STRETCHLEARN · WORKBOOK

# Figma Prototyping & Interactions — Workbook

This workbook turns the course into one shipped high-fidelity prototype, a usability test, and a stakeholder demo. You will set the prototype's purpose and fidelity, build a variable map and a conditional-logic plan, convert flat states into multi-state interactive components, add overlays and fixed navigation, then write a test script, moderate five sessions, and assemble a findings-to-decisions deck. Work one section per module. Use the included templates to track variables, log every interaction, plan test tasks, and rank findings so the prototype reaches a usability test and a stakeholder room without guesswork.

## The Prototyping Mindset and Core Interactions

Set the prototype's purpose and fidelity, then practise the trigger-action model and lock in your motion defaults.

### Worksheet: Prototype Purpose and Fidelity Brief

Before building anything, define what your prototype is for. Fill this brief so the question sets your fidelity and you avoid over-building screens no one will test.

Prototype name

The one question this prototype must answer

Primary use (concept test / usability test / stakeholder demo / handoff reference)

Required visual fidelity (low / medium / high) and why

Required functional fidelity (low / medium / high) and why

The hero flow that must be fully real

What you will deliberately fake or skip

### Exercise: Map Triggers and Actions for One Flow

Take the hero flow from your brief and, before opening Figma, plan each interaction as a trigger-plus-action sentence so you know what to wire.

• List every interactive element in the flow (each button, field, toggle, and link).

• For each, choose the trigger (On Click, While Hovering, While Pressing, After Delay, Drag, Key).

• For each, choose the action or chain of actions (Navigate, Open or Swap or Close overlay, Change to, Set Variable, Conditional).

• Mark any element that should chain multiple actions from one trigger and write them in order.

### Checklist: Motion Defaults Sanity Check

[ ] A standard duration is chosen for small, medium, and large transitions, all within roughly 100 to 500 ms.
[ ] Elements entering or responding to a tap use Ease Out.
[ ] Elements leaving the screen use Ease In.
[ ] Elements moving from one on-screen position to another use Ease In And Out.
[ ] Spring presets (Gentle or Bouncy) are reserved for playful moments only.
[ ] Frames are duplicated rather than rebuilt so Smart Animate layer names stay identical.

## Variables: Giving a Prototype Memory

Plan and build the variables that let your prototype hold input, count and calculate, and theme itself.

### Worksheet: Variable Map

List every piece of state your prototype must remember and assign each the correct variable type. Use the Variable Map template to record names, types, and defaults before you create them in Figma.
What the prototype must remember (e.g. cart count, typed name, dark mode on)

Variable name (clear and self-explanatory, e.g. cart_count)

Type (string / number / boolean / color)

Default value

Collection it belongs to (e.g. App State / Theme)

Layers or text that read from it

Actions that change it

### Exercise: Capture Input and Echo It Back

Build a flow where the user types something and a later screen displays it, proving your prototype has memory.
• Create a string variable (for example user_name) and bind a text field so typed text writes into it.

• On a later screen, bind a text layer to that variable and confirm it shows the typed value at runtime.

• Write one place in your flow where dynamic text increases realism (greeting, search label, step indicator, confirmation).

• Note one input you must collect as a number via stepper buttons instead of a text field, and why.

### Exercise: Build a Counter and a Theme Switch

Use Set Variable arithmetic and color variables with modes to add a working counter and one-click theming to your prototype.
• Create a number variable (for example cart_count) and wire a button to set it to itself plus 1; bind a badge to display it.

• Add a minus button that subtracts 1 so you have a working quantity stepper.

• Create a Theme collection with Light and Dark modes and define semantic color variables (background, surface, text, primary).

• Bind your layers to those color variables and confirm switching modes re-themes the whole screen at once.

### Checklist: Variables Health Check

[ ] Every variable has a clear, self-explanatory name, not var1 or temp.
[ ] Each variable uses the correct type for what it must do (number for counting, boolean for conditions).
[ ] Color variables are defined by role (background, text, primary), not by raw color name.
[ ] No layer fill uses a hard-coded hex where a color variable should be bound.
[ ] Every on-screen counter, badge, and label reads from a variable rather than fixed text.
[ ] Variables are grouped into sensible collections (e.g. App State and Theme).

## Conditional Logic and Interactive Components

Add if-else branching and convert flat states into multi-state interactive components, then combine the techniques in one flow.

### Worksheet: Conditional Logic Plan

Plan each decision point in your prototype as an if-else rule before wiring it. Use the Conditional Logic Plan template so every branch has a clear condition and both outcomes are defined.
Trigger element and event (e.g. On Click of Log in)

Condition expression (using your variables, e.g. email is not empty and password is not empty)

If true: action(s) to run

Else: action(s) to run

Variables this condition reads

Variables this branch changes

How the user sees the result (navigate / error overlay / message variable)

### Exercise: Wire a Validated Login Gate

Build a login screen that branches on input using one conditional, so an empty field shows an error and a filled field proceeds.
• Create the variables you need (for example email and password as strings).

• On the Log in button, add a Conditional that checks both fields are not empty.

• In the if branch, Navigate to the dashboard; in the else branch, set an error message and open an error overlay.

• Test it by leaving a field blank and confirm the conditional blocks the path and shows the error.

### Exercise: Convert a Flat State into an Interactive Component

Take one element you previously drew as separate frames and rebuild it as a single multi-state interactive component with baked-in behaviour.

• Create a component set with a State property (for example a toggle with Off and On, or a button with Default, Hover, Pressed).

• Wire While Hovering and While Pressing for momentary states so they revert automatically.

• Wire On Click with Change to for any persistent flip (toggle on and off, tab selected), using Smart Animate around 200 ms.

• Drop two or three instances onto a frame and confirm each behaves independently with no per-instance wiring.

### Checklist: Logic and Component Review

[ ] Every decision point uses a Conditional rather than a hard-wired single path.
[ ] Conditions reference variables and use correct comparisons (equal, not equal, greater than, less than).
[ ] Each conditional defines both the if and the else outcome so no path dead-ends.
[ ] Interactive components use While Hovering and While Pressing for states that should revert.
[ ] Persistent flips use On Click with Change to and an appropriate Smart Animate duration.
[ ] The tested hero path is fully functional and any areas outside it are cleanly faked.

## Overlays, Testing, and Stakeholder Demos

Add overlays and fixed navigation, run a five-person usability test, and present findings as decisions.

### Exercise: Build a Modal, a Toast, and Pinned Navigation

Add app-like chrome to your prototype using overlays, swap overlay, and fixed-position elements so it stops feeling like a slideshow.

• Design a modal as its own frame and open it with Open overlay, centered, with the background dimmed and close-on-outside-click on.

• Build a stepped dialog using Swap overlay so a confirmation becomes a success message without the panel closing.

• Add a toast that appears on an action and auto-dismisses using After Delay after about two seconds.

• Set a header or bottom tab bar to Fix position when scrolling and confirm it stays pinned as content scrolls.

### Worksheet: Usability Test Plan

Plan a moderated test of your prototype with five participants. Use the Usability Test Plan template to write tasks as goals with observable success states and a neutral moderation script.

Task 1 written as a goal (not steps) and its observable success state

Task 2 written as a goal and its success state

Task 3 written as a goal and its success state

Participant profile (who resembles a real user)

Welcome and think-aloud instruction (what you will say to start)

Neutral prompts to use when a participant is stuck (e.g. what did you expect to happen)

Wrap-up questions to ask after the tasks

### Exercise: Moderate and Log Five Sessions

Run the test with about five participants and capture what they do, not just whether they succeed, so you collect real findings.

• For each participant, record task success or failure and where exactly they hesitated.

• Note the precise words participants used and any workarounds or misreads you observed.

• Resist helping when someone is stuck; write down what you wanted to say but did not, since that gap is the finding.

• After all sessions, list the recurring problems that appeared across multiple participants.

### Checklist: Demo and Findings Readiness

[ ] The prototype opens cleanly in presentation view and the hero path runs without fumbles after rehearsal.
[ ] The demo opens by stating the one question the prototype was built to answer.
[ ] Findings are translated into user goals and business impact, not Figma mechanics.
[ ] Findings are ranked by severity: blockers, major issues, and minor polish.
[ ] Every notable problem is paired with a concrete proposed fix.
[ ] The demo ends with a clear recommendation and the decision being requested.
[ ] A live prototype link is available for stakeholders to explore afterward.

## Your Action Plan

1. Write the purpose-and-fidelity brief and lock the one question your prototype must answer.
2. Map the hero flow as trigger-plus-action sentences and set your standard durations and easing curves.
3. Build the Variable Map, then create every variable in Figma with clear names, correct types, and defaults.
4. Add memory: capture typed input into a string variable and echo it on a later screen.
5. Add a working number counter with Set Variable arithmetic and a theme switch using color variables with Light and Dark modes.
6. Plan every decision point in the Conditional Logic Plan, then wire a validated login gate with one conditional.
7. Convert at least one flat multi-frame element into a single multi-state interactive component with baked-in behaviour.
8. Add overlays for a modal and a toast, a swap-overlay stepped dialog, and pin the navigation with fixed position.
9. Write the Usability Test Plan with tasks as goals, recruit about five participants, and run and log the sessions neutrally.
10. Assemble the findings into a severity-ranked list with a proposed fix each, then rehearse and deliver the stakeholder demo ending in a clear decision request.