
Module 5: UX and Design Collaboration

Understand UX craft well enough to review designs, run usability tests, and work productively with product designers.



AUTHOR

Varun Kumar

Product Management Master Handbook
v0.1 · 23 October 2026

Built from the 55-lecture course study guide and extended to the 12-module Product Core curriculum.
Hypothetical examples are labelled; no real company results are invented.

Contents

5.1 UX versus UI	3
5.2 Design Thinking and Human-Centred Design	4
5.3 Nielsen's 10 usability heuristics	4
5.4 Laws of UX that every PM should know	6
5.5 Information architecture, user flows and task flows	7
5.6 Wireframes and prototypes: low to high fidelity	9
5.7 Design systems, responsive design and mobile vs web	11
5.8 Accessibility	11
5.9 Usability testing	13
5.10 A/B testing in UX	14
5.11 Design critique and working with product designers	14
5.12 Figma fundamentals for PMs	15
5.13 Journey mapping, onboarding and checkout optimisation	16
5.14 Error states and empty states	17
5.15 UX metrics	18
5.16 Design handoff	18
5.17 Exercises with model answers	19
5.18 Interview perspective	20
5.19 Key takeaways	20
5.20 Check your understanding	22
5.21 Recommended resources	23

MODULE 5

UX and Design Collaboration

Understand UX craft well enough to review designs, run usability tests, and work productively with product designers.

≈ 10 h study

About the screens in this module

All wireframes are **illustrative, simplified sketches made for teaching**. They are *not* screenshots of any real company's interface and must never be presented as such.

5.1 UX versus UI

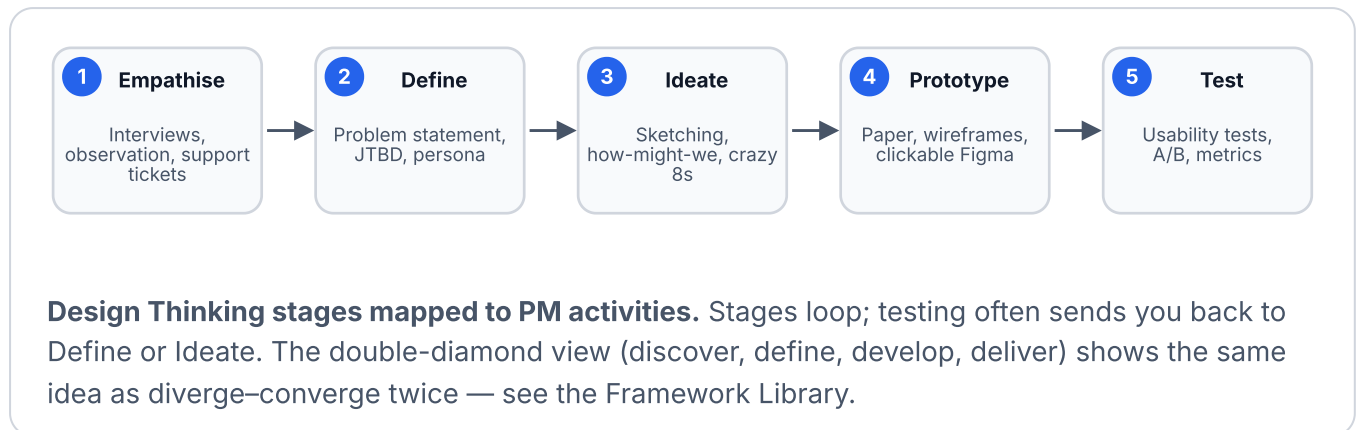
From your course notes (Lecture 16)

UI is the interface people see and use: text, controls, layout, colour, typography. **UX** is the complete experience of achieving a goal, including usability, accessibility, waiting, trust, service and what happens when something goes wrong. *A large readable Pay button is UI. A checkout that clearly shows the total, works with a screen reader and explains a failed payment is part of UX.*

	UX (experience)	UI (interface)
Scope	The whole journey, before/during/after the screen	Visual and interactive layer
Questions	Can people achieve their goal? Do they trust it?	Is it clear, consistent, accessible, attractive?
Artefacts	Research, journeys, flows, prototypes, test findings	Style guides, components, high-fidelity screens
Failure looks like	Users abandon, get lost or distrust	Inconsistent, hard to read or hard to tap

5.2 Design Thinking and Human-Centred Design

Design Thinking is a human-centred, iterative approach to problem solving with five stages (Stanford d.school): **Empathise** → **Define** → **Ideate** → **Prototype** → **Test**. **Human-Centred Design (HCD)**, standardised in ISO 9241-210, emphasises understanding users, tasks and environments; involving users throughout; evaluating with users; iterating; considering the whole user experience; and using multidisciplinary skills.



From your course notes (Lecture 16)

A possible design sequence is: define the problem and requirements, map the flow, sketch or wireframe, build a prototype, test, revise and implement — *iterative, not mandatory*. Include users with varied needs and abilities. A **wireframe** shows structure; a **prototype** lets people experience or test interactions.

5.3 Nielsen's 10 usability heuristics

Jakob Nielsen's ten general principles for interaction design are a fast way to review any screen (a **heuristic evaluation**).

#	Heuristic	Meaning	Example (checkout)
1	Visibility of system status	Keep users informed with timely feedback	"Processing payment..." spinner with a timeout message
2	Match between system and real world	Speak the user's language	"Delivery fee", not "ancillary charge code 4"
3	User control and freedom	Easy undo/exit	"Edit cart" and "Cancel" always available
4	Consistency and standards	Follow conventions	Same "Pay" button position on every step
5	Error prevention	Prevent problems before they occur	Disable "Pay" until a payment method is chosen; confirm address
6	Recognition rather than recall	Make options visible	Show saved cards, not "enter the last four digits"
7	Flexibility and efficiency of use	Shortcuts for experts	"Order again", saved addresses
8	Aesthetic and minimalist design	Remove irrelevant info	Hide secondary promotions during payment
9	Help users recognise, diagnose, recover from errors	Plain-language errors with a fix	"Your card was declined. Try another card or UPI."
10	Help and documentation	Searchable, task-focused help	Link to refund FAQ from the order screen

From your course notes (Companion ch. 3; Picture lesson)

Review **visibility of status, clear labels, error recovery, keyboard and screen-reader access, colour contrast and understandable prices**. A *design review (heuristic evaluation)* can find possible problems but does not replace testing with real users. An AI review is a useful draft, not a substitute.

How to run a heuristic evaluation: 3–5 evaluators review independently against the list, rate each problem's **severity** (0 cosmetic ... 4 catastrophe), then consolidate. It is cheap and fast but misses problems real users hit.

5.4 Laws of UX that every PM should know

5.4.1 Jakob's Law

Users spend most of their time on *other* sites/apps, so they expect yours to work the same way.

Implication: follow conventions (cart icon top-right, logo links home) unless you have a strong, tested reason. Innovate on value, not on basic mechanics.

5.4.2 Hick's Law

The time to choose increases with the number and complexity of choices.

Hick-Hyman law: $T = b \times \log_2(n + 1)$

T = decision time; n = number of equally likely choices; b = empirically fitted constant

Illustrative calculation (b = 0.1 s, a made-up constant for illustration)

7 menu options: $T = 0.1 \times \log_2(8) = 0.1 \times 3 = \mathbf{0.30\ s}$. 15 options: $T = 0.1 \times \log_2(16) = 0.1 \times 4 = \mathbf{0.40\ s}$. Doubling the options adds one bit, not double the time — but real interfaces also add scanning and comprehension costs. **Implication:** group, prioritise, progressively disclose; don't hide key actions.

5.4.3 Fitts's Law

The time to reach a target depends on distance and size: closer and larger targets are faster to hit.

Fitts's law (Shannon form): $T = a + b \times \log_2(D / W + 1)$

D = distance to target; W = target width (along the movement axis); a, b = constants

Index of difficulty ID = $\log_2(D / W + 1)$

Worked example

A button 40 px wide at a distance of 400 px: $ID = \log_2(400/40 + 1) = \log_2(11) = \mathbf{3.46\ bits}$. Making it 80 px wide: $ID = \log_2(6) = \mathbf{2.58\ bits}$ — about **25% lower**. For touch, thumb reach zones and minimum target sizes ($\approx 44 \times 44$ pt iOS guidance; 48×48 dp Android/Material guidance; WCAG 2.2 requires at least 24×24 CSS px as a minimum) matter more than the formula. **Implication:** make primary actions large and in easy reach; keep destructive actions small and apart.

5.4.4 Gestalt principles

How people perceive visual groups:

Principle	Idea	Use
Proximity	Close items are grouped	Space between form sections
Similarity	Similar items are related	Same style for all primary buttons
Common region	Items in a shared boundary are grouped	Card around the order summary
Continuity	Eyes follow smooth lines	Aligned lists and stepper progress
Closure	We complete incomplete shapes	Minimal icons
Figure-ground	We separate focus from background	Modal overlays dim the page
Focal point	Distinct items draw attention	One highlighted primary CTA

5.4.5 Cognitive Load Theory

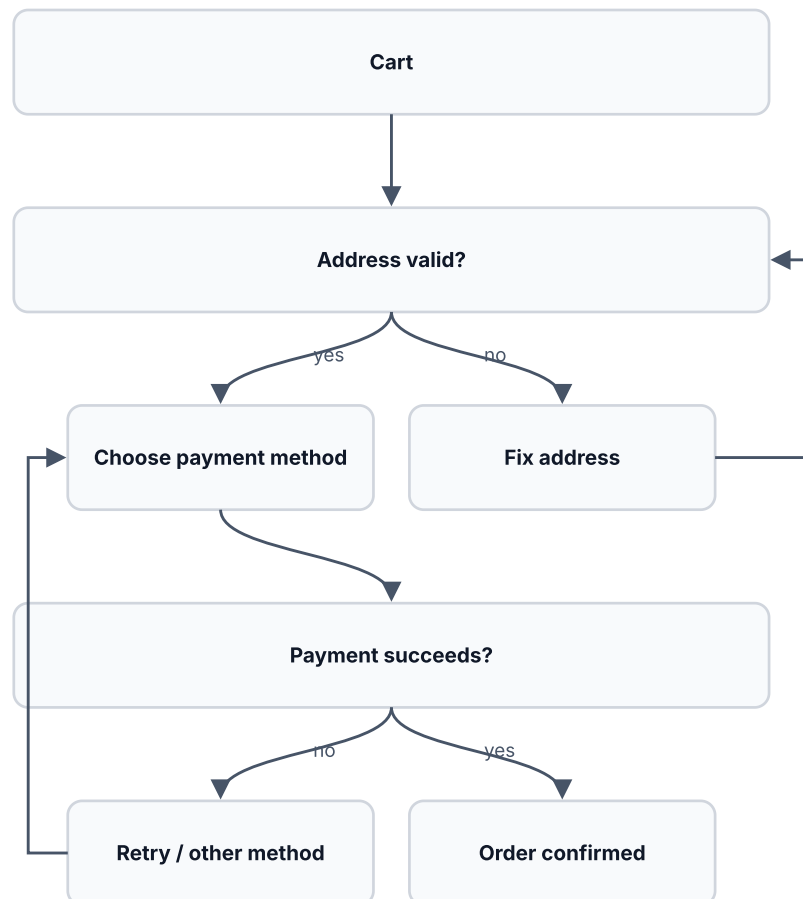
Working memory is limited. **Intrinsic load** is the inherent difficulty of the task; **extraneous load** is added by poor design (clutter, confusing labels); **germane load** is effort that builds understanding. Designers reduce extraneous load: chunk information, use progressive disclosure, defaults, recognisable patterns and consistent layouts. Miller's "7±2" is often over-quoted — treat it as a reminder that capacity is small, not as a design rule.

5.5 Information architecture, user flows and task flows

- **Information architecture (IA)** organises content so people can find it: structure, **navigation**, **labelling**, **search**. Methods: **card sorting** (how users group items), **tree testing** (can users find items in your proposed structure?).
- A **user flow** maps paths and decisions through the product toward a goal; a **task flow** is one task in detail, typically for a single user type.

From your course notes (Lecture 5; Companion ch. 3)

A journey can contain many flows. Include branches: valid address, invalid address, payment failure, retry and cancellation — and label the desired outcome for each. Start/end markers and connectors make the sequence readable. Use the appropriate level of detail; *a functional flow, not backend architecture*.



Task flow — pay for an order (hypothetical). Decision points and recovery loops are where bugs and abandonment hide. Every “no” branch needs a designed state.

5.6 Wireframes and prototypes: low to high fidelity

Fidelity	What it is	Use for	Risk
Sketch / paper	Hand-drawn boxes	Exploring many ideas in minutes	Hard to share remotely
Low-fi wireframe	Grey boxes, real labels, no visual polish	Layout, flow, content priority	People critique visuals instead
High-fi mockup	Real visual design	Brand, UI details, stakeholder review	Looks “finished”; invites premature sign-off
Interactive prototype	Linked screens with transitions	Usability tests, stakeholder demos	Not connected to a real backend; may mislead about feasibility

Cart

Your cart

Veg biryani × 1

₹260

Raita × 1

₹60

Subtotal

₹320

Taxes and fees are calculated at payment.

Proceed to pay

1

Total is unknown until the payment step.

2

Vague promise: 'fees at payment'.

3

Primary button hides the cost.

Payment

Payment

Subtotal

₹320

Delivery fee

₹45

Platform fee

₹5

Taxes

₹18

Total

₹388

Pay ₹388

1

Total is 21% above the subtotal the user saw.

2

Effort is already invested — a trust shock right when money is requested.

Before — checkout with late fees (illustrative wireframe, not a real product screenshot).

Hypothetical baseline. The total (₹388) is 21% higher than the subtotal shown earlier ($388 \div 320 = 1.21$). This is the pattern interviewees in the earlier modules described.

Cart (redesign)

Your cart

Veg biryani × 1	₹260
Raita × 1	₹60
Subtotal	₹320
Delivery fee (est.)	₹45
Platform fee	₹5
Taxes (est.)	₹18
Estimated total	₹388

Pay ₹388

- 1 Full breakdown visible before paying.
- 2 Labels say 'est.' where numbers can change.
- 3 Button repeats the total — no surprises.

Fee service unavailable

Your cart

Subtotal ₹320

Delivery fee **Estimate unavailable**

Couldn't load the fee. Retry?

Retry

- 1 Never invent a number when the estimate fails.
- 2 Recovery action is visible and keyboard-accessible.

After — fee shown early, with honest fallback (illustrative wireframe, not a real product screenshot). Hypothetical redesign. It follows heuristics 1 (visibility), 5 (error prevention), 9 (error recovery). Hypothesis to test: earlier fee display raises checkout completion without raising complaints (Module 7).

From your course notes (Companion ch. 17; Lecture 16)

"Show an estimated total before checkout." In scope: item cost and an estimate based on the address. Out of scope: changing the fee policy. Acceptance: **label estimates, refresh when the address changes, and show a clear fallback when the fee service fails.** Ask design about *empty, loading, error, keyboard and accessible states*. A prototype is not necessarily connected to a real backend.

5.6.1 Clickable prototype example

A clickable prototype for the redesign would link: *Cart (redesign)* → *Pay* → *Confirmation*, plus a branch *Cart (fee unavailable)* → *Retry* → *Cart (redesign)*. In Figma: duplicate frames, name them, switch to **Prototype** mode, drag a connection from the button to the next frame, set the trigger (On click) and animation (Instant/Smart animate), then present and share a view-only link. Test it with 5 participants using a task such as: "Order this meal and tell me the total you expect to pay *before* you tap pay."

5.7 Design systems, responsive design and mobile vs web

- A **design system** is a shared library of design tokens (colours, spacing, type), reusable **components**, patterns and guidelines — plus code. Benefits: consistency, speed, accessibility baked in. As a PM: *don't request one-off components without a reason; use existing ones; contribute needs via the system's process.*
- **Responsive design** adapts layouts to screen sizes (breakpoints; fluid grids; flexible images). Design mobile-first, then enhance.
- **Mobile vs web UX:**

	Mobile app	Responsive web
Interaction	Touch, gestures, thumb reach; smaller screens	Mouse/keyboard and touch
Capabilities	Camera, push notifications, biometrics	Limited by browser; no install friction
Constraints	App-store review, OS versions	Cross-browser testing, performance on slow networks
Design focus	One main task per screen, bottom navigation	Wider layouts, persistent navigation

From your course notes (Companion ch. 17; Extra topic 14)

Avoid changing a shared production library without permission. Review variations, reusable components and responsive layouts with your designer; compare decisions to acceptance criteria and record unresolved questions in the PRD.

5.8 Accessibility

Accessibility means people with different abilities can perceive, understand and use the product.

WCAG (Web Content Accessibility Guidelines, currently 2.2) is organised around four principles — **Perceivable, Operable, Understandable, Robust** — with conformance levels A, AA, AAA (AA is the common target).

From your course notes (Extra topic 14)

Inclusive design also considers varied devices, language, connectivity, literacy and circumstances. **Accessibility is an ongoing product quality concern, not a finishing decoration.** A form should use labels in addition to colour, have visible keyboard focus, describe errors in text and work with assistive technology; **a delivery estimate must not rely only on a red or green indicator.** Involve people with disabilities; use automated checks *alongside* manual and assistive-technology testing; ask qualified specialists about applicable standards and legal obligations.

Contrast ratio is $(L_1 + 0.05) \div (L_2 + 0.05)$, where L is relative luminance (lighter ÷ darker). WCAG AA requires **4.5:1** for normal text and **3:1** for large text (≥ 24 px, or ≥ 18.66 px bold) and for UI component boundaries.

Worked check — contrast of common text colours on white (computed)

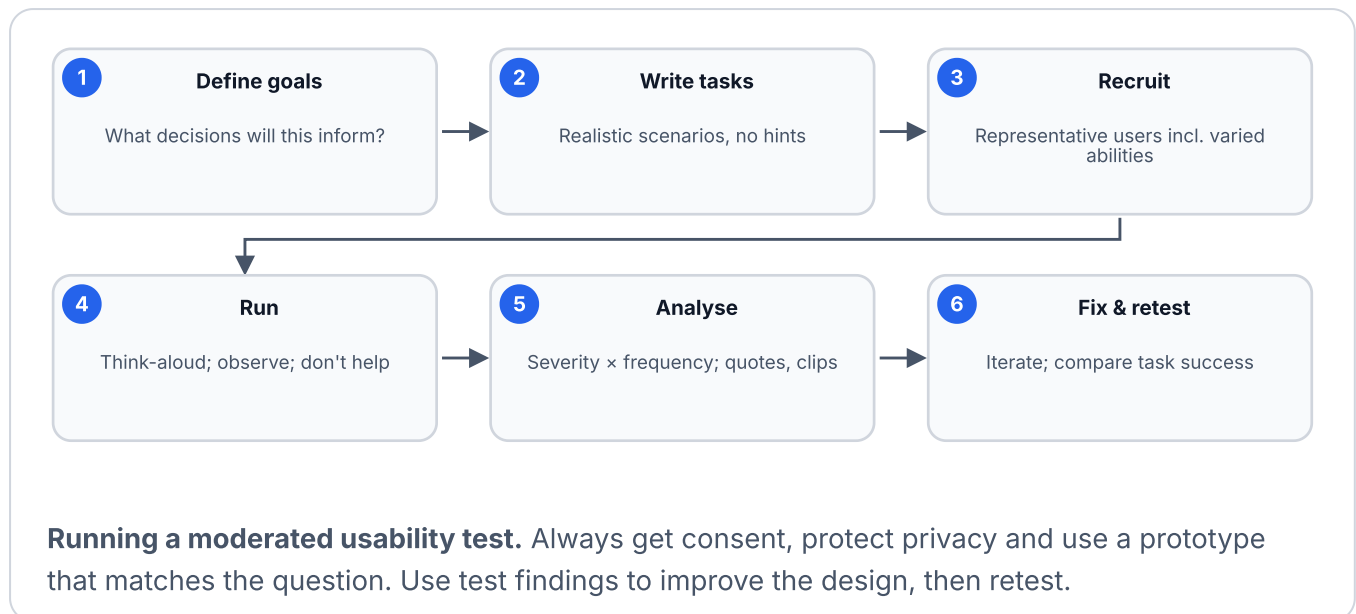
Text colour	Ratio on white	AA normal text (4.5:1)?
#101828 (navy)	17.75 : 1	Pass
#2563EB (royal blue)	5.17 : 1	Pass
#667085 (grey)	4.97 : 1	Pass
#767676	4.54 : 1	Pass (barely)
#777777	4.48 : 1	Fail
#949494	3.03 : 1	Fail (large text/UI only)
#10B981 (emerald)	2.54 : 1	Fail — white text on this green is also 2.54 : 1

This is why this handbook uses a deeper green (#047857) for filled buttons with white text: the brand emerald alone does not meet AA for text.

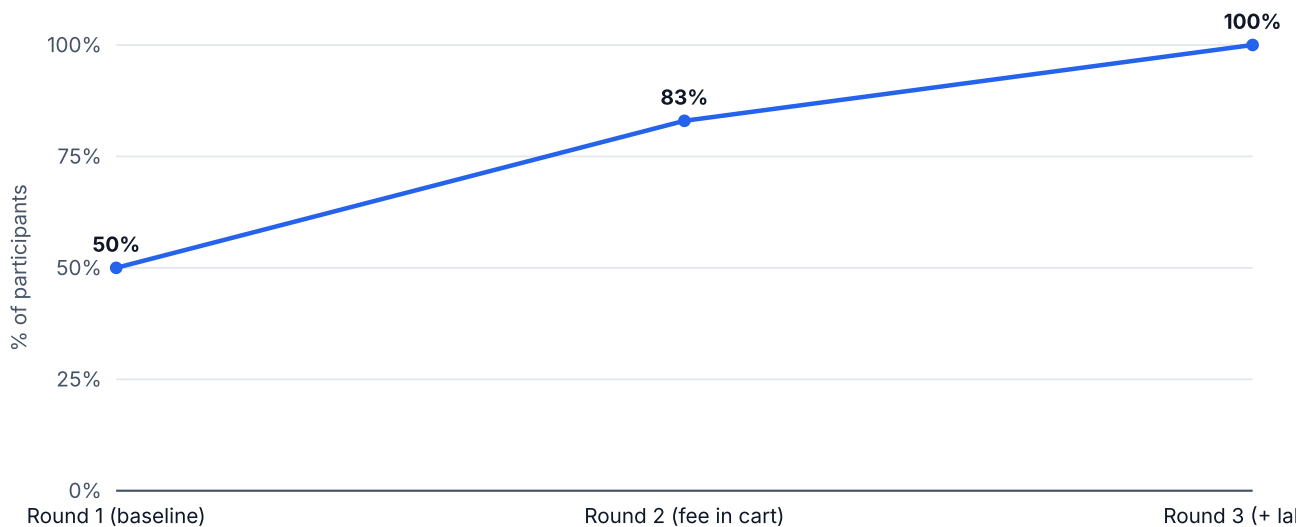
PM accessibility checklist: semantic structure and labels · keyboard operability and visible focus · contrast · text alternatives for images · captions for video · target sizes · error messages in text with guidance · no information by colour alone · respects reduced motion · tested with a screen reader on key flows · accessibility acceptance criteria in every story.

5.9 Usability testing

A **usability test** observes representative users attempting realistic tasks. Typical: **5–8 participants per round** — Nielsen and Landauer's model estimates one user finds ~31% of problems, so five users find about $1 - (1 - 0.31)^5 \approx 84\%$, which is why iterating with small rounds beats one big test (the figure is a model for finding *problems*, not for estimating *rates*).



Task wording: bad — “Click the fee breakdown.” Good — “You're about to pay for this meal. How much will you be charged in total? Tell me what you're looking at.” **Measures:** task success, time on task, errors, confidence, plus **SUS** (System Usability Scale, 10 questions; average across products $\approx$ 68 on a 0–100 scale).



Task success across design iterations (hypothetical, 6 participants per round). Hypothetical. With 6 participants each step is one or two people — treat the shape as a direction, then confirm with live behavioural data. Measure whether the **real** completion rate improves.

From your course notes (Lecture 16; Picture lesson)

Observe real people completing a task. Make a prototype, test with users with varied needs, improve and retest. The session contrasts *discovering a problem* with *testing whether people can navigate a proposed design*. A design review can find possible problems but does not replace testing with real users.

5.10 A/B testing in UX

A/B tests measure **behaviour at scale** (completion rates, clicks to goal), not understanding. Use usability tests to learn *why*, A/B tests to learn *how much*. Avoid testing tiny cosmetic differences with small traffic; test meaningful hypotheses and pair with guardrails (complaints, errors). See Module 7 for the statistics.

5.11 Design critique and working with product designers

Design critique is structured feedback on a design against goals — not personal taste. Good format: *goal and constraints* → *what's being reviewed* → *feedback framed as observations tied to user/goal* ("I notice... it may cause...") → *questions* → *decisions*. Avoid solutioneering in comments.

From your course notes (Lecture 16)

The instructor described modifying a designer's file without coordination, causing inconsistencies and confusion. **Agree on ownership, explain user and business context, use the design system and resolve disagreements through evidence.** *The claim that PMs should never edit Figma and that UI is solely a developer's responsibility is too rigid; roles depend on the team.* Designers contribute to solutions; PMs can learn and create designs when appropriate.

How to be a great partner to designers

1. Bring the **problem and evidence**, not a solution.
2. Involve them in **discovery** from day one.
3. Share constraints early (tech, legal, timing).
4. Give **specific, user-centred feedback**.
5. Protect time for exploration and for **accessibility and edge states**.
6. Respect craft: ask "what are you optimising for?".
7. Decide *together* what to test.

5.12 Figma fundamentals for PMs

From your course notes (Companion ch. 17)

Figma Design is a collaborative design tool. A **frame** represents a screen or container; text, shapes and components form the interface; a **component** is a reusable element; a **prototype** connects screens to illustrate a flow. Editing rights and features depend on the team and plan. Beginner exercise: create a personal file, add a phone-sized frame, place a heading, meal card, price and button; duplicate the frame and change it to show the total fee; in prototype mode connect the button to the next frame, preview and test it. PM workflow: use **comments** to describe the user problem and evidence ("Three test participants missed the total; could we make it visible before payment?"). Share prototype links with appropriate permissions.

Figma concept	What it is	PM relevance
Frame	Screen/container	Check screens match the flow
Auto layout	Responsive spacing rules	Explains how content grows (long names, translations)
Component & variants	Reusable element with states (default, hover, error, disabled)	Ensures states are designed
Styles / variables	Colour, type, spacing tokens	Consistency, theming
Prototype	Clickable flow	Testing and demos
Comments	Contextual feedback	Give specific, evidence-based feedback
Dev Mode	Specs and code snippets for engineers	Handoff clarity

5.13 Journey mapping, onboarding and checkout optimisation

- **Onboarding** — get users to their first meaningful value quickly: reduce steps, defer non-essential fields, use progressive profiling, show progress, explain permissions *when needed* (just-in-time), offer a sample/empty-state starter.
- **Checkout** — show the total early, minimise fields, offer guest checkout and saved payment methods, give clear error recovery, keep the primary action consistent.

Friction	UX fix	Metric to watch	Guardrail
Long sign-up form	Ask only essentials; social/phone sign-in where appropriate	Sign-up completion	Fraud/abuse rate
Permission prompt on first open	Ask in context, explain value	Permission grant rate	Uninstall rate
Hidden fees	Show total early with a breakdown	Checkout completion	Complaints
Payment failure dead end	Retry, alternative methods, saved cart	Payment success after failure	Duplicate-charge reports
Empty first screen	Sample content, guided first action	Activation within 7 days	Support contacts

Common onboarding and checkout friction → fix → metric. Every fix is a hypothesis. Test it, and watch the guardrail so you do not trade trust for conversion.

5.14 Error states and empty states

- **Error states:** say what happened, why (if useful), and how to fix it; preserve user input; offer a way forward; log for diagnosis. Avoid blame ("Invalid input!") and codes alone.
- **Empty states:** teach the product — explain the benefit, show the primary action, optionally sample data.
- **Loading states:** skeleton screens beat endless spinners; give time estimates for long tasks.

State	Weak	Strong
Payment error	"Error 502"	"We couldn't reach your bank. Your card hasn't been charged. Try again or use UPI."
Empty orders	Blank	"No orders yet. Your first order will appear here. Browse popular meals →"
Offline	Frozen app	"You're offline. Showing saved items; ordering resumes when you're back online."

5.15 UX metrics

Google's HEART framework (Happiness, Engagement, Adoption, Retention, Task success) with the **Goals–Signals–Metrics** process helps choose UX metrics.

Dimension	Goal	Signal	Metric
Happiness	Users feel confident paying	Satisfaction after order	CSAT; checkout SUS
Engagement	Users interact with the breakdown	Taps on fee details	% sessions opening breakdown
Adoption	First-time buyers complete checkout	First order placed	First-order conversion
Retention	Buyers come back	Second order in 14 days	14-day repeat rate
Task success	People pay without errors	Completed vs failed attempts	Completion rate; error rate; time on task

HEART applied to a checkout redesign (hypothetical). Pick the few dimensions that match the goal; not every project needs all five. Add guardrails (complaints, accessibility issues).

Other UX measures: **task success rate**, **time on task**, **error rate**, **SUS**, **CSAT** (satisfaction), **CES** (effort), **NPS** (recommendation — widely used, with limitations as a UX measure), **accessibility issue count**, **support contacts per 1,000 orders**.

5.16 Design handoff

A good handoff is a **conversation**, not a file drop: walk through the flow, states (empty/loading/error/offline), responsive behaviour, accessibility, copy, analytics events, and edge cases. Provide: final frames, component references, spacing/typography specs (Dev Mode), prototype link, acceptance criteria, **open questions**, and an agreed **QA review** (design reviews the build before release).

Handoff checklist: all states designed · copy final (or flagged) · accessibility requirements listed · events named · animation specs · assets exported · decisions logged · QA owner named · definition of done includes design review.

5.17 Exercises with model answers

Exercise 1 · Heuristic evaluation

Evaluate the “Before” checkout screens against the 10 heuristics. List three violations with severity.

Model answer

(1) **Visibility of system status** — the final cost isn't visible until payment (severity 3: causes abandonment). (2) **Match with the real world / consistency** — “fees are calculated at payment” hides a concept users need now (severity 2). (3) **Error prevention** — no way to preview the total before committing (severity 3). Also note heuristic 9: no designed failure state if fee calculation fails. Recommend the “After” design and test it.

Exercise 2 · Apply Fitts's law

A primary button is 48 px wide and 500 px from the thumb's resting point. What is the index of difficulty? What happens if you widen it to 96 px?

Model answer

$ID = \log_2(500/48 + 1) = \log_2(11.42) \approx 3.51 \text{ bits}$. With 96 px: $\log_2(500/96 + 1) = \log_2(6.21) \approx 2.64 \text{ bits}$ — about 25% easier. In practice also move it into the thumb zone and keep a ≥ 44 px height.

Exercise 3 · Accessibility check

Your error text is #949494 on white, 14 px. What is wrong and how do you fix it?

Model answer

Contrast is $\sim 3.03:1$; normal text needs $\geq 4.5:1$. Fix: darken to $\approx \#767676$ ($4.54:1$) or darker (e.g., $\#595959 = 7.0:1$); don't rely on colour alone — add an icon and the text “Error:”. Test with a screen reader that the error is announced and associated with the field.

Exercise 4 · Plan a test

You redesigned onboarding. Write three realistic tasks and two metrics for a usability test with 6 users.

Model answer

Tasks: (1) "You've just downloaded the app. Set it up so it can deliver to your office." (2) "Find a vegetarian meal under ₹300 and add it to the cart." (3) "Check what you'd pay in total." Metrics: task success rate, time on task, errors/hesitations, confidence rating (1–5), SUS at the end. Pre-define success (e.g., ≥ 5 of 6 complete task 3 unaided).

5.18 Interview perspective

"How would you improve the checkout experience of a food-delivery app?" — structure

1) Clarify goal (completion? AOV? trust?) and users. 2) Map the journey and find the biggest friction with data (funnel by step/device) and research. 3) Hypotheses: fees shown late, payment failure recovery, form length. 4) Solutions with trade-offs (show fee early; guest checkout; retry). 5) Prioritise (impact/effort/risk). 6) Design considerations: states, accessibility, honest fallbacks. 7) Test: usability test then A/B with guardrails. *Model answer highlights:* "I'd start with the funnel to find where users leave, then interview people who abandoned. If fees surprise them, I'd test showing a full breakdown in the cart with 'estimate' labels and a clear fallback. Primary metric: checkout completion; guardrails: complaints and payment errors; segment by new vs returning users." *Mistakes:* jumping to UI changes without evidence; ignoring failure states and accessibility.

"How do you work with designers when you disagree?" — 60-second answer

"I start by getting aligned on the user problem and the success metric. Then I ask what the designer is optimising for and share the evidence I have. If we still disagree, we agree on a quick test — a usability session or A/B — and a decision rule in advance. I give feedback as observations tied to the user, not preferences, and make sure the final decision is documented."

5.19 Key takeaways

- **UX** = whole experience; **UI** = the interface layer.

- Use **heuristics** and the **laws of UX** (Jakob, Hick, Fitts), **Gestalt** and **cognitive load** to review designs quickly.
- Design **all states**: empty, loading, error, offline, success.
- **Accessibility** is a quality requirement: check contrast, keyboard, labels, target sizes.
- **Usability tests with ~5–8 users per round** find most problems; A/B tests measure behaviour at scale.
- Partner with designers by bringing **problems and evidence**, respecting ownership and the design system.
- Make handoff a **conversation** with states, events, accessibility and QA agreed.

5.20 Check your understanding

CHECK YOUR UNDERSTANDING

Which statement correctly contrasts UX and UI?

- A. UI is the whole experience; UX is the buttons.
- B. UX is the whole experience; UI is the interface people see and use.
- C. They are the same.
- D. UX applies only to mobile apps.

Why: UX covers the full experience of achieving a goal, including usability, trust and failure handling; UI is the visual and interactive layer.

Hick's law says decision time...

- A. Decreases with more options
- B. Increases with the number and complexity of choices
- C. Is constant
- D. Depends only on colour

Why: More (and more complex) choices take longer to process; group and prioritise options.

Fitts's law implies a primary mobile button should be...

- A. Small and far from the thumb
- B. Large and within easy reach
- C. Hidden in a menu
- D. Red

Why: Time to reach a target falls as it gets closer and larger.

Normal body text needs at least what contrast ratio for WCAG AA?

- A. 2 : 1
- B. 3 : 1
- C. 4.5 : 1
- D. 21 : 1

Why: WCAG 2.2 AA requires 4.5:1 for normal text, 3:1 for large text and UI components.

Why run several small usability rounds rather than one large one?

- A. It is cheaper per participant only
- B. Each round finds most problems, you fix them, then retest to find the next layer
- C. Statistics require it
- D. Designers prefer it

Why: Small samples find most usability problems; iterating compounds the benefit.

A good error message...

- A. Shows an error code only
- B. Blames the user
- C. Explains what happened and how to recover, in plain language
- D. Is hidden

Why: Heuristic 9: help users recognise, diagnose and recover from errors.

5.21 Recommended resources

- [Nielsen Norman Group: 10 usability heuristics](#) and [heuristic evaluations](#).
- [Laws of UX](#) — concise write-ups of Jakob's, Hick's, Fitts's laws and more.
- [W3C: WCAG 2 overview](#) and [How to Meet WCAG \(Quick Reference\)](#).
- [Figma Help: Share files and prototypes](#).
- [Google HEART framework \(Rodden, Hutchinson, Fu\)](#).
- [Stanford d.school: Design Thinking Bootleg](#).