktop and shrinks.

Overflow “no” on phones: not a success — the page fits only because it was zoomed out.

Overflow “yes” at 768: the fixed 960 px body is wider than a tablet window: a WCAG 1.4.10 failure in the making.

16.0 and 48.0 everywhere: the responsive page’s base font and field height survive every width.

A clean number can hide a bad reason; always ask why a check passed. Book: Section 10.6.

Respect the Data Plan

Many users buy data by quota; every unnecessary megabyte is the **user's** money.

A 2.5 MB page opened twice a day uses about 150 MB a month — for one booking page.

Pages that still work with images off serve users in data-saver mode.

A performance budget is a speed rule **and** a fairness rule.

Book: Sections 10.3.6 and 10.8.

Tests

`pytest` (3 tests): the formula against a hand value, $0.6+1.0=1.6$ s; both templates in both languages with no placeholder left; viewport and `tel` only on the responsive page.

`Ch10Test.java`: the same seven checks, “all 7 checks passed”.

Book: Section 10.7.

Making It Better

Weak signal: the slim page takes 2.72 s; the photo is 70 of 76 kB. Lazy-load it, use `srcset`, or drop it: without it $t \approx 1.2 + 0.12 = 1.32$ s.

Second visit: 1.2 s of the 2.72 s are round trips; HTTP caching and a small service worker skip them.

What checks cannot see: field order, wording, landscape in a dashboard holder — take the phone to the base and watch.

Book: Section 10.8.

Using It for Real

Design at 360 px, then 320, then add breakpoints where content stretches.

Always include the viewport line.

Write down a budget: on the Lighthouse profile, at most about **389 kB** for 2.5 s.

Run `check_responsive.py` on your own pages; use input types and autocomplete everywhere.

Book: Section 10.9.

HCI in Practice: Profitina

The **dashboard cockpit** fits a phone: one request, every item one tap from its section.

The **community widget** lives in a narrow sidebar on profitina.com and full-width in the app — a container-query problem.

Server-Sent Events keep one long connection: the $k \cdot RTT$ of Equation (10.4) is paid once, not at every poll.

Optimistic posting keeps interactions responsive in the INP sense on slow networks.

Book: Section 10.10.

Common Mistakes

No viewport line; fixed widths; breakpoints by device.

Placeholders as labels; the wrong keyboard.

Testing only on the median; hover-only interactions.

Quoting statistics without asking what was counted.

Book: Section 10.11.

Summary

Responsive, mobile-first design; read APJII, StatCounter and Ookla for what they count.

$w_{css} = w_{dev} / r$ (Equation (10.1)); without a viewport line sizes shrink by s (Equation (10.2), Equation (10.3)).

Thumbs, forms, Core Web Vitals; $t \approx k \cdot RTT + 8S/B$ (Equation (10.4)) and a budget.

Measured: 4 px text and 2.5 MB versus 16 px text, 48 px targets and 76 kB (Table 10.1).

Book: Section 10.12.

Exercises to Try

10.1 Effective size of 13 px text on 320 px and 412 px phones.

10.2 Largest page within 2.5 s on the Lighthouse profile and on the weak signal.

10.3 Answer the colleague who wants “desktop first, because 60% is desktop”.

10.5 Best `type`, `autocomplete` and `inputmode` for the four booking fields.

Full solutions are in the book. Book: Sections 10.13 and 10.14.

Next Time

Chapter 11 — Accessibility and Inclusive Design (WCAG 2.2). A puskesmas queue kiosk that shut out four patients. The law in Indonesia and the EU, WCAG 2.2 and 3.0, contrast computed, screen readers, and an audit in Java and Python.

Before then: run the check against both servers and try Exercises 10.1 and 10.2. Book: Chapter 11.

Lecture 11 — Accessibility and Inclusive Design (WCAG 2.2)

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 11

Lecturer: Irfan Subakti — yifana@gmail.com

Today we make a puskesmas queue kiosk usable by everyone who walks through its door. We see disability as a mismatch between person and design, learn what the law and WCAG 2.2 require, compute contrast by hand, see how screen readers read a page, and build a contrast checker and an audit in Java and Python. Book: Chapter 11, Sections 11.1–11.14.

What This Meeting Covers

1. **Four patients** — and the page that shut them out.
2. **Ideas** — the social model, inclusive design, the persona spectrum.
3. **Law and standards** — UU 8/2016, PP 42/2020, the European Accessibility Act, WCAG 2.2 = ISO/IEC 40500:2025, WCAG 3.0's status.
4. **Contrast and assistive technology** — three equations; roles, names and states; the first rule of ARIA.
5. **Testing** — WebAIM Million 2026, four layers of testing, and our own programs.

Book: Chapter 11.

A Morning at Puskesmas Sukamaju

Pak Harto, 71, eyes weakened with age: cannot see the pale grey hint; does not know which box to fill.

Mbak Laras, blind, uses TalkBack: hears “image, image, button, link” — nothing else.

Mas Dimas, broken right arm, uses the keyboard: Tab skips the three service boxes.

Bu Wulan, holding her baby: misses the tiny help icon with her thumb.

None of them is unusual; the failures are the web’s most common ones, and each has a simple fix. Book: Section 11.1.

Disability Is a Mismatch

The **social model:** a person with low vision is disabled *by* pale text; the same person is not disabled by large text.

Accessibility: people with disabilities can use a design effectively.

Inclusive design: design for the full range of human diversity from the start.

Persona spectrum: permanent (one arm), temporary (broken arm), situational (holding a baby).

Designs can be changed — which is why the social model empowers designers. Book: Section 11.2.

Who Is Affected

WHO: about **1.3 billion people — 16%, one in six** — experience significant disability, and the number grows as populations age.

Add temporary and situational limitations, and almost everyone is poorly served at some moment of the day.

Age changes eyesight, hearing, fine motor control and the speed of learning new interfaces.

That is why older patients notice an inaccessible kiosk first. Book: Section 11.3.1.

What the Law Requires

UU 8/2016: persons with disabilities have rights to accessibility and to information and communication, including in public services.

PP 42/2020: accessibility to housing, **public services** and disaster protection; providers must offer accessible services.

European Accessibility Act: applies since **June 28, 2025** to e-commerce, banking, e-books, communications and more; standard EN 301 549.

WCAG 2.2 (W3C Recommendation, updated December 12, 2024) = **ISO/IEC 40500:2025** (October 2025). **WCAG 3.0**: Working Draft of September 10, 2026, years of work ahead.

In September 2026, build and test against WCAG 2.2 level AA. Book: Section 11.3.2.

WCAG 2.2 on One Page

POUR: Perceivable, Operable, Understandable, Robust.

Levels **A** (minimum), **AA** (the usual legal target), **AAA** (highest).

86 success criteria: nine added to 2.1, and 4.1.1 Parsing removed.

Behind the common failures: 1.1.1, 1.3.1, 1.4.3, 1.4.11, 2.1.1, 2.4.4, 2.4.7, 3.1.1, 4.1.2.

Chapters 9 and 10 already used 2.5.8, 2.5.1, 2.5.7, 1.4.10 and 1.3.5. Book: Section 11.3.3.

The Nine New Criteria

Table 11.1 The nine success criteria added in WCAG 2.2.

Criterion	Level
2.4.11 Focus Not Obscured (Minimum)	AA
2.4.12 Focus Not Obscured (Enhanced)	AAA
2.4.13 Focus Appearance	AAA
2.5.7 Dragging Movements	AA
2.5.8 Target Size (Minimum)	AA
3.2.6 Consistent Help	A
3.3.7 Redundant Entry	A
3.3.8 Accessible Authentication (Minimum)	AA
3.3.9 Accessible Authentication (Enhanced)	AAA

Table 11.1: most of them help people with cognitive, motor or low-vision disabilities. Book: Section 11.3.3 and Table 11.1.

Contrast, Step 1: Linearize

$$c_{lin} = \begin{cases} c/12.92 & \text{if } c \leq 0.04045 \\ \left(\frac{c+0.055}{1.055}\right)^{2.4} & \text{otherwise} \end{cases} \tag{11.1}$$

Each channel 0–255 is divided by 255 to give c.

Equation (11.1) undoes the gamma curve with which screens encode brightness.

WCAG 2.2 uses **0.04045**; old tutorials show 0.03928 (no practical difference).

Book: Section 11.3.4.

Contrast, Steps 2 and 3

$$L = 0.2126\, R_{lin} + 0.7152\, G_{lin} + 0.0722\, B_{lin} \tag{11.2}$$

$$CR = \frac{L_1 + 0.05}{L_2 + 0.05} \tag{11.3}$$

Equation (11.2): relative luminance — green weighs most, blue least.

Equation (11.3): lighter over darker, each plus 0.05 for screen glare; from 1 to 21.

AA: **4.5:1** normal text, **3:1** large text (18 pt or 14 pt bold) and control parts (1.4.11); AAA: 7:1.

Book: Section 11.3.4.

Worked Example: the Pale Hint

#9CA3AF on white. Red 156: $c = 0.6118$, $R_{lin} = 0.3325$; $G_{lin} = 0.3663$; $B_{lin} = 0.4287$.

$L = 0.2126 \times 0.3325 + 0.7152 \times 0.3663 + 0.0722 \times 0.4287 = 0.3636$.

$C\,R = 1.05 / 0.4136 = 2.54$ — barely more than half of 4.5.

That is why Pak Harto could not see it. Book: Section 11.3.4, Example.

How a Screen Reader Reads a Page

TalkBack, VoiceOver, NVDA, JAWS read the **accessibility tree**: role, name, state.

`<button>`: role, Tab, Enter/Space. `<div onclick>`: no role, skipped by Tab (2.1.1).

`<img>` needs `alt` (or `alt=""` if decorative); image-only links and buttons have no name (2.4.4, 4.1.2).

`<label>` names a field (1.3.1); headings form the outline; `lang` chooses pronunciation (3.1.1).

First rule of ARIA: *if a native element does the job, use it. Book: Section 11.3.5.*

The Web in 2026

WebAIM Million (February 2026): failures on **95.9%** of home pages; **56.1** errors per page on average.

Six failures = **96%** of errors: low contrast 83.9%, missing alt 53.1%, missing labels 51.0%, empty links 46.3%, empty buttons 30.6%, missing language 13.5%.

Pages with ARIA averaged **more** errors (59.1 against 42).

Common failures are simple; automated tests see only part of the problem. Book: Section 11.3.6.

Four Layers of Testing

1. Automated checks — our programs, DevTools, axe, WAVE.

2. Keyboard walk-through — Tab, Shift+Tab, Enter, Space, arrows; focus visible and never hidden.

3. Screen reader and zoom — TalkBack/VoiceOver/NVDA; 200% and 400%.

4. People with disabilities — real tasks, Chapter 7's methods.

Layers 1 and 2 on every change; all four before a public service goes live. Book: Section 11.4.

Failures, Criteria and Our Checks

Table 11.2 The most common failures, the criteria behind them, and this chapter's checks.

Failure (WebAIM 2026)	WCAG 2.2	Our check
Low-contrast text	1.4.3, 1.4.11	`contrast`
Image without alt	1.1.1	`ally_audit`
Input without label	1.3.1, 4.1.2	`ally_audit`
Empty link / button	2.4.4, 4.1.2	`ally_audit`
Missing page language	3.1.1	`ally_audit`
Clickable non-control, heading jump, duplicate id	2.1.1, 1.3.1, 4.1.2	`ally_audit`

Table 11.2. Book: Section 11.4 and Table 11.2.

The Contrast Checker

Listing 11.1 From `contrast.py`: Equation (11.1), Equation (11.2) and Equation (11.3) in three small functions.

```python
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)
```

*Listing 11.1. For a failing pair the tool also **suggests** a darker color that passes, keeping the hue. `Contrast.java` is the same. Book: Section 11.5.2.*

The Kiosk's Colors

```
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
```

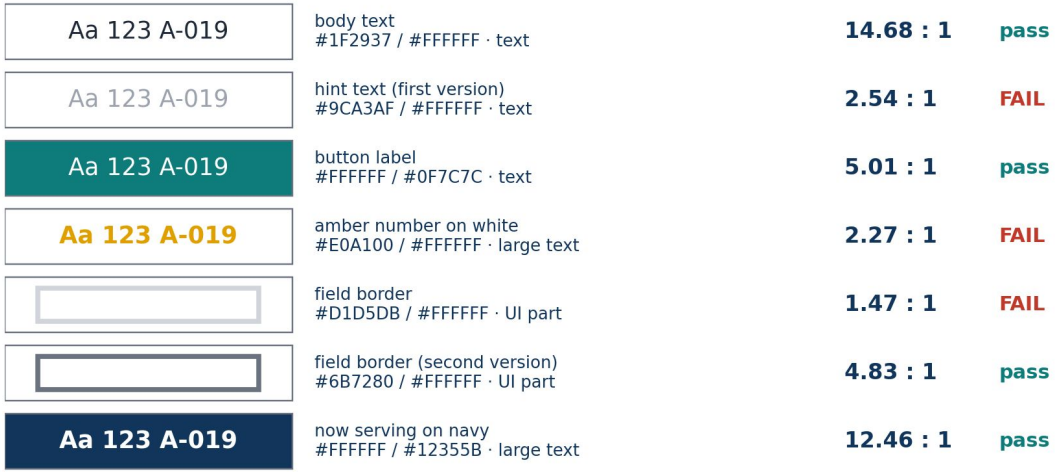


Figure 11.1 The kiosk's seven color pairs with their ratios; three fail AA.

Figure 11.1: pale hint 2.54, amber number 2.27, light border 1.47. The repaired page uses #4B5563, dark text for the number, and a #6B7280 border. Book: Section 11.6.

The Audit

Two regular expressions — `TAG` and `PAIR` — so Java (no HTML parser in the JDK) can be identical.

Labeled = `<label for>`, wrapped in `<label>`, `aria-label` or `aria-labelledby`.

Named = text, an image with non-empty `alt`, or `aria-label`.

`onclick` on anything but a link or button = not reachable by keyboard.

```
../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)
```

Twelve problems in the first page, none in the repaired one; each line maps onto one of the four patients. Book: Sections 11.5.3 and 11.6.

The Repaired Page

`lang`, decorative logo with `alt=""`, headings h1 → h2.

Real `<button>`s 64 px high; a labeled card field with a hint via `aria-describedby`, `inputmode="numeric"`, a 3:1 border.

`aria-live="polite"` announces “Now serving: A-020”.

A 4 px amber `:focus-visible` ring shows where the keyboard is.

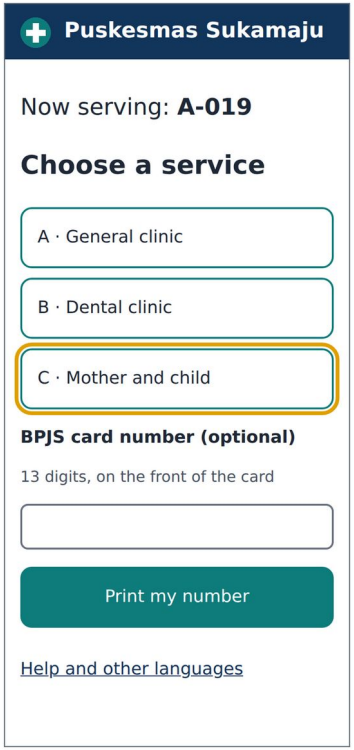


Figure 11.2 The repaired kiosk after three presses of Tab, with the focus ring on the third service.

Figure 11.2. Book: Sections 11.5.4 and 11.6.

Writing Good Alternative Text

Describe the **purpose**, not the pixels: a print icon in a button is “Print my number”, not “printer icon”.

Decorative images get `alt=""` so screen readers skip them; the logo next to the visible name is decorative.

Never a file name (“print.png”) or “image of…”; the screen reader already says “image”.

Charts need the message in text, or a table: “Queue waiting time fell from 40 to 25 minutes in August”.

Exercise 11.4 turns the file-name rule into code. Book: Sections 11.3.5 and 11.11.

A Ten-Minute Keyboard Test

Put the mouse away. Press Tab from the top: does every control get focus, in reading order?

Is the focus always visible, and never hidden under a sticky header or a pop-up (2.4.7, 2.4.11)?

Can every action be done with Enter, Space and the arrows — no trap you cannot Tab out of?

Zoom to 200% and 400%: does the page reflow without sideways scrolling (1.4.4, 1.4.10)?

Ten minutes of this finds problems that no automated tool reports. Book: Sections 11.4 and 11.8.

Tests

`pytest` (4 tests): black on white 21:1, a color on itself 1:1, #767676 on white **4.54**; the suggested fix passes; 12 problems vs 0; a wrapped label counts.

`Ch11Test.java`: the same six checks, “all 6 checks passed”.

Book: Section 11.7.

Beyond the Automated Layer

Keyboard: six stops — three services, the card field, Print, Help — in reading order; the ring always visible; nothing hidden (2.4.11).

Improve: after choosing a service, announce “Your number is B-024”.

Screen reader: “Choose a service, heading level 2; A, General clinic, button; …”.

People: do patients say “Mother and child” or “KIA”? Keep a staff member who takes numbers for those who ask.

Book: Section 11.8.

Using It for Real

Put “WCAG 2.2 level AA” into every public-service order (UU 8/2016, PP 42/2020).

Run the audit and the contrast check on every release, like unit tests.

Fix with native HTML first; test the design-system palette once, early.

Publish an accessibility statement with another way to get help.

Book: Section 11.9.

HCI in Practice: Profitina

A readiness **score** must never be color alone (1.4.1): number, level word and explanation as text.

The weekly **trend chart** needs a text or table alternative and CSV export.

UU PDP self-service (consent, export, purge, HTML legal pages) is also an accessibility matter: consent a screen-reader user cannot perceive is not meaningful.

Link Safety warnings must be announced as text, not only as a colored icon.

Book: Section 11.10.

Common Mistakes

Clickable `div`s; images without (or with bad) `alt`; placeholders as labels.

Pale text and borders; `outline: none`; color as the only signal.

ARIA as decoration; trusting the tool; accessibility as a final step.

Book: Section 11.11.

Summary

Disability is a mismatch; the persona spectrum widens who benefits. One in six people (WHO).

UU 8/2016, PP 42/2020, the EAA (June 28, 2025), ISO/IEC 40500:2025; WCAG 3.0 still a draft.

POUR, A/AA/AAA, 86 criteria, nine new (Table 11.1); contrast by Equation (11.1), Equation (11.2), Equation (11.3).

Four layers of testing (Table 11.2); 12 problems found and fixed (Figure 11.2).

Book: Section 11.12.

Exercises to Try

11.1 Contrast of #767676 on white by hand.

11.2 Amber on navy, white on amber — which for “now serving”?

11.4 Report an `alt` that is only a file name.

11.6 A CAPTCHA login and WCAG 3.3.8 — why passkeys help (90% aware, 75% enabled on at least one account, FIDO 2026).

Full solutions are in the book. Book: Sections 11.13 and 11.14.

Next Time

Chapter 12 — Emotional Design, Trust and Ethical UX. A laundry's membership that copied the tricks of big apps. Norman's three levels, trust, dark patterns, the DSA, the FTC, UU PDP and PP 33/2026, and a consent linter and ledger.

Before then: walk through `queue_v2.html` *with Tab and a screen reader, and try Exercises 11.1 and 11.3. Book: Chapter 12.*

Lecture 12 — Emotional Design, Trust and Ethical UX

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 12

Lecturer: Irfan Subakti — yifana@gmail.com

Today we design for feelings honestly. A laundry copied the persuasion tricks of large apps and spent years of trust in weeks. We name the tricks, see what the EU, the FTC and Indonesia's UU PDP and PP 33/2026 say about them, and build a consent linter and a consent ledger in Java and Python. Book: Chapter 12, Sections 12.1–12.14.

What This Meeting Covers

1. Bu Sri's membership — how "growth" tricks turned a trusted laundry into "a loan-shark app".

2. Emotional design — Norman's three levels and the aesthetic–usability effect.

3. Trust — competence, benevolence, honesty; passkeys.

4. Dark patterns and the law — a catalog, two measures, the DSA, the FTC, UU PDP and PP 33/2026.

5. Tools — the fair-flow review, a linter for described flows, and a ledger of consent events.

Book: Chapter 12.

Bu Sri’s Membership

Laundry Wangi, a *kiloan* laundry; for years orders by WhatsApp, honest replies, a wallet returned with its money counted.

The new app’s *Member Hemat*: Rp25,000 a month — with a red “50% off!” banner, a countdown that **restarts at 09:59**, “Only 3 places left today!” every day.

One **pre-ticked** box for terms, privacy, promotions and sharing with partner shops; a big “I agree to everything”; a tiny “No thanks, I like paying more”.

Join in 2 taps; leave in 6 steps via WhatsApp.

A month of sign-ups, then a number in a repair shop’s broadcast, complaints in the group chat, and “Kok sekarang Ibu kayak aplikasi pinjol?” Book: Section 12.1.

The Idea in Plain Words

Emotional design: deliberately designing how an interface makes people feel — at best, *delight* ("being ironed — ready at 16:00").

Trust: willingness to rely on a system you cannot fully check; slow to build, quick to lose.

Dark pattern (deceptive pattern, Brignull 2010): a design that makes users do what they did not mean to, for the business's benefit.

Consent: freely given, specific, informed, unambiguous — and withdrawable.

Book: Section 12.2.

Norman's Three Levels

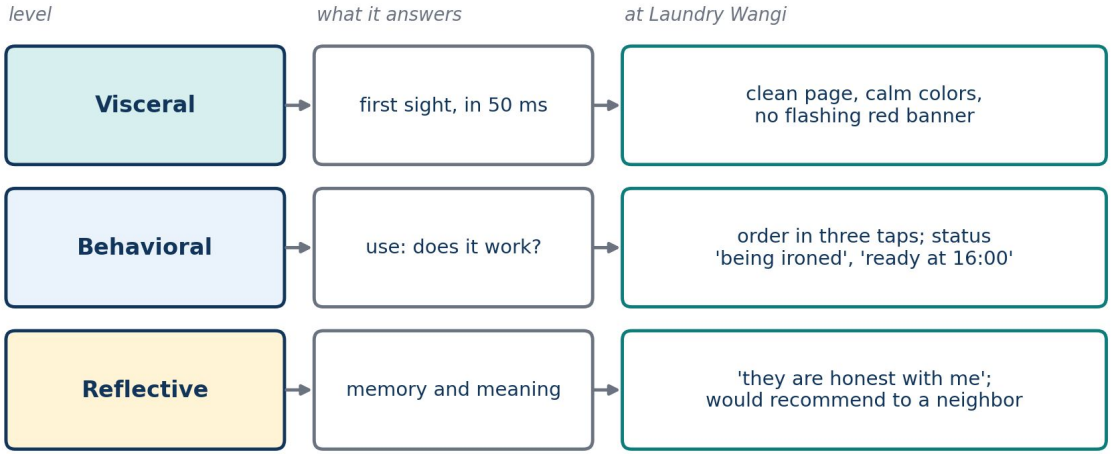


Figure 12.1 Norman's three levels of emotional design applied to Laundry Wangi.

*Figure 12.1: **visceral** (first impression within about 50 ms, Lindgaard et al.), **behavioral** (does it work?), **reflective** (do I trust them, would I recommend?). Book: Section 12.3.1.*

The Aesthetic–Usability Effect

Kurosu and Kashimura (1995): attractive interfaces are **rated** easier to use, even when they are not.

The visceral level colors the behavioral judgment.

A reason to care about appearance — and a warning: beauty can hide usability problems from opinion-only tests.

Dark patterns work on the visceral and behavioral levels and **damage the reflective level**, where trust lives.

Bu Sri's customer spoke from the reflective level. Book: Section 12.3.1.

How Trust Is Built and Lost

Competence (it works), **benevolence** (it acts for me), **honesty** (it tells the truth, keeps promises).

Built by: accurate status, clear prices, visible undo and exit, admitted errors, asking only for needed data.

Lost by: a discovered lie, a hidden cost, data used unexpectedly, an exit much harder than the entrance.

FIDO 2026: 47% likely to abandon a purchase or sign-in over a forgotten password; 90% aware of **passkeys**, 75% enabled one.

Passkeys improve the behavioral and the reflective level at once. Book: Section 12.3.2.

What Research Found

Mathur et al. (2019): crawled about 11,000 shopping sites; **1,818** dark-pattern instances on **1,254** sites (about 11%) — fake timers, fake low stock, confirmshaming.

Gray et al. (CHI 2024): an ontology uniting the taxonomies — *sneaking, obstruction, interface interference, forced action, social engineering*.

Ahuja, Gunawan, Bielova and Santos (CHI 2026): 59 patterns mapped to the DSA's autonomy violations; eight design factors.

HCI research and regulation now work on the same questions. Book: Sections 12.3.3 and 12.3.4.

The Catalog at Laundry Wangi

Table 12.1 The dark patterns of the first Laundry Wangi flow and their fair alternatives.

Pattern	What was done	Fair alternative
False urgency	Countdown restarts at 09:59	A real deadline, or none
False scarcity	“Only 3 places left” daily	Real limits only
Pre-ticked box	Optional purposes on	Optional purposes off
Bundled consent	One tick for four things	One choice per purpose
Interface interference	Big “agree”, tiny grey “no”	Equal buttons
Confirmshaming	“I like paying more”	Neutral: “Not now”
Hard to cancel	Join 2 steps, leave 6	Leave where you joined

Table 12.1. Book: Section 12.3.3 and Table 12.1.

Two Measures

$$A = \frac{s_{out}}{s_{in}} \tag{12.1}$$

$$P = \frac{w_a h_a}{w_r h_r} \tag{12.2}$$

Equation (12.1) **friction asymmetry:** steps to leave over steps to join; fair means $A \le 1$.

Equation (12.2) **prominence ratio:** area of accept over area of reject; this book's rule of thumb is $P \le 2$ and the same style (not a legal threshold).

Laundry Wangi: $A = 6/2 = 3$; $P = (320 \times 56)/(90 \times 14) = 17920/1260 = 14.2$.

Book: Section 12.3.3.

The Law: EU and USA

DSA Article 25 (applies to all online platforms since February 17, 2024): no interfaces that deceive, manipulate or materially distort free and informed decisions.

Digital Fairness Act: announced; expected to be presented in Q4 2026 — dark patterns, addictive design, unfair personalization.

FTC: *Bringing Dark Patterns to Light* (2022); the click-to-cancel rule was **vacated** by the Eighth Circuit in July 2025; a new advance notice of rulemaking went to OMB on January 30, 2026.

The principle of Equation (12.1) remains policy in intent in the USA. Book: Section 12.3.4.

The Law: Indonesia

UU PDP 27/2022: explicit valid consent as a lawful basis (Art. 20); written or recorded, distinguishable, clear language (Art. 22); right to withdraw (Art. 9); evidence of consent (Art. 24); parental consent for children (Art. 25).

PP 33/2026: implementing regulation, enacted **July 16, 2026**, in force **January 16, 2027**; consent explicit and valid, for specific communicated purposes.

The pre-ticked, bundled box and WhatsApp-only withdrawal fit none of these.

Designing the consent screen is how the law's requirements reach the user. Book: Section 12.3.4.

Measuring Feelings

Questionnaires: **UEQ** (attractiveness, perspicuity, efficiency, dependability, stimulation, novelty) and its 8-item **UEQ-S**; SUS for perceived usability (Chapter 7).

For trust, behavior speaks louder: cancellations, consent withdrawals, complaints, recommendations.

A design that raises sign-ups **and** withdrawals and complaints is not a success.

Book: Section 12.3.5.

Persuasion That Respects the User

Transparency: would it still work if the user fully understood it? A true saving does; a fake countdown does not.

Reversibility: can the user easily undo what the design encourages?

Alignment: does it serve the user's goal as well as the business's?

Pass all three, and be as persuasive as you like.

Book: Section 12.3.6.

The Fair-Flow Review

Every claim true? Optional purposes off? One choice per purpose, plain words?
Equal choices look equal ($P \le 2$)? Neutral refusal? Leaving as easy as joining ($A \le 1$)?
Withdrawal where consent was given? Consent recorded as evidence (who, purpose, when, text version)?
Total cost clear before paying — renewal date, price, how to stop, reminder?

Table 12.2 A shortened fair-flow review.

Question	Check
Claims true?	Countdowns, stock, prices
Optional off?	Promotions, sharing, profiling
Equal choices equal?	$P \le 2$ by Equation (12.2)
Exit as easy as entry?	$A \le 1$ by Equation (12.1)
Consent as evidence?	Events with text versions

Table 12.2: every "no" is a defect to fix before release; the book's table has nine rows. Book: Section 12.4 and Table 12.2.

Describing a Flow

Listing 12.1 `flow_v1.txt`: the first flow, one line per purpose, button, countdown, steps and withdrawal.

```
# 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
```

Listing 12.1: a program can only check what is described. Book: Section 12.5.1.

The Linter's Two Measures

Listing 12.2 From `consent_lint.py`: Equation (12.2) and Equation (12.1) as rules.

```
    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"))
    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"))
```

Listing 12.2. The other rules: optional-and-on, `bundled|yes`, shaming phrases (English and Indonesian), a resetting countdown, withdrawal "only" elsewhere. `ConsentLint.java` is the same. Book: Section 12.5.2.

Running the Linter

```
../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)
```

Eight findings in the first flow — one for each thing Bu Sri's customers met — and none in the second, whose purposes are off, buttons equal, countdown gone, and cancellation two steps. Book: Sections 12.5.3 and 12.6.

The Two Consent Screens

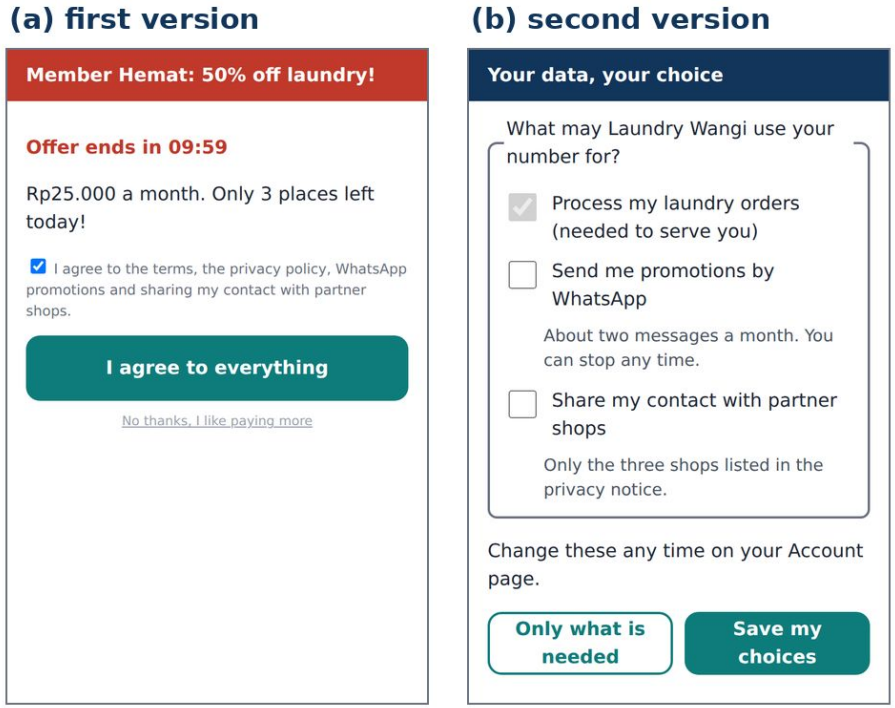


Figure 12.2 The first and the second consent screen on a phone.

Figure 12.2: (a) red banner, restarting countdown, one pre-ticked box, a big agree button and a shaming link; (b) one choice per purpose, plain explanations, two equal buttons. Book: Section 12.6.

A Ledger, Not a Flag

Consent is a **history**: give, withdraw, give again to a newer text.

Keep every event: time, user, purpose, action, text version; derive the current state.

One question for the rest of the system: *may we use this data for this purpose now?*

```
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
```

U17's later withdrawal wins; U23's consent is to text v3 — a v4 needs a new question (Exercise 12.5). This is the evidence UU PDP Art. 24 asks for. Book: Sections 12.5.4 and 12.6.

Why Small Businesses Risk More

Customers of a *kiloan* laundry live in the same neighborhood and share one group chat.

One bad experience reaches dozens of people who never used the app — within minutes.

One good experience travels the same way: a mixed-up shirt returned with an apology; an honest “the rain will delay drying”.

For Bu Sri, ethical design is not a moral luxury but the most sensible business strategy.

Book: Sections 12.1 and 12.3.2.

Microcopy That Calms

Compare a red “Payment failed!” with “Payment has not arrived. Your balance was not charged; try again or pay at the counter.”

The second answers three questions at once: **what happened, is my money safe, what can I do now?**

The same rule for delays, refusals and errors: say what is true, say what it means for the user, offer the next step.

For customers who have met online fraud, such sentences separate a trustworthy service from a suspicious one.

Book: Section 12.8.

Tests

`pytest` (3 tests): eight findings with 14.2 and 3.0 checked by hand; the second flow clean; a later withdrawal wins and a never-asked purpose is not allowed.

`Ch12Test.java`: the same six checks, “all 6 checks passed”.

Book: Section 12.7.

Making It Better

True delight: status per order (“drying — the rain may add half a day”), and “You saved Rp18,000 in September”.

Reminders before charging: renewal date on the Account page, a reminder a day before, stop in one step.

Beyond the linter: vague purposes (“to improve our services”) pass a description check; read them aloud to an outsider.

Book: Section 12.8.

Using It for Real

Run the fair-flow review on every flow that asks for money, data or attention; lint every release.

Keep a consent ledger with text versions; show it on the Account page with a withdraw control per purpose.

Write purposes in the user’s language with concrete consequences.

Count the costs of tricks next to sign-ups.

If you would not say it at your counter, your app should not say it either. Book: Section 12.9.

HCI in Practice: Profitina

UU PDP self-service: explicit consent, Argon2id, login rate limit, data export, email/password change, full account purge, HTML legal pages — the exit as available as the entrance.

Pro subscription (Rp50,000, 30 days): auto-renew from balance with a reminder within 24 hours before.

Channels: a 50 PROFIT balance gate that is checked, **not deducted** — and says so; stated limits.

Content engine: serves an AI draft only when better, and says so when no model is available. *The app advises; the user decides.*

Book: Section 12.10.

Common Mistakes

Copying growth tricks; fake urgency and scarcity.

Pre-ticked and bundled consent; unequal buttons; guilt-trip wording.

Exits harder than entrances; one overwritten consent flag; vague purposes.

Mistaking a sign-up spike for success.

Book: Section 12.11.

Summary

Visceral, behavioral, reflective (Figure 12.1); trust = competence, benevolence, honesty.

Dark patterns (Table 12.1) on about 11% of 11,000 shopping sites; A (Equation (12.1)) and P (Equation (12.2)) make two measurable.

DSA Art. 25; DFA expected late 2026; FTC rule vacated, rulemaking restarted; UU PDP and PP 33/2026 (from January 16, 2027).

The review (Table 12.2), the linter and the ledger turn principles into checks (Figure 12.2).

Book: Section 12.12.

Exercises to Try

12.1 Compute A for a one-tap subscription with a seven-step cancellation, and design $A \leq 1$.

12.2 P for a 300 × 48 “Accept all” against a 70 × 16 “Settings” link.

12.5 Why U23’s v3 consent does not cover v4, and a `may_use` that checks versions.

12.6 When is a scarcity message information, and when is it a dark pattern?

Full solutions are in the book. Book: Sections 12.13 and 12.14.

Next Time

Chapter 13 — Designing AI Interfaces: Chat, Copilots and Agents.

Uncertainty, explanation, human-in-the-loop and trust when the system itself is a model.

Before then: lint a flow from an app you use, and try Exercises 12.1 and 12.5. Book: Chapter 13.

Lecture 13 — Designing AI Interfaces: Chat, Copilots and Agents

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 13

Lecturer: Irfan Subakti — yifana@gmail.com

Today we design the conversation between people and a system that guesses: what it must show, when it must ask, what it may do alone, and how we measure whether people rely on it wisely. Book: Chapter 13, Sections 13.1–13.12.

What This Meeting Covers

1. **The problem** — a laundry's midnight WhatsApp orders, and three questions from its owner.
2. **Chat, copilot, agent** — three kinds of AI interface and the line of consequence between them.
3. **The theory** — guidelines for human-AI interaction, levels of automation, trust, reliance and calibration, agents in 2025–2026, the rules of 2026.
4. **The method** — the AI interaction canvas and a release checklist.
5. **The programs** — an order assistant and a reliance analysis in Java and Python, with real output.

Every program today exists in Java and Python, and every output shown is the output the program really printed. Book: Chapter 13.

Bu Wati's Midnight Messages

Profitina Laundry receives twenty or thirty WhatsApp orders between ten at night and two in the morning.

The messages are short and incomplete: "I have a bag of laundry, pick up at 16 please" gives neither weight nor service nor day.

By the time Bu Wati answers in the morning, some customers have gone to a shop that answered at midnight.

Her nephew proposes an AI assistant. Her three questions: *How will customers know it is a machine? What happens when it misunderstands? Who is to blame when it books the wrong thing?*

These are questions about the interface around the model, not about the model. Two assistants on the same model can be trustworthy or dangerous depending on these design decisions. Book: Section 13.1.

Chat, Copilot and Agent

> **DEFINITION.** A *chat interface* answers in words and the person decides. A *copilot* proposes changes inside another tool that the person accepts or rejects. An *agent* takes actions in the world on the person's behalf: booking, paying, sending, deleting.

A wrong chat reply wastes a minute; a wrong accepted suggestion becomes the person's mistake; a wrong agent action *has already happened*.

The more an AI can do, the more its interface must show what it is about to do and let people stop it.

In July 2025 ChatGPT agent began browsing, filling forms and purchasing, and is trained to ask permission before actions with real-world consequences. Bu Wati's assistant is a small agent: it chats, proposes and books. Book: Sections 13.1 and 13.2.1.

Four Questions Every AI Interface Must Answer

Classic interfaces are deterministic: the same button does the same thing. An AI system guesses, and a guess can be wrong in a new way each time. Table 13.1 lists what people need to know.

Table 13.1 The four questions a person has about any AI system.

Question the person has	What the interface must show
What can it do, and how well?	Its scope and limits, before the first use
How sure is it about this answer?	Uncertainty in a form people understand
What will it do next?	The proposed action, before it happens
How do I correct, stop or undo it?	A way back, and a way to a person

A quick test for any AI product: if you cannot answer one of the four from the screen alone, the design has a gap. Book: Section 13.2.2 and Table 13.1.

A Worked Example by Hand

"I have a bag of laundry, pick up at 16 please": service — not said, *assume* wash and fold and say so; weight — not said, *ask*; speed — assume regular; pickup — "at 16", tomorrow is only *likely*.

The customer answers "about 2.5 kg". The minimum charge is 3 kg, so the assistant must say *why* it charges for 3.

$$P = r_s \cdot \max(w, w_{\min}) \cdot e \tag{13.1}$$

By Equation (13.1): $P = 8000 \cdot \max(2.5, 3) \cdot 1 = 24000$ rupiah. Only after "yes" does it book.

r_s is the rate per kilogram of service s, w the weight, $w_{\min} = 3$ the minimum, and $e = 1.5$ for express, otherwise 1. Guessing the weight is not acceptable because the price depends on it. Book: Section 13.2.3.

Guidelines for Human-AI Interaction

Amershi et al. (CHI 2019) reduced 150+ recommendations to **18 guidelines**, tested with 49 designers on 20 AI products; maintained today as the Microsoft HAX Toolkit.

Initially: G1 make clear what it can do; G2 make clear how well.

When wrong: G8 efficient dismissal; G9 efficient correction; G10 scope services when in doubt; G11 make clear why it did what it did.

Over time: G13 learn from behavior; G17 global controls (reach a person at any time).

Google's People + AI Guidebook covers the same ground from a product team's view; its advice on trust is that explanations should help people calibrate reliance, not increase it. Both agree: design around what happens when the AI is wrong. Book: Section 13.3.1 and Table 13.2.

The Autonomy Ladder

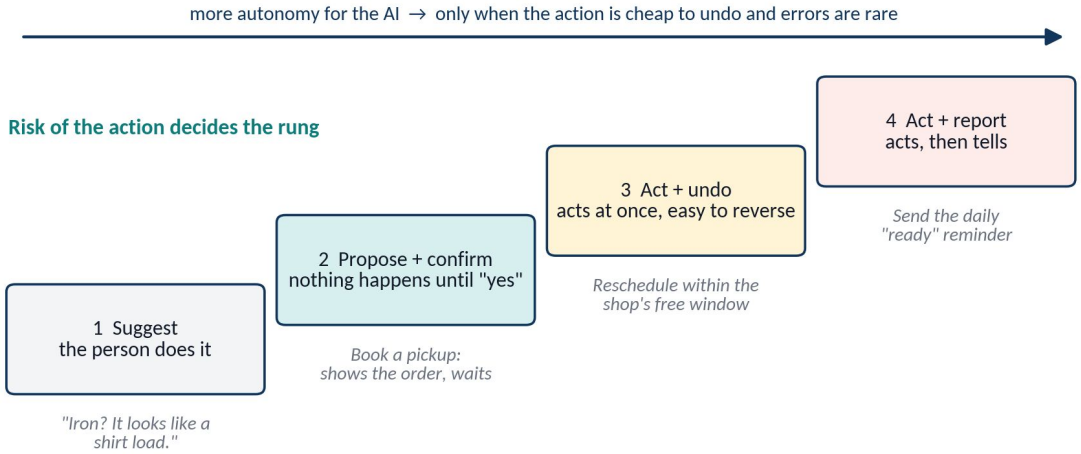


Figure 13.1 The autonomy ladder: risk and reversibility of each action decide how much the AI may do without asking.

Figure 13.1 turns Parasuraman, Sheridan and Wickens's levels of automation and Horvitz's mixed-initiative principles into four rungs: suggest, propose and confirm, act with undo, act and report. Book: Section 13.3.2 and Figure 13.1.

Autonomy Is Chosen per Action

> **KEY INSIGHT:** Autonomy is not a property of the AI; it is a property of each action. A payment, a deletion or a message in the customer's name sits low on the ladder however good the model is.

Suggest the service (rung 1); propose a pickup and wait for "yes" (rung 2); send the daily "ready" reminder alone (rung 4).

Horvitz (1999): act alone only when the expected benefit exceeds the expected cost of being wrong; ask when unsure, because asking is cheaper than repairing.

A booking sends a person on a motorbike across town. Even at high confidence it stays on rung two, where one "yes" costs the customer a second. Book: Sections 13.3.2 and 13.8.

Trust and Reliance

Lee and See: the goal is **appropriate reliance** — rely when the AI is right, not when it is wrong.

$$\text{over-reliance} = \frac{\text{accepted and AI wrong}}{\text{AI wrong}} \tag{13.2}$$

$$\text{under-reliance} = \frac{\text{rejected and AI right}}{\text{AI right}} \tag{13.3}$$

$$\text{team right} = \frac{\text{accepted and AI right} + \text{rejected and AI wrong}}{\text{all cases}} \tag{13.4}$$

Equation (13.2) counts automation bias, Equation (13.3) counts wasted good advice, and Equation (13.4) is what the shop cares about: the share of final decisions that are right. Book: Section 13.3.3.

What CHI 2026 Found

Raees, Khan, Lykourentzou and Papangelis reviewed human-AI decision-making studies: people **over-rely on AI advice without analytically engaging with it**.

Interventions that make people engage (decide first, show why) help reliance but cost time and effort.

Every design that reduces over-reliance adds friction; the designer decides where the friction belongs.

There is no free lunch in reliance design. Measure both kinds of error, not just one. Book: Section 13.3.3.

Calibration: Is the Confidence Honest?

A system is *calibrated* when, among the cases where it says 80%, it is right about 80% of the time. Modern neural networks are often overconfident (Guo et al., 2017).

$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{n_b}{N} \left| \mathrm{acc}(b) - \mathrm{conf}(b) \right| \tag{13.5}$$

Equation (13.5) averages the gap between stated confidence and actual accuracy over B bins, weighted by bin size.

ECE 0 means honest confidence; 0.10 means off by ten points on average. When ECE is high, a percentage misleads; words with measured thresholds serve people better. Book: Section 13.3.3.

Agents and Tools in 2025–2026

Agents that operate software: ChatGPT agent narrates on screen, asks before purchases, requires oversight for sending email, and lets the person interrupt, take over or stop.

A common tool protocol: the Model Context Protocol connects AI applications to tools; in December 2025 it joined the Agentic AI Foundation under the Linux Foundation.

Design consequence: an agent must show *which* tool it will use with *what* data, and let the person approve.

These are rungs two and three of the ladder plus guideline G17's global controls. Book: Section 13.3.4.

The Rules in 2026

EU AI Act Article 50 (from August 2, 2026): tell people they are interacting with AI; mark AI-generated content. The 2026 omnibus delayed high-risk duties to 2027–2028 but kept Article 50.

Indonesia: UU PDP 27/2022 and PP 33/2026 — consent free, specific, unambiguous, unbundled, withdrawable; no deceptive consent requests.

Two presidential regulations on an AI roadmap (2026–2029) and AI ethics were awaiting signature as of September 2026.

The practical rule holds everywhere: say it is a machine, say what data it uses, let the person reach a human. Book: Section 13.3.5.

The AI Interaction Canvas

Table 13.2 Selected rows of the AI interaction canvas for the order assistant.

Row	Order assistant
1. Job	Takes pickup orders; not complaints, not refunds
2. Disclosure	First line of every chat
4. Uncertainty	Words: sure ≥ 0.9, likely ≥ 0.6, else assumed
5. Ask or assume	Asks weight and time; may assume service and speed
6. Actions and rungs	Quote: rung 1; booking: rung 2; reminder: rung 4
7. Undo and stop	“no” before booking; UNDO for one message after
8. Human handoff	STAFF; the whole chat is passed on
10. Failure	Say so and pass to a person

Table 13.2 shows eight of the ten rows; the book adds inputs and data (row 3) and measures (row 9). Fill it with the people who own the risk: a programmer alone leaves rows 7, 8 and 10 empty. Book: Section 13.4 and Table 13.3.

The Release Checklist

A person who did not build the assistant answers “yes” to each, from the screen alone:

Do I know it is a machine? Can I see what it understood? Do I know which parts it guessed?

Can I say no? Can I undo? Can I reach a person?

Does it ask before it spends my money or acts in my name?

Seven questions, one per failure that users actually meet. Book: Section 13.4.

The Order Assistant: How It Works

Keyword rules, not a language model: the subject is the interface around the guess. Replacing the rules by a model changes only `parse`.

`parse` turns a message into four **slots** with confidences: service, weight, speed, pickup.

`label` turns confidence into words; `Assistant` keeps the remembered slots, a pending proposal, and the last booking with a one-message undo window.

Java uses a `record Slot` and `LinkedHashMap`s; integer arithmetic in tenths of a kilogram keeps prices identical in both languages.

Listings 13.1 and 13.2 in the book give the complete programs `laundry_assistant.py` and `LaundryAssistant.java`. Book: Sections 13.5.1 and 13.5.2.

Four Design Decisions in the Code

Disclosure first: “[Automated assistant. Type STAFF at any time to reach a person.]”

Ask, do not guess (G10): missing weight or pickup time produces a question, with a way out for those who do not know.

Label assumptions (G11): “assumed – please check” appears exactly where a default was used.

Remember the conversation (G13): a slot is replaced only when a new message says it more clearly.

Each decision maps to one row of Table 13.2 in the book. Book: Section 13.5.1.

Running It: Conversation 1

```
=== 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.
```

Everything except the speed was given; the assistant proposes, labels the assumption, books on “yes” and cancels on “undo”. Book: Section 13.5.3.

Running It: Conversation 2

```
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)
```

The assistant asks for the missing weight, remembers the hour from the first message, marks the day as only likely, and explains the minimum charge. A second “undo” later in this chat arrives too late and is refused politely. Book: Section 13.5.3.

Measuring Reliance: the Study

Next feature: a suggestion to staff whether a garment is machine-safe, from a photo of its care label.

Version A shows the suggestion only; **version B** adds confidence in words and a one-line reason.

Before a real study, a *simulated* log of 200 cases per version tests the analysis. Built-in truths: the model is overconfident by 8 points; staff in B accept less when confidence is low or the reason looks wrong.

`make_log.py` *writes the seeded log;* `reliance.py` *and* `Reliance.java` *compute Equation (13.2), Equation (13.3), Equation (13.4) and Equation (13.5). Book: Section 13.6.*

The Result

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

B cut over-reliance from 91.2% to 48.4% but raised under-reliance from 6.1% to 26.1%; right final decisions moved only from 65.0% to 67.0%. Showing confidence changed how people were wrong more than how often. Book: Section 13.6.

Why: an Overconfident Model

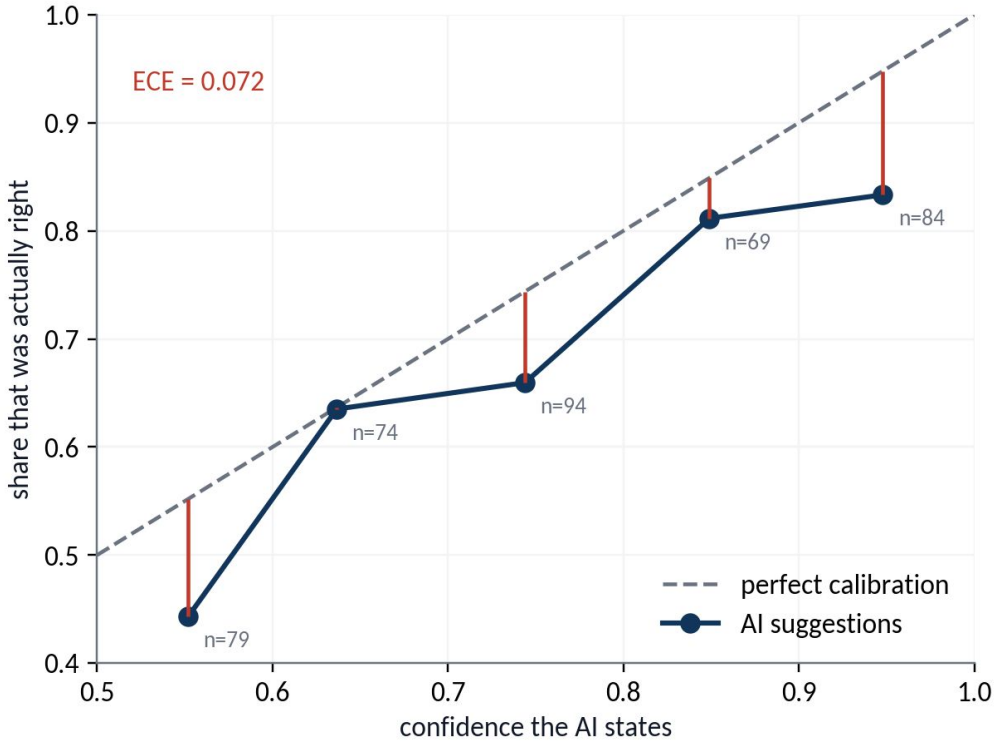


Figure 13.2 Calibration of the simulated care-label model: points below the diagonal mean the model claims more than it earns.

Figure 13.2: in the top bin the model claims 0.948 but is right 0.833 of the time; ECE is 0.072. Lesson: first calibrate, then show confidence, then measure reliance again. Book: Section 13.6 and Figure 13.2.

Testing the Promises

```
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
```

“Undo expires after one message” is not an implementation detail; it is a promise the interface makes. A test that pins the promise stops a later change from breaking it silently. Book: Section 13.7.

Making It Better

Explain the guess, briefly: “tomorrow, because you wrote at night and gave no day”.

Set thresholds from data: “sure” must be right at least nine times in ten.

Ask in order of cost: the slot whose error costs most comes first.

Leave a visible record: a summary the customer can forward or screenshot.

Tempting and wrong: booking without confirmation when every slot is “sure”.

Every improvement follows from a guideline or a measurement. Book: Section 13.8.

Using It for Real

Pak Joko’s catering: menu, boxes, time, address — and **dietary needs**, which may *never* be assumed. Any mention of an allergy goes to a person.

Log proposals, confidences and outcomes, with separate, plain consent under PP 33/2026.

Decide in advance what happens when the model or network is down; read real chats every week.

The rule: for any slot where a wrong default could harm someone, there is no default. Book: Section 13.9.

HCI in Practice: Profitina

Honest failure: when no local model is available, the app says so instead of silently serving a template.

A quality gate: an AI draft is served only when it beats a built-in template; after the September 2026 revision AI drafts win in 5 of 6 benchmark cases.

The app advises, the user decides: signals behind every score; nothing published without an explicit action.

Profitina's AI (Qwen2.5) runs locally, so user data never leaves the server for AI processing; its feedback pipeline makes "learn over time" visible. Book: Section 13.10.

Common Mistakes

Hiding that it is a machine; guessing what must be asked; uncalibrated percentages.

Measuring only over-reliance — report Equation (13.2), Equation (13.3) and Equation (13.4) together.

One autonomy level for the whole product; no way out; forgetting the conversation; an undo that is not really one.

Each mistake in the book comes with its fix. Book: Section 13.11.

Summary: Five Things to Remember

1. Chat answers, copilots propose, agents act; the more an AI can do, the more its interface must show and let people stop.

2. Answer the four questions of Table 13.1 from the screen alone.

3. Choose autonomy per action from risk and reversibility (Figure 13.1).

4. Measure over-reliance, under-reliance and right final decisions; calibrate before showing confidence (Equation (13.5)).

5. Disclose, minimize data, and always offer a person — the law of 2026 and good design agree.

Book: Section 13.12.

Next Time

Chapter 14 — Data-Informed UX. Events, funnels, HEART metrics and A/B experiments that can be trusted.

The reliance log of today is the kind of evidence Chapter 14 teaches us to collect at scale.

Before then: work Exercises 13.1, 13.3, 13.4 and 13.6.

Book: Sections 13.13 and 13.14.

Lecture 14 — Data-Informed UX: Analytics, Experiments and Metrics

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 14

Lecturer: Irfan Subakti — yifana@gmail.com

Today we replace opinions with what people actually do: events, funnels, metrics derived from goals, and A/B experiments that we check before we believe them, all collected with consent. Book: Chapter 14, Sections 14.1–14.12.

What This Meeting Covers

1. **The problem** — a warung's ordering site with visitors but few orders, and three competing opinions.
2. **Events, funnels, metrics** — step and overall conversion, and where improvement pays.
3. **The theory** — HEART, Indonesia's 2026 numbers read critically, controlled experiments, the four traps, metrics for AI features, consent.
4. **The method** — the one-page experiment plan.
5. **The programs** — a funnel and an A/B check in Java and Python, on a healthy log and a broken one.

Every output shown is real output from the book's programs. Book: Chapter 14.

Bu Tini’s Quiet Checkout

Bu Tini’s warung has an ordering website: menu → cart → address → payment → done. Hundreds open the menu every week; few orders arrive.

Three explanations: the photos, the long menu, the old-fashioned colors. Each is a guess and a week of work.

What nobody knows is *where* people stop. A few well-chosen events answer that.

The steepest drop turns out to be at the address form. Raka builds version B: address remembered, one-tap QRIS. Is B really better?

Answering “is it really better, or just a good week?” honestly is what a controlled experiment is for. Book: Section 14.1.

What People Say and What People Do

Interviews and think-aloud (Chapters 5 and 7) explain **why**, with a handful of people.

Analytics records **what** thousands do, without asking, and cannot say why.

A funnel shows 44% abandon the checkout; only a conversation reveals that typing “Jl. Kenanga Blok C-12 No. 7, RT 03/RW 05” on a small phone is the reason.

Data-informed, not data-driven: numbers inform decisions made by people who understand the context. Book: Section 14.2.1.

Events, Funnels and Metrics

DEFINITION. An *event* records that something happened (who, what, when). A *funnel* is an ordered list of steps with the users reaching each. A *metric* is a number computed from events that tracks a goal. A *conversion rate* is "users who did X divided by users who could have".

$$s_k = \frac{n_k}{n_{k-1}}, k = 2, \ldots, K \tag{14.1}$$

$$c = \frac{n_K}{n_1} = s_2 \cdot s_3 \cdots s_K \tag{14.2}$$

Equation (14.1) is the step conversion; Equation (14.2) the overall conversion. Because steps multiply, improving the weakest step raises the overall conversion most. Book: Section 14.2.2.

A Worked Example by Hand

1,000 open the menu → 610 cart → 430 checkout → 245 payment → 228 done.

Steps: 0.61, 0.705, **0.570**, 0.931. Overall: $228/1000 = 0.228$.

Raise the weakest step to 0.66: overall $0.61 \cdot 0.705 \cdot 0.66 \cdot 0.931 \approx 0.264$, about **36 more orders** per thousand.

Perfect the already strong payment step: at most $0.228 \cdot (1/0.931 - 1) \approx 0.017$, **17 orders**.

The funnel does not say how to fix the checkout; it says where fixing pays. Book: Section 14.2.3.

Choosing Metrics: HEART

Rodden, Hutchinson and Fu (Google, CHI 2010): **H**appiness, **E**ngagement, **A**doption, **R**etention, **T**ask success — and the **Goals–Signals–Metrics** process.

Table 14.1 The HEART Goals–Signals–Metrics process applied to the warung; engagement is left out on purpose.

Category	Goal	Metric
Happiness	Ordering feels easy	Share answering 6–7 on “ordering was easy”
Adoption	Walk-in customers try ordering online	New online customers per week
Retention	Online customers order again	30-day repeat rate
Task success	People who want to order can finish	Overall conversion; time to order
Task success	Orders arrive correct	Wrong-address orders per 100

Table 14.1: for an ordering site, more time on the page means struggle, not engagement. A metric that rises when the experience worsens is worse than no metric. Book: Section 14.3.1.

Indonesia in 2026, Read Critically

APJII 2026: 81.7% of Indonesians online (about 235 million), up from 80.66%; Java 85.95%, Maluku–Papua 69.74%; cities 84.75%, villages 78.18%.

StatCounter, August 2026: 60.59% of *page views* desktop, 38.98% mobile, 0.43% tablet.

Page-view shares count page views on tracked sites — not people. Office computers, tracking-code placement and bots shift them.

Always ask what a number counts before deciding what it means. For Bu Tini, the only device mix that matters is in her own logs. Book: Section 14.3.2.

Controlled Experiments

Week-to-week comparisons mix the design with payday, weather and holidays. **Randomize** visitors between A and B at the same time.

$$z = \frac{p_B - p_A}{\sqrt{\acute{p}(1-\acute{p})\left(\frac{1}{n_A} + \frac{1}{n_B}\right)}} \tag{14.3}$$

$$d \pm 1.96\sqrt{\frac{p_A(1-p_A)}{n_A} + \frac{p_B(1-p_B)}{n_B}} \tag{14.4}$$

Equation (14.3) tests whether the difference exceeds chance ($|z| > 1.96$ means $p < 0.05$); Equation (14.4) gives the size of the effect with its uncertainty, which is what the owner needs. Kohavi, Tang and Xu's book is the standard guide. Book: Section 14.3.3.

How Many Users?

$$n = \frac{(z_{\alpha/2} + z_{\beta})^2 [p_1(1-p_1) + p_2(1-p_2)]}{(p_2 - p_1)^2} \tag{14.5}$$

With $z_{\alpha/2} = 1.96$ (5%, two-sided) and $z_{\beta} = 0.8416$ (80% power), Equation (14.5) gives users per variant.

Halving the effect you want to detect **quadruples** the sample.

Example: 20% → 22% needs 6,507 per variant.

Compute n before the experiment; it decides how long the experiment runs. Book: Section 14.3.3.

The Four Traps

Sample ratio mismatch: unequal groups mean lost users. Check first, stop if $p<0.001$ (Fabijan et al., KDD 2019).

$$\chi^2=\frac{(n_A-e)^2}{e}+\frac{(n_B-e)^2}{e} \tag{14.6}$$

Peeking: stopping when p first dips below 0.05 inflates false positives.

Novelty and learning: run at least one, preferably two, full weekly cycles.

Goodhart's law: a measure that becomes a target stops being a good measure — protect **guardrail metrics**.

In Equation (14.6), $e=(n_A+n_B)/2$ for a planned 50/50 split, with one degree of freedom. Hiding the delivery fee would raise conversion and be a dark pattern; guardrails catch it. Book: Section 14.3.4.

Metrics for AI Features

Acceptance rate rises when suggestions improve — and when people stop reading them.

Add the **edit rate**, the **override-and-outcome** pair (Chapter 13's log) and a **harm guardrail**.

Primary metric: the business outcome (orders completed correctly), with acceptance as context only.

A version that raises acceptance while right final decisions stay flat has made people more trusting, not the system more useful. Book: Section 14.3.6.

Data with Consent

PP 33/2026 (implementing UU PDP 27/2022): consent free, specific, unambiguous, unbundled; withdrawable; processing stops within 3 × 24 hours.

Delivery address: needed for the service. Analytics events: need their **own** consent.

GA4 consent mode: from June 15, 2026, `analytics_storage` and `ad_storage` are governed separately.

The simplest compliance: anonymous first-party events on your own server, never an address or phone number in analytics. Book: Section 14.3.5.

The Experiment Plan

Table 14.2 The experiment plan for the warung's checkout, decided before the first visitor.

Item	Warung checkout experiment
Hypothesis	43% drop at the address step; remembering it and one-tap QRIS will raise orders
Variants	A: current; B: address remembered + QRIS
Primary metric	`order_done` / `view_menu`
Guardrails	Wrong addresses; complaints; payment failures
Minimum detectable effect	+2 percentage points
Sample size	About 7,000 per variant; at least two full weeks
Checks first	Sample ratio ($p \geq 0.001$)
Decision rule	Ship B if the 95% interval is above 0 and guardrails hold

Table 14.2 stops the team from choosing the flattering metric afterwards and turns disagreements into separate experiments. Book: Section 14.4 and Table 14.2.

The Event Log and Its Simulation

One row per event: `user_id,variant,step`.

`make_events.py`: 5,000 simulated visitors, coin-flip assignment, continue-probabilities per step; only checkout → pay differs (A 0.58, B 0.66).

A second log, `events_srm.csv`, silently loses 15% of B's visitors — a logging bug on older phones.

Testing an analysis against a truth you set yourself is how you trust it before real traffic arrives. Book: Section 14.5.1.

The Funnel Program

`funnel.py` / `Funnel.java` read the log into sets of user ids per variant and step: count **distinct users**, not events.

```
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%
```

A `TreeMap` and `LinkedHashMap` in Java keep the output identical to Python's. Book: Sections 14.5.2 and 14.5.3.

The Funnel, Drawn

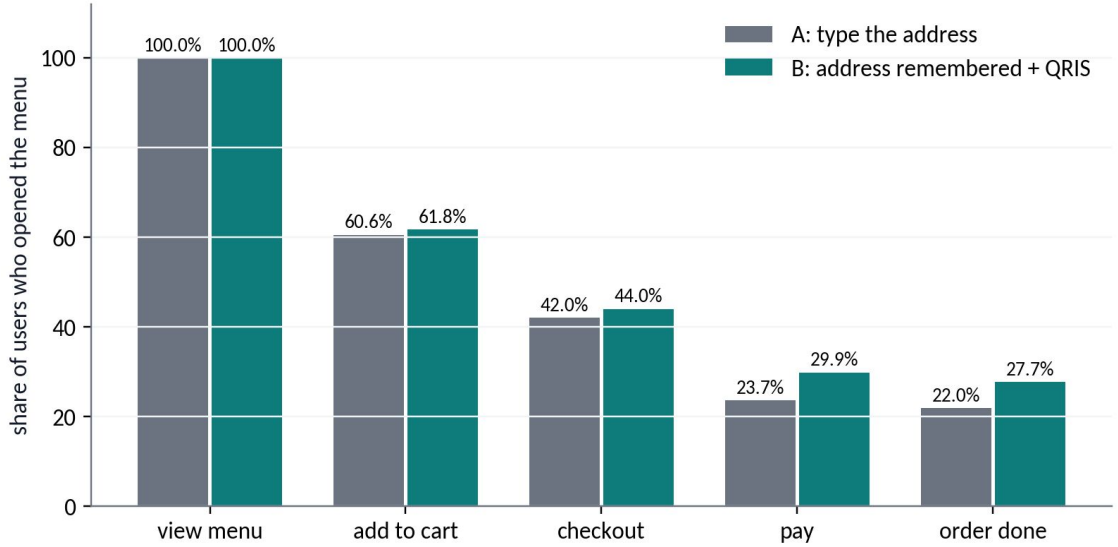


Figure 14.1 The checkout funnel: share of menu openers reaching each step, for A and B.

Figure 14.1: identical until checkout; B keeps 67.9% at payment where A keeps 56.4%, and the gap widens to 27.7% against 22.0%. Descriptive only: it cannot rule out chance or a broken split. Book: Section 14.5.3 and Figure 14.1.

The A/B Check, in Order

`abtest.py` / `AbTest.java`: first the sample ratio (Equation (14.6)); only if it passes, the difference and its interval (Equation (14.4)), the lift, z (Equation (14.3)), and the sample size (Equation (14.5)).

Both p-values come from one function: $\operatorname{erfc}\left(|z|/\sqrt{2}\right)$ and $\operatorname{erfc}\left(\sqrt{\chi^2/2}\right)$.

Java has no `erfc`, so both languages use the same published approximation (error $< 1.2 \times 10^{-7}$) and print the same digits.

The order of the checks is the order of the plan. Book: Sections 14.6.1 and 14.6.2.

Running It on the Healthy Log

```
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
```

The split passes; the whole interval lies above zero and above the two-point minimum. Yet detecting a two-point change would need 6,940 per variant: with a smaller true effect, "not significant" would have meant "we could not tell". Book: Section 14.6.3.

Running It on the Broken Log

```
users: A = 2472, B = 2155
sample ratio check: chi2 = 21.72, p < 0.0001
STOP: sample ratio mismatch - find the bug before reading on
```

Nobody can say whether B's lost 15% would have ordered or left. The result cannot be repaired afterwards.

The program refuses to go on: a result with a sample ratio mismatch is untrustworthy until the cause is found. Book: Section 14.6.4.

Testing Statistical Code

```
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
```

The check that $|z| = 1.96$ *gives* $p = 0.05$ *ties the code to a number every statistics table agrees on; forgetting the* $\sqrt{2}$ *would be caught at once. Book: Section 14.7.*

Making It Better

Segment, but plan the segments — twenty unplanned segments find one “significant” by chance.

CUPED — adjust with pre-experiment behavior to shrink intervals (Deng et al., 2013).

Guardrails next to the primary metric; tap heatmaps; watch the metric after shipping.

Pair the numbers with people — five think-aloud sessions supply the next hypothesis.

A tap heatmap is a cheap cousin of eye tracking that records nothing about the person. Book: Section 14.8.

Using It for Real

Collect events on your own server, after separate analytics consent; never names, phones or addresses.

Assign variants deterministically from a hash of the anonymous id.

Keep a decision log: plan, output, decision — for every experiment.

Pak Made's laundry on Bali uses the same programs with different steps (`open_form`, `choose_service`, `choose_time`, `confirm`). Book: Section 14.9.

HCI in Practice: Profitina

A weekly AI-visibility score per brand, with trend, CSV export, scheduled re-scans and alerts.

Goal: be cited by AI answer engines; signals: measured site properties; metric: the score — so each signal is explained, to avoid Goodhart.

“N new” counters count only top-level posts; one request gathers the dashboard; users can export or purge their data.

Metric definitions are interface decisions. Book: Section 14.10.

Common Mistakes

Easy metrics; events instead of users; page views read as people; before-and-after comparisons.

No sample ratio check; peeking; p without effect size; “not significant” read as “no effect”.

No guardrails; personal data inside analytics.

Each mistake in the book comes with its fix. Book: Section 14.11.

Summary: Five Things to Remember

1. Funnels count distinct users per step; fix the weakest step first (Equation (14.2)).

2. Derive metrics from goals with HEART (Table 14.1); drop metrics that rise when things get worse.

3. Randomize, size in advance (Equation (14.5)), and report the interval (Equation (14.4)), not only p.

4. Check the sample ratio first (Equation (14.6)); protect guardrails.

5. Collect anonymous events, with their own consent, under PP 33/2026.

Book: Section 14.12.

Next Time

Chapter 15 — Emerging Interfaces. Spatial, wearable and ambient computing; sizes as visual angles; gaze and the Midas touch.

Before then: work Exercises 14.1, 14.2, 14.3 and 14.6.

Book: Sections 14.13 and 14.14.

Lecture 15 — Emerging Interfaces: Spatial, Wearable and Ambient Computing

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 15

Lecturer: Irfan Subakti — yifana@gmail.com

Today interfaces leave the rectangle: they float in a headset, sit in glasses, tap on a wrist, glow on a wall. We learn to size anything by the angle it covers, to keep looking from becoming choosing, and to design for the people who will never own the devices. Book: Chapter 15, Sections 15.1–15.12.

What This Meeting Covers

1. **The problem** — Mbah Sumi misses her turn in a puskesmas waiting room.
2. **The idea** — spatial, wearable, ambient; size is an angle.
3. **The theory** — the devices of 2025–2026, gaze and the Midas touch, glanceable wearables, calm technology, comfort, safety, privacy, accessibility, spatial ideas on flat screens.
4. **The method** — the context-of-use checklist.
5. **The programs** — a target-size calculator and a dwell-time analysis in Java and Python.

Every output shown is real output from the book's programs. Book: Chapter 15.

Mbah Sumi and the Queue Screen

Mbah Sumi, 72, waits for her blood-pressure check. Her number, L-017, will appear on a television five meters away and be called over an echoing loudspeaker.

Her eyes are no longer sharp and her hearing is worse. Twice she has missed her turn and waited another hour.

Glasses, watches and headsets promise help — but none of the patients owns glasses, and few own smartwatches.

The first improvement is a larger number on the television, sized from the bench, costing nothing and helping everyone. The same geometry sizes a floating button in a headset. Book: Section 15.1.

Beyond the Rectangle

> **DEFINITION.** A *spatial* interface places content in three-dimensional space around the person. A *wearable* interface is carried on the body and used in short glances. An *ambient* interface is built into the environment and informs at the edge of attention.

In all three, attention is usually elsewhere: walking, cooking, talking, dozing on a bench.

Two demands follow: **easy to perceive at a glance** and **hard to trigger by accident**.

Most of the chapter follows from those two demands. Book: Section 15.2.1.

Size Is an Angle

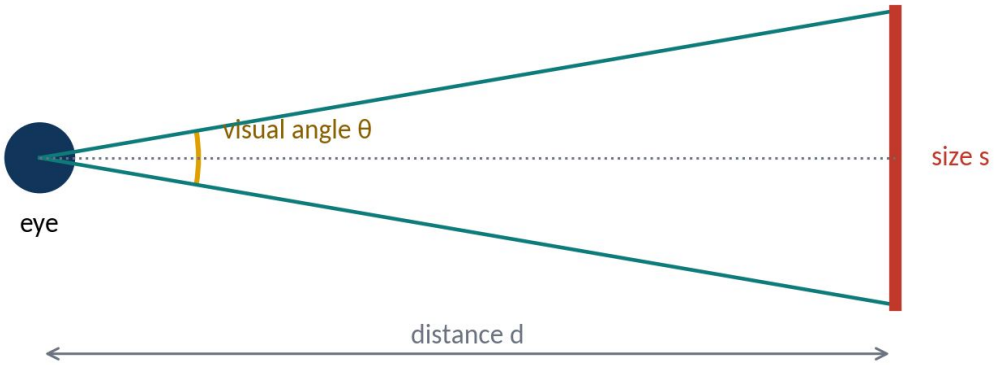


Figure 15.1 The visual angle of an object of size s at distance d.

$$\theta = 2\arctan\left(\frac{s}{2d}\right) s = 2d\tan\left(\frac{\theta}{2}\right) \theta_{\text{deg}} \approx 57.3\frac{s}{d} \tag{15.1}$$

Figure 15.1 and Equation (15.1): the same angle comes from a small object nearby or a large one far away. A thumbnail at arm's length covers roughly one to two degrees. Book: Section 15.2.2.

A Worked Example by Hand

A 4 cm number seen from 5 m: $\theta = 2\arctan(4/1000) \approx 0.46^\circ$, about 27.5 minutes of arc.

The 20/20 line of a Snellen chart is 5 minutes; a person who reads only the 20/100 line needs 25 — the number is at the edge of legibility.

For 1.5° from 5 m: $s = 1000 \cdot \tan(0.75^\circ) = 13.1$ cm — about three times taller.

On a 43-inch television (screen about 53 cm high), a 13 cm number still leaves room for the service name and the next numbers. Book: Section 15.2.3.

The Devices of 2025–2026

Table 15.1 Devices that shape spatial and wearable design in 2026, with their inputs and outputs.

Device	Main inputs	Main outputs
Apple Vision Pro with M5 (Oct. 2025)	Eyes, hands, voice	Micro-OLED up to 120 Hz, spatial audio
Samsung Galaxy XR, Android XR (Oct. 2025)	Eyes, hands, voice with Gemini	Displays with passthrough
Meta Ray-Ban Display + Neural Band (Sep. 2025)	EMG wristband gestures, voice	Small display off to one side
Ray-Ban Meta Audio (Sep. 2026)	Voice, touch	Audio only, no camera
Android XR audio glasses (fall 2026)	Voice with Gemini, frame tap	Audio
Apple Watch, watchOS 26	Touch, double tap, wrist flick	Screen, haptics

Table 15.1: inputs move away from surfaces; outputs shrink to glances and sounds; an AI assistant sits in most of them. In April 2026 MacRumors reported weak Vision Pro sales despite the M5 refresh: expensive headsets remain a small market. Book: Section 15.3.1.

Looking Is Not Choosing

The eyes move three or four times a second, whether or not the person wants anything.

Midas touch (Jacob, 1990): if everything you look at activates, you cannot look without triggering.

Solutions: a **second signal** (look + pinch on Vision Pro and Galaxy XR; EMG gestures on the Neural Band) or a **dwell time**.

Gaze is also imprecise and varies by person (Feit et al., CHI 2017); Apple asks for spatial targets of at least 60 points, versus 44 on iOS.

Dwell matters where hands are not available: motor impairments, sterile or dirty hands, public eye-tracking kiosks. Book: Section 15.3.2.

Wearables: Glance, Tap, Go

Show one thing: the number and how many are ahead, not a menu.

Prefer haptics: a tap on the wrist is private and needs no reading.

Offer gestures that need no aiming: double tap (2023) and wrist flick (watchOS 26) help anyone with a busy hand.

Few, distinct patterns: a gentle double tap = “get ready”; a long, strong buzz = “go now”.

Two patterns can be learned; eight will be confused. Each pattern gets a single meaning. Book: Section 15.3.3.

Voice on the Body

Audio glasses rely almost entirely on voice. Spoken output is linear and fleeting: **most important word first** — “L-017, your turn, door three”.

Repeat before acting, as the laundry assistant of Chapter 13 repeated the order.

Speaking is public: a patient may not want to say a symptom aloud; offer a silent alternative.

Plan for real languages: recognizers trained on standard Indonesian handle Javanese or Madurese speech less reliably — test with real users’ voices.

Voice and wrist gestures are the two inputs that need no aiming, and both help people whose hands or eyes are busy. Book: Section 15.3.3.

Calm Technology

Weiser (1991): computers disappear into everyday objects. Weiser and Brown (1996): technology should be **calm**, moving information between the center and the periphery of attention.

A lamp above each door that brightens as the queue moves; a colored band on the television.

Ambient signals carry little information: pair them with a precise channel for the moment of action.

Ambient says “pay attention”; the screen, the voice or the wrist says “go now”. Book: Section 15.3.4.

Comfort, Safety, Privacy, Accessibility

Comfort: cybersickness when eyes and balance disagree — stable worlds, short sessions, never required.

Safety: people wearing displays notice less of the world.

Privacy: cameras record bystanders; images and voices are personal data under UU PDP 27/2022.

Accessibility: every input mode excludes someone; W3C's XR Accessibility User Requirements ask for alternatives, captions and customization.

The rule: at least two ways in and two ways out, one of which works without the new device. Book: Section 15.3.5.

Spatial Ideas on Flat Screens

Liquid Glass (June 2025, iOS 26): translucent controls that refract the content behind them, inspired by visionOS.

Text on glass has no fixed background: a label passing 4.5:1 over white may fail over a photo.

Test legibility over the worst background; respect “reduce transparency” and “increase contrast”.

Depth, motion and translucency can guide attention and also cost legibility; offer the calm, solid version as a setting. Book: Section 15.3.6.

The Context-of-Use Checklist

Table 15.2 The context-of-use checklist answered for the puskesmas waiting room.

Question	Puskesmas waiting room
How far is the person from the information?	Bench to wall: 3–5 m
Are the hands free?	Often not: children, bags, canes
Where is the attention?	On companions, a phone, dozing
Who else is present?	Other patients; no names on screen
What devices do people have?	Basic phones; few watches; no glasses
What abilities vary?	Low vision, hearing loss, low literacy, many languages
What happens if it fails?	Staff call the number aloud

Table 15.2 usually shows that the new device is the last step: first a larger number, then a calm lamp, then optional SMS and a wrist tap; glasses last, if at all. Book: Section 15.4.

The Target-Size Calculator

`visual_angle.py` / `VisualAngle.java` apply Equation (15.1) at four distances: phone, kiosk, virtual panel, wall television.

```
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
```

A 1 cm button, fine at 1.91° in the hand, shrinks to 0.19° across the room. Book: Section 15.5.

Any Angle from the Command Line

```
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
```

`python visual_angle.py 0.5 1.5`: at 3 m a 1.5° number needs 7.85 cm; at 5 m, 13.1 cm, as computed by hand.

The helper `g` in Java prints 1.0 as "1", like Python's `%g`, so the headings match. Book: Section 15.5.

Choosing a Dwell Time: the Data

An eye-controlled kiosk for patients who cannot use their hands. How long must a look last?

`make_gaze.py`: 1,000 simulated glances; about one in five intended (patience median 1,200 ms), the rest reading glances (median 260 ms, near the 225–250 ms fixations of silent reading).

$$J(T) = 3F(T) + M(T) \tag{15.2}$$

In Equation (15.2), F is false activations per 100 glances and M the percentage of intended selections missed; one false activation counts as three misses — a design decision, not a law. Book: Sections 15.6.1 and 15.6.2.

The Result

```
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
```

At 300 ms almost a third of all glances would trigger a button. The lowest cost is at 700 ms. Book: Section 15.6.3.

The Trade-off, Drawn

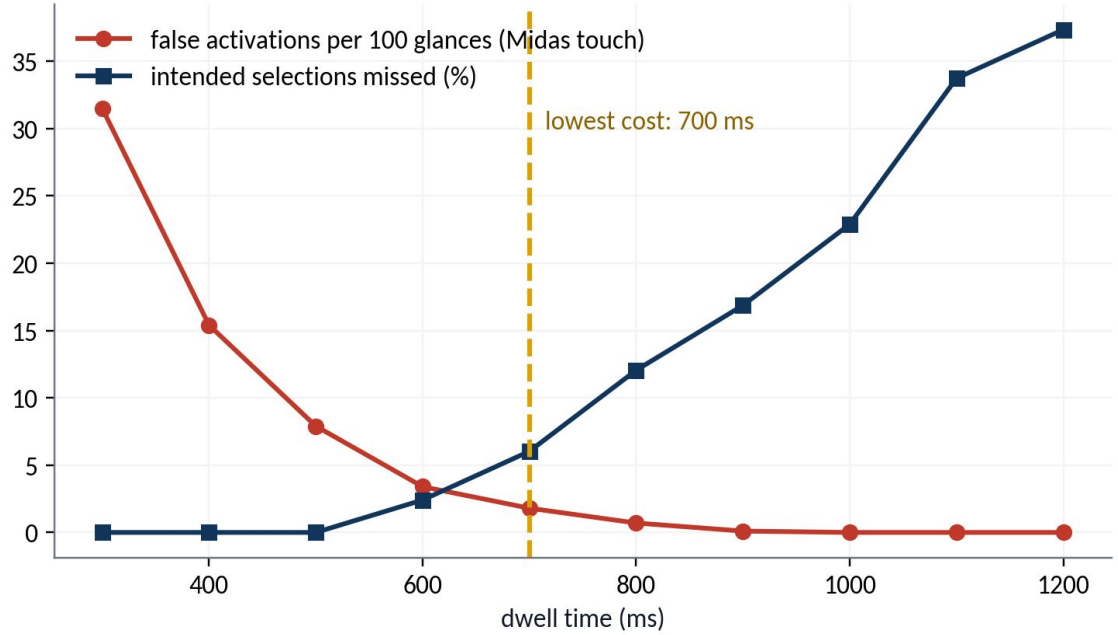


Figure 15.2 False activations fall and missed selections rise as the dwell grows.

Figure 15.2: neither error vanishes at any dwell. The only way out is a second signal — hence look + pinch, and a confirmation step on a public gaze kiosk. Book: Section 15.6.3 and Figure 15.2.

Testing Properties, Not Only Numbers

```
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
```

The dwell tests check a property that stays true when the simulated log is replaced by real data. Book: Section 15.7.

Making It Better

Confirm, do not only dwell: select at 700 ms, commit with a separate YES.

Adapt the dwell to the person; show a filling ring while dwelling.

Size from the worst reasonable case: the farthest seat, the patients' eyesight, the tracker's measured error.

Design ambient and precise channels together; a lamp that disagrees with the screen teaches people to ignore it.

Book: Section 15.8.

Using It for Real

Farthest bench 6 m → `python visual_angle.py 1.5` → a 15.7 cm number, bold and high-contrast.

A colored strip above each door: grey → amber at five numbers left → teal when called.

Optional SMS at three numbers ahead, after separate consent; a watch shows it on the wrist.

Cost order = reach order: font setting, lamp, SMS, gaze kiosk.

The same method sizes a bus-terminal departure board, a factory warning light and a market price sign. Book: Section 15.9.

HCI in Practice: Profitina

Through audio glasses or a voice assistant, people hear **one** spoken answer, perhaps naming one source.

Being the quoted source matters more when there is no page of links — the case for Generative Engine Optimization.

Profitina's own interface: SSE-driven "N new" counters as a quiet peripheral signal; notifications gathered in one cockpit; AI run locally.

Book: Section 15.10.

Common Mistakes

Sizes without distances; designing from the nearest seat; letting a look activate.

Too many haptic patterns; ambient signals asked to carry precise information.

Requiring the new device; forgetting bystanders; ignoring comfort.

Each mistake in the book comes with its fix. Book: Section 15.11.

Summary: Five Things to Remember

1. Spatial, wearable and ambient interfaces must be easy to perceive at a glance and hard to trigger by accident.

2. Size is a visual angle; convert it with Equation (15.1) for the farthest realistic distance.

3. Looking is not choosing: add a second signal, or choose a dwell with the trade-off of Figure 15.2.

4. Ambient says “pay attention”; a precise channel says “go now”.

5. Answer Table 15.2 first: the cheapest improvement for everyone comes before the new device.

Book: Section 15.12.

Next Time

Chapter 16 — Capstone. The whole book on one problem: a queue kiosk for Mbah Sumi's puskesmas, researched, built, tested and improved.

Before then: work Exercises 15.1, 15.3, 15.4 and 15.6.

Book: Sections 15.13 and 15.14.

Lecture 16 — Capstone: Designing and Evaluating a Complete Interactive System

Course: Human-Computer Interaction (IMK)

Book: Irfan Subakti, *Human-Computer Interaction*, Chapter 16

Lecturer: Irfan Subakti — yifana@gmail.com

Today the whole book meets one real problem: a queue kiosk for a community health center, researched, defined, built in Python and Java, tested with users including older adults, and improved from the evidence. Book: Chapter 16, Sections 16.1–16.14.

What This Meeting Covers

1. **The brief** — Puskesmas Sehat Selalu and the rule “nobody may be worse off than today”.
2. **The path** — four phases, and every chapter from 2 to 15 as a decision.
3. **Discover and define** — research at the desk, personas, requirements, human-factors budget, accessibility and law.
4. **Develop** — paper prototype, design decisions, the running kiosk.
5. **Deliver and iterate** — pilot test, SUS, UMUX-Lite, completion, contrast; findings, severities, retest; presenting decisions.

Every screenshot is of our own running kiosk, and every number is real program output. Book: Chapter 16.

The Brief

About two hundred patients on a busy morning; one registration desk; a line out of the door; four services: general, KIA, dental, older adults.

The head of the puskesmas wants a touchscreen kiosk for patients who know their service. A team of three students, one donated tablet, one semester.

One rule: **nobody may be worse off than today.**

A kiosk fast for young people and baffling for Mbah Sumi would shorten the line and still make the puskesmas worse for those who need it most. Book: Section 16.1.

A Complete Interactive System

More than a screen: the place, the paper slip it replaces, the desk it relieves, the nurse, the patients who will not use it, and the law.

ISO 9241-210: understand the context, specify requirements, produce designs, evaluate — and repeat.

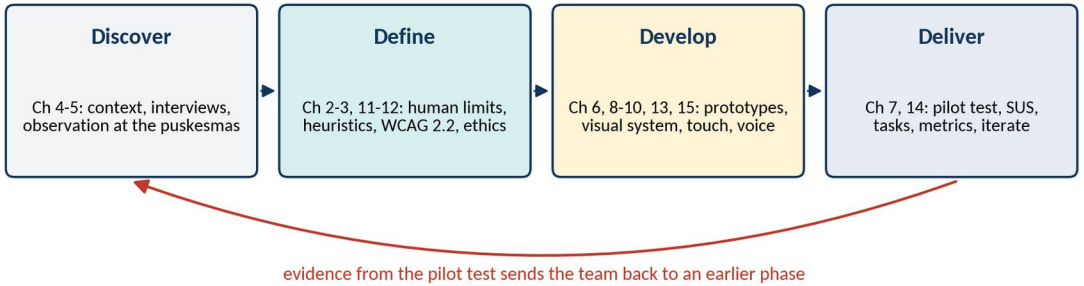


Figure 16.1 The capstone path in four phases, with the chapters each phase applies and the loop back.

Figure 16.1 uses the Design Council's Double Diamond. The arrow at the bottom matters most: evidence sends the team back. Book: Section 16.2.1.

Every Chapter as a Decision

Table 16.1 Selected chapters and the decision each contributed to the queue kiosk.

Ch	What it decided for the kiosk
2	Four choices; large targets near the hand; nothing to remember
3	Visible status; the desk's own words; error prevention
5	Two mornings of observation; eight interviews
6	Paper prototype, then working HTML
7	Pilot test with eight: tasks, SUS, UMUX-Lite
11	Contrast, target size, skip link, language, status messages, larger text
12	No personal data; no pressure; urgent help from a person
13	**No AI**: nothing needs a guess, and a guess could harm
14	HEART metrics; anonymous counts only
15	Sizes from viewing distance

Table 16.1 shows ten of the fourteen rows; the book adds Chapters 4, 8, 9 and 10. Choosing not to use a technology is a design decision too. Book: Section 16.2.2 and Table 16.1.

Discover: Research at the Desk

Finding 1: about two thirds were returning patients who knew their service and queued only for a slip.

Finding 2: some older patients “do not dare” to touch public screens.

Finding 3: the clerk moves seriously ill patients forward; a kiosk might hide them.

Finding 4: Indonesian, Javanese and Madurese; a request for “simple Indonesian, not officials’ Indonesian”.

Observation with permission, no names or faces recorded; six patients and two staff interviewed. Book: Section 16.3.

Personas and the Key Scenario

Bu Lestari, 34: KIA every month, baby in one arm, wants the number in ten seconds.

Mbah Sumi, 72: alone every second Tuesday, reads large print only, never used a kiosk, will not use one that makes her feel slow.

Scenario sentence that became requirement 1: *she must be able to take a number with one touch, or ignore the kiosk entirely and go to the desk as before.*

Design for the least advantaged persona; the others benefit too. Book: Section 16.3.

Define: Testable Requirements

1. One touch for a returning patient. **2.** No personal data at all.

3. Urgent symptoms → a person at once. **4.** Indonesian and English, desk words, larger text.

5. Nobody forced: desk stays open, slip and kiosk number equivalent. **6.** Number and people ahead in large type.

Each requirement is traced to a finding and to a chapter. Book: Section 16.4.1.

The Human-Factors Budget

$$T_{\text{choose}} = a + b \log_2(n+1) \tag{16.1}$$

$$ID = \log_2\left(\frac{D}{W} + 1\right) \tag{16.2}$$

Hick (Equation (16.1)): four services $= \log_2 5 = 2.32$ bits; eight would be $\log_2 9 = 3.17$ bits.

Fitts (Equation (16.2)): 64-pixel buttons at up to 300 pixels give 2.51 bits; 32-pixel links give 3.37 bits.

Four choices and tall, full-width buttons: cheap for the team, good for everyone, best for unsteady hands. Book: Section 16.4.2.

Accessibility as a Requirement

Table 16.2 Selected WCAG 2.2 success criteria and how the kiosk meets them.

WCAG 2.2 criterion	How the kiosk meets it
1.4.3 Contrast (Minimum)	Lowest text pair 7.56:1
1.4.11 Non-text Contrast	Focus ring changed from 2.27:1 to 12.46:1
2.4.1 Bypass Blocks	Skip link, visible on focus
2.5.8 Target Size (Minimum)	Buttons 64 px tall, full width
3.1.2 Language of Parts	`lang` on the link to the other language
4.1.3 Status Messages	Number in a `role="status"` region

Table 16.2: WCAG 2.2 became ISO/IEC 40500:2025 in October 2025. Book: Section 16.4.3 and Table 16.2.

Law and Ethics as Constraints

UU 8/2016 and **PP 42/2020**: accessible public services are a right; a puskesmas is a public service.

European Accessibility Act (since June 28, 2025) covers self-service terminals — a sign of where the world is going.

UU PDP 27/2022 and PP 33/2026: the kiosk collects nothing personal, so it needs no consent screen.

Chapter 12: no timers, no advertising, no pressure; urgent help points to a person, never a diagnosis.

The cheapest way to comply with a data-protection law is to collect nothing it protects. Book: Section 16.4.3.

Develop: the Paper Prototype

Sketch 1 opened with a welcome screen and “Start”: all four patients touched the logo; one asked “is it broken?”.

Sketch 2 put the four services first: all four chose correctly without hesitation.

Mbah Sumi: the words were “too small for old eyes” — the larger-text mode began here.

A ten-minute prototype removed a whole screen from the design. Book: Section 16.5.1.

The Running Kiosk

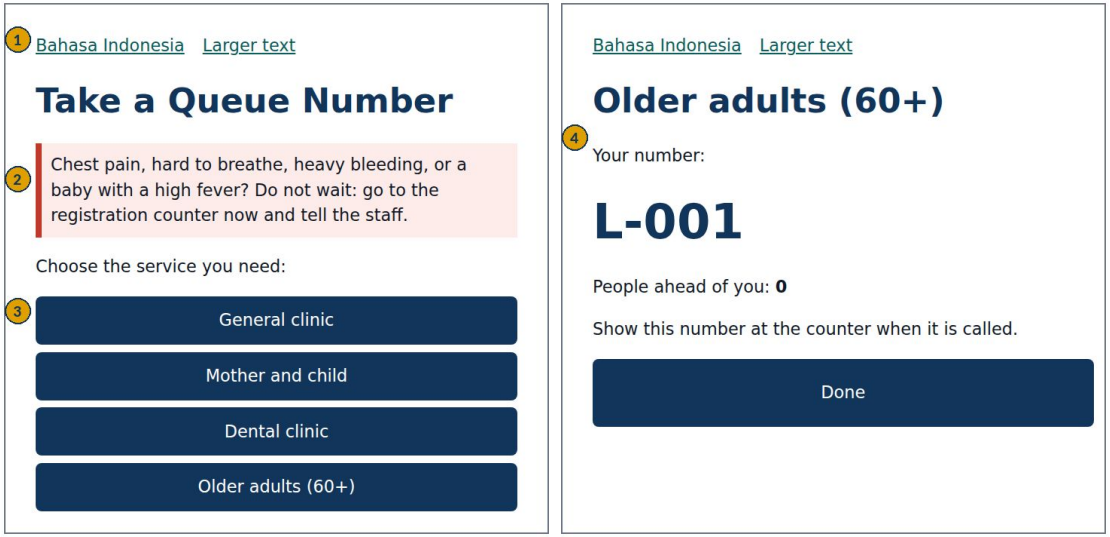


Figure 16.2 The running kiosk: 1 language and text size, 2 urgent-help note, 3 the four services, 4 the number and the people ahead.

Figure 16.2 was captured with Playwright from our own server. Deliberately absent: an estimated waiting time (a guess) and JavaScript (a way to break). Book: Section 16.5.2.

Building It in Python and Java

`kiosk_app.py` (Flask, port 4160): `TEXT` holds both languages; `page` adds `lang`, the skip link, tools and style; routes `/`, `/take`, `/status.json`.

`KioskServer.java` (`com.sun.net.httpserver`, port 4161) serves **byte-identical HTML**, checked with `curl`.

A Java lesson: `Map.of` rejects `null` keys — the first version crashed without `lang`. Now: a default, and a handler that logs and answers instead of hanging.

A kiosk must never hang silently: a patient before a frozen screen does not know whether to wait or leave. Book: Sections 16.6.1 and 16.6.2.

Driving It from a Terminal

```
$ 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
```

```
L-001
ahead of you: 0
L-002
ahead of you: 1
```

The key lines of the two returned pages; `/status.json` then reports two numbers given for L. Book: Section 16.6.3.

The Pilot Test

Eight participants: three 18–39, two 40–59, three 60+. Enough to find serious problems (Nielsen and Landauer), far too few for precise averages.

T1 take a general-clinic number. **T2** switch to English and take a mother-and-child number. **T3** enlarge the text and take an older-adults number.

Then ten SUS statements and two UMUX-Lite statements; data in `sus.csv`, `umux_lite.csv`, `tasks.csv` (a practice data set).

Think-aloud notes explain the numbers. Book: Section 16.7.1.

The Measures

$$\text{SUS} = 2.5\left[\sum_{i\in[1,3,5,7,9]}(a_i - 1) + \sum_{i\in[2,4,6,8,10]}(5 - a_i)\right] \tag{16.3}$$

$$\hat{p} = \frac{x + z^2/2}{n + z^2},\ \hat{p} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n + z^2}} \tag{16.4}$$

Equation (16.3): 0 to 100, not a percentage; about 68 is average. UMUX-Lite: two items, SUS-comparable $= 0.65 \cdot \text{UMUX-Lite} + 22.9$.

Equation (16.4): the adjusted Wald interval stays sensible at 8 of 8. Times: geometric mean.

Sauro and Lewis recommend the adjusted Wald interval for small samples. Book: Section 16.7.2.

Color Contrast, Computed

$$L = 0.2126\,R_{\text{lin}} + 0.7152\,G_{\text{lin}} + 0.0722\,B_{\text{lin}}, \quad \text{ratio} = \frac{L_1 + 0.05}{L_2 + 0.05} \tag{16.5}$$

Each sRGB channel is linearized first; Equation (16.5) then gives the ratio of lighter to darker.

Text needs 4.5:1; focus rings and other non-text elements 3:1.

`evaluate.py` *and* `Evaluate.java` *check every color pair of the style sheet, including the draft focus ring. Book: Sections 16.7.2 and 16.7.3.*

The Results

```
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
```

The contrast part of the output shows every pair passing except the draft focus ring (2.27:1), already replaced by navy (12.46:1). Book: Section 16.7.4.

Reading the Results

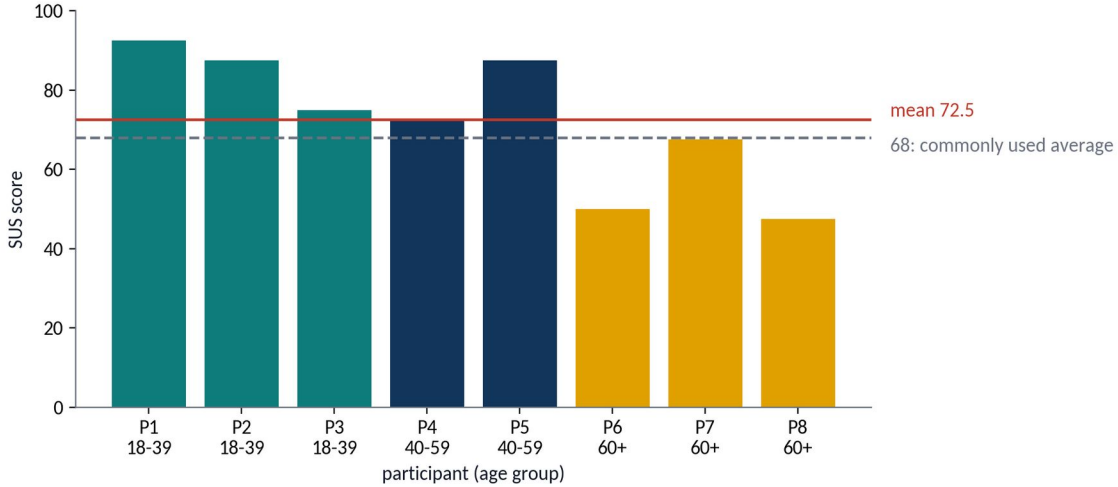


Figure 16.3 SUS by participant and age group; all three participants aged 60+ scored below 68.

Figure 16.3: the mean 72.5 looks fine, the interval [58.3, 86.7] says eight people cannot tell, and the pattern says the kiosk is not yet good enough for older adults. T2 failed twice ("did not see" the language link); T3 once ("nothing happened"). Book: Section 16.7.4.

Iterate: Findings to Changes

Table 16.3 Pilot findings with Nielsen severity ratings and planned changes.

Finding	Severity	Change for iteration 2
Language link not seen by older users	3	Two large language buttons above the services
Larger-text change not noticed	2	Larger text in one touch; a confirmation line
Older adults' SUS below 68	3	Retest with at least five older adults
Focus ring 2.27:1	2	Fixed: navy ring, 12.46:1
Fear of touching the kiosk	2	A helper nearby in month one; "The desk is open as usual"

Table 16.3 ties every change to evidence and ends with a retest judged by T2 and T3 for older adults, not by the overall mean. Book: Section 16.8 and Table 16.4.

Testing the Code

```
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
```

The servers: identical HTML from both; an invalid service code redirects (302 Flask, 303 Java); Playwright fails if an expected button is missing.

Book: Section 16.9.

Presenting Design Decisions

1. The problem and the people. **2.** Requirements traced to findings and chapters.

3. The design, each decision tied to a requirement. **4.** The evidence, including the disappointing results.

5. What changes next, and how it will be judged. **6.** What was deliberately *not* done, and why.

Point 6 often persuades sceptics most: it shows fashionable options were weighed and rejected for reasons tied to users. Book: Section 16.10.

Using It for Real

Browser in kiosk mode; mounted at wheelchair height; a thermal printer for a paper number.

Counts in a small database, reset each morning; a `/call` endpoint for the desk (Exercise 16.6).

The status endpoint feeds the waiting-room television, sized as in Chapter 15; a production server with automatic restart.

Village offices, bank branches, post offices and vehicle-registration offices use the same process — and the same rule. Book: Section 16.11.

HCI in Practice: Profitina

Research question: how can a non-engineer understand why AI answer engines cite, or ignore, a website? Answer: every signal explained in plain words.

Self-service UU PDP compliance: consent, export, email and password change, full purge — in the product.

Continuous evaluation: well over a thousand automated tests gate every release; a health-verified staged release; a visible feedback pipeline.

The same cycle as the kiosk, at a larger scale. Book: Section 16.12.

Common Mistakes

Starting with the solution; designing for the average user; treating a small test as precise.

Checking accessibility at the end; adding technology because it is available; collecting data you do not need.

Stopping after one iteration; presenting only the good news.

Each mistake in the book comes with its fix. Book: Section 16.13.

Summary: Five Things to Remember

1. A complete system includes the service around the screen; run the cycle of Figure 16.1 and loop back.

2. Every method earns its place — or is left out on purpose (Table 16.1).

3. Human-factors laws (Equation (16.1), Equation (16.2)) and WCAG 2.2 (Table 16.2) are requirements from day one.

4. Measure with SUS (Equation (16.3)), intervals (Equation (16.4)) and contrast (Equation (16.5)); report by the groups that matter.

5. Turn findings into prioritized, evidence-tied changes and a retest (Table 16.3).

Book: Section 16.14.

After This Course

Pick a real place near you where people struggle with an interface: a warung, a puskesmas, a village office, a laundry.

Run the whole cycle once: watch, define, prototype, build, test with five people, fix, test again.

Work Exercises 16.1–16.6; they rehearse every calculation and decision of today.

The rule of Puskesmas Sehat Selalu travels well: nobody may be worse off than before. Book: Sections 16.15 and 16.16.