STRETCHLEARN · WORKBOOK

UI Micro-interactions — Workbook

This workbook turns the course into a working micro-interaction kit and three prototyped flows. You will audit real products for the four-part model, specify motion with real durations and named easing, design the full state set for a button, choose the right loading pattern, and prototype everything in Figma with Smart Animate, variants, and interactive components. Work one section per module and finish with a documented kit plus three flows: a button with all states, a form with live validation, and a loading-to-content transition using a skeleton screen. Use the included templates to spec motion, capture every state, and run a handoff check so nothing reaches a developer as a guess.

What a Micro-interaction Is and Why It Matters

Learn to dissect any interaction into trigger, rules, feedback, and loops, and build instincts for timing and easing.

Exercise: Dissect Three Real Micro-interactions

Open three apps you use daily. Pick one micro-interaction from each (for example a like, a pull-to-refresh, a toggle) and break each one down using Dan Saffer's four-part model. Watch each interaction several times before writing.

• For interaction one, write its trigger, rules, feedback, and loops or modes as four short lines.

• Estimate the duration of the feedback motion in milliseconds and name the easing it appears to use (ease-out, spring, linear).

• Identify one interaction that gives weak or no feedback, and describe in a sentence how you would improve it.

• Note which of the four jobs each interaction does: status feedback, system status, error prevention, or delight.

Worksheet: Screen Audit Sheet

Choose one screen from a product you are designing or studying, such as a sign-in or settings page. List every interactive element and record the feedback it gives at each stage so you can spot the gaps.

Screen name and product

Interactive element (button / field / toggle / link)

Feedback on hover (describe or write none)

Feedback on press or tap (describe or write none)

Feedback on result (success / error / loading)

Gap or opportunity spotted

Priority to fix (high / medium / low)

Checklist: Timing and Easing Sanity Check

[ ] Each transition has a chosen duration between roughly 100 and 500 milliseconds.
[ ] No routine feedback motion runs longer than 500 ms without a deliberate reason.
[ ] Elements entering use ease-out so they arrive and settle.
[ ] Elements leaving use ease-in so they accelerate away.
[ ] No interface motion is left as linear unless intentionally so.
[ ] Any spring or bounce is subtle and reserved for playful elements.

Feedback States and Hover and Tap Transitions

Design the complete state set for a component and the hover, focus, and pressed transitions that connect them.

Exercise: Map Every State of One Button

Choose one button from your project, such as a primary Save button. Design or sketch all eight possible states and define what changes between them, using the State Set Worksheet template to record the details.

• Sketch the default, hover, focus, active, loading, disabled, success, and error states for your button.

• For the default-to-hover transition, write the property that changes, the amount, the duration, and the easing.

• Describe your focus ring style: thickness, colour, and offset, and confirm it has strong contrast on every background.

• Decide what happens after loading: does it show success then return to default, and how long does success stay visible?

Exercise: Design an Animated Toggle or Checkbox

Pick a switch, checkbox, or radio group and design its state-change motion so the new state is unmistakable without relying on colour alone.

• Write the trigger, rules, feedback, and loop for your control in four lines.

• Describe what moves or draws (thumb sliding, checkmark drawing in) so a colour-blind user can still read the change.

• Set the total duration and easing, and note any stagger, for example the fill before the checkmark.

• Decide whether to add a subtle scale bounce, and if so, the peak scale and whether it settles.

Worksheet: State Transition Spec

For your chosen component, record the exact motion of each transition between states so it can be prototyped and handed off without guesswork.

Component name

From state and to state

Trigger (hover / press / click / system)

Property that changes (fill / shadow / position / opacity)

Start value and end value

Duration in milliseconds

Easing (ease-out / ease-in / spring / custom bezier)

Checklist: Accessibility Pass for States

[ ] Every interactive element has a visible focus state for keyboard users.
[ ] No essential information is hidden behind hover alone.
[ ] State changes are signalled by more than colour (position, icon, or text).
[ ] The pressed state is visibly distinct from the hover state.
[ ] The disabled state is clearly non-interactive and lower contrast.
[ ] Focus order has been tabbed through and focus is always visible.

Loading States, Progress, and Skeleton Screens

Choose the right waiting experience for each situation and design empty, success, and error feedback that helps.

Exercise: Choose a Loader for Three Scenarios

List three real waits in your product (for example submitting a form, loading a feed, uploading a file). For each, decide the right loading pattern based on expected wait time and whether progress is known.

• For each wait, estimate the duration band: under 100 ms, 100 ms to 1 s, 1 to 10 s, or over 10 s.

• Decide for each whether you can estimate progress, and therefore whether to use a determinate or indeterminate loader.

• Pick the pattern for each: nothing, inline spinner, progress bar, or skeleton screen, and justify it in one line.

• For any wait over a few seconds, write the stage labels you would show, such as Uploading then Processing then Done.

Exercise: Build a Skeleton Screen for One View

Take a content-rich view such as a feed, profile, or dashboard and design its skeleton screen so it mirrors the real layout and swaps in smoothly.

• List each content element and the grey placeholder shape and size that will stand in for it.

• Confirm placeholder positions match the final layout so nothing jumps when content loads.

• Decide whether to add a shimmer sweep, and describe its direction and speed.

• Specify the swap from skeleton to content: cross-fade duration in milliseconds and easing.

Worksheet: Empty, Success, and Error State Planner

Plan the moments after a wait for one feature: the empty state, the success confirmation, and the error. Write copy and behaviour so each one helps the user move forward.

Feature or screen

First-use empty state message and primary action

No-results empty state message and suggested next step

Success feedback (toast / inline / button checkmark) and how long it shows

Error message text (plain language: cause and fix)

Where the error appears (next to field / top of form / toast)

How the error clears once the user fixes it

Checklist: Loading and Feedback Pass

[ ] No loader is shown for waits under about 100 ms.
[ ] Determinate progress is used wherever completion can be estimated.
[ ] Skeleton screens are used only for content-rich views, not single quick actions.
[ ] The skeleton-to-content swap is a fade, not a hard cut.
[ ] Error messages name the cause and the fix in plain language.
[ ] Toasts last only 3 to 5 seconds and are never used for must-read information.

Prototyping Micro-interactions in Figma

Prototype your interactions with Smart Animate, variants, and interactive components, then document them for handoff.

Exercise: Your First Smart Animate Transition

In Figma, build a two-frame Smart Animate transition for one micro-interaction, paying close attention to identical layer naming between frames.

• Create frame A with an element, duplicate it to frame B, and change one or two properties in B.

• Confirm the changed layer has the exact same name in both frames so Smart Animate can match it.

• Connect A to B with an interaction, set Smart Animate, choose Ease Out, and set the duration in milliseconds.

• Play it, then note one thing that fell back to a dissolve instead of animating and why (usually a naming mismatch).

Exercise: Build a Stateful Interactive Button

Create a button component with variants for Default, Hover, and Pressed, and wire the transitions using momentary triggers so states revert automatically.

• Add a variant property called State with values Default, Hover, and Pressed, and design each one.

• Wire Default to Hover with While Hovering, Smart Animate, around 120 ms.

• Wire Default to Pressed with While Pressing, Smart Animate, around 80 to 100 ms.

• Drop two or three instances onto a frame and confirm each reacts to hover and press on its own.

Worksheet: Lottie Versus Figma Decision Log

For each micro-interaction in your prototype, decide whether to keep it native in Figma or build it as a Lottie animation, and record the reason so the choice is deliberate.

Micro-interaction name

_____

_____

Type (hover / toggle / loader / success / empty-state animation)

_____

_____

Build in Figma or Lottie

_____

_____

Reason for the choice

_____

_____

If Lottie: source (LottieFiles / custom After Effects) and any colour or speed edits

_____

_____

Where the developer will place it (web / iOS / Android)

_____

_____

Checklist: Developer Handoff Readiness

[ ] Every micro-interaction has a written trigger, property, start and end values.
[ ] Each transition lists a duration in milliseconds and a named easing or bezier.
[ ] Any sequencing or stagger is noted with its offset.
[ ] Layer names are consistent so Smart Animate behaves as intended.
[ ] A live Figma prototype link or short screen recording is attached.
[ ] Any Lottie animations are exported as JSON and linked with colour and speed settings.

Your Action Plan

1. Audit two products you use daily and log five strong micro-interactions using the trigger, rules, feedback, loops model.
2. Lock your motion defaults: choose your standard durations for small, medium, and large transitions and your default easing curves.
3. Pick one core component, such as a primary button, and design its full eight-state set on paper or in Figma.
4. Specify every state transition with the State Transition Spec template: property, values, duration, and easing.
5. Decide the loading pattern for each real wait in your product using the duration bands and the determinate-versus-indeterminate test.
6. Design and prototype one skeleton screen for a content-rich view, including the cross-fade to real content.
7. Write empty, success, and error states for one key feature using plain, helpful copy.
8. In Figma, build the stateful interactive button with Default, Hover, and Pressed variants wired with momentary triggers.
9. Identify any interaction better suited to Lottie and source or build the JSON animation for it.
10. Assemble a one-page handoff: prototype link, the motion specs, and any Lottie files, then review it against the Handoff Readiness checklist.