STRETCHLEARN · WORKBOOK

Mobile App Design — Workbook

This workbook turns the course into a real design project. You will pick an app idea and carry it through the same steps a product designer follows: choosing platform patterns, sizing touch targets, mapping navigation and gestures, designing adaptive and accessible screens, and preparing a developer handoff. Work each section as you finish the matching course module, and use the templates to keep your platform decisions, spec values, and handoff checklist in one place.

Platform Foundations: iOS HIG and Material Design 3

Lock in your platform strategy and build the iOS-versus-Android decision habit before you draw a single screen.

Worksheet: Define Your App and Platform Strategy

Choose one app idea you will design throughout this workbook. Fill in each field so every later decision has a clear reference point.

App name and one-sentence purpose

Primary user and the single most important task they do

Target platforms (iOS only / Android only / both)

Platform strategy (fully native / adaptive shared / single cross-platform look) and why

Three to five top-level destinations the app needs

Design canvas sizes you will work on (e.g. iPhone 393 x 852 pt, Android 360 x 800 dp)

Exercise: Build Your iOS-versus-Android Pattern Table

Using the Human Interface Guidelines (developer.apple.com/design) and Material Design 3 (m3.material.io), find the correct native control for each common job on both platforms. This table becomes your reference for the whole project.

• For a confirm-a-destructive-action moment, what is the iOS pattern (action sheet / alert) and the Android pattern (Material dialog)?

• For the primary create action, where does it live on iOS (nav bar / toolbar) versus Android (often a FAB)?

• For picking from several options, which control do you use on each platform (iOS action sheet versus Material bottom sheet or menu)?

• Which system fonts and icon sets apply (iOS: SF Pro and SF Symbols; Android: Roboto and Material Symbols)?

Checklist: Foundations Readiness

[ ] I can explain why a mobile app is OS-governed, not a free-form page
[ ] I design in points (iOS) and dp (Android), never raw pixels
[ ] I keep spacing and sizes on an 8-unit grid with a 4-unit half step
[ ] I have chosen a deliberate platform strategy and written it down
[ ] I have a pattern table that names the correct native control per job on each platform
[ ] I have both the HIG and Material 3 sites bookmarked for reference

Touch, Targets, and Safe Areas

Make your screens physically usable by sizing targets, respecting safe areas, and placing actions in the thumb zone.

Exercise: Audit Touch Targets on a Real Screen

Take one screen from your app (or an app you use) and measure every interactive element against the minimums: 44 x 44 pt on iOS, 48 x 48 dp on Android, with at least 8 pt/dp between targets.

• List each tappable element and its current size; which ones fall below the 44 pt / 48 dp minimum?

• Where are two targets closer than 8 pt/dp apart, risking a mis-tap, and how will you separate them?

• For small icons (e.g. a 24 pt close button), how will you give each a full-size invisible hit area?

Worksheet: Safe-Area and System-Chrome Plan

Plan how your screen respects the regions the OS reserves. Fill in each field for your main screen.

Top inset handling (anchor nav bar to safe area; clear the notch / Dynamic Island)

Bottom inset handling (lift primary buttons above the ~34 pt home indicator, with 16 pt padding)

Which elements extend edge-to-edge (backgrounds, hero media) versus pinned to safe area

Status-bar content color (light or dark) so it stays legible over your background

Landscape / split-screen check: what shifts and what must not slip under a rounded corner

Exercise: Map the Thumb Zone

Using the thumb-zone model (easy bottom arc, harder top corners), place your app's actions where a one-handed thumb can reach the frequent ones.

• Which one or two actions are most frequent, and are they in the bottom/center easy-reach zone?

• Which controls are rare or low-risk and can safely sit in the harder-to-reach top area?

• Where could a bottom sheet bring a choice down into the thumb zone instead of a top-anchored dialog?

Checklist: Touch and Layout Readiness

[ ] Every interactive element meets the 44 pt (iOS) or 48 dp (Android) minimum hit area
[ ] Adjacent targets have at least 8 pt/dp of clear space
[ ] Small icons have full-size invisible touch areas centered on them
[ ] Interactive content sits inside the safe area; only backgrounds run to the edge
[ ] Primary buttons are lifted above the home indicator

[ ] Frequent actions are in the thumb zone; I tested the core task one-handed on a real phone

Navigation, Gestures, and Components

Structure how users move through the app, design gestures that cooperate with the OS, and use native components correctly.

Worksheet: Navigation Map

Define the navigation skeleton of your app before designing screens. Fill in each field so back behavior and structure are intentional.

Top-level structure (tab bar / bottom nav with 3 to 5 peers, listed)

Which sections use stack navigation (drill-in to detail) and what pushes onto what

Anything in a drawer, and the justification for hiding it instead of using a tab

Back behavior per screen (iOS back button + edge swipe; Android system back path to exit)

Active-state treatment for the selected tab (color + weight, not color alone)

Exercise: Plan Gestures Without Colliding With the OS

List the gestures your app will use and confirm none fight the system gestures (swipe-up home, edge-back, swipe-down notifications). Make sure every essential action also has a visible tap.

• Which custom gestures do you want (swipe-to-delete on rows, pull-to-refresh, long-press menu), and where do they live relative to the screen edges?

• For each gesture-only shortcut, what is the visible, tappable alternative for users who never discover it?

• What feedback (motion plus a light or medium haptic) confirms each gesture, and is the haptic strength proportional to the event?

Worksheet: Component and State Spec for One Screen

Pick your busiest screen and specify the native components and the required states. Fill in each field.

List/row design (row height ~44 to 56 pt/dp, primary + secondary text, leading icon, trailing affordance)

Form fields and the matching keyboard type for each (email, number pad, etc.)

Overlay choices (iOS action sheet/alert/sheet versus Android bottom sheet/dialog/menu) and when each appears

Loading state design (skeleton placeholder versus spinner)

Empty state copy and first action, plus error state message and retry

Checklist: Navigation and Components Readiness

[ ] Top-level navigation uses 3 to 5 labeled tabs, not a buried drawer
[ ] Back behavior is defined for every screen and no screen is a dead end
[ ] Custom gestures stay clear of OS edge gestures
[ ] Every essential action is reachable by a plain tap, not gesture-only
[ ] Gestures give visual feedback plus an appropriate haptic
[ ] Loading, empty, and error states are designed for each data screen

Adaptive Layout, Accessibility, and Handoff

Make the design hold across screen sizes, pass accessibility checks, and ship as a buildable spec.

Worksheet: Adaptive Layout Plan

Define how your main screen reflows across Material window size classes. Fill in each field.

Compact (<600 dp) layout (single column, bottom navigation)

Medium (600 to 840 dp) changes (navigation rail/drawer, possible two columns)

Expanded (840 dp+) changes (side navigation, list-detail two-pane)

Grid setup (columns, margins, gutters; e.g. 4 columns / 16 dp margins on phone)

Figma constraints and auto-layout rules so elements stretch, pin, or wrap correctly

Exercise: Run the Accessibility Checks

Audit your key screen against the four core checks: contrast, dynamic type, screen-reader labels, and motion. Use a contrast tool (Stark or a Figma contrast plugin) for exact ratios.

• Does normal text meet 4.5:1 contrast (3:1 for large text and meaningful icons), and where does any pair fail?

• Does the layout grow gracefully at the largest dynamic-type setting without clipping or truncating?

• What accessibility label does each icon-only button need (e.g. magnifier announces as Search), and what is the logical focus order?

• What is the reduced-motion alternative (simple fades) for any large slide or parallax?

Worksheet: Design Tokens Starter

Define the named tokens developers will build from, instead of scattering raw values. Fill in a starting value for each.

Color tokens (primary, on-primary, surface, on-surface, error)

Type tokens (display, title, body, label sizes and weights)

Spacing tokens (4, 8, 16, 24, 32 scale)

Radius tokens (small, medium, large corner radii)

Elevation/shadow or state tokens (default, pressed, disabled)

Checklist: Handoff Readiness

[ ] Layout verified at Compact, Medium, and Expanded sizes with nothing clipping
[ ] All text and icons pass WCAG contrast (4.5:1 normal, 3:1 large/icons)
[ ] Text scales with dynamic type; icon-only buttons have accessibility labels
[ ] Reusable components with variants for default, pressed, and disabled states
[ ] Color, type, spacing, and radius are defined as named tokens
[ ] Every screen state (default, loading, empty, error, success) is present
[ ] Assets exported (SVG icons; @2x/@3x and Android density buckets) and a prototype shows the flow
[ ] Redlines/specs readable in Figma Dev Mode, with interaction and haptic notes written down

Your Action Plan

1. Pick one app idea and write its purpose, primary task, target platforms, and platform strategy
2. Build your iOS-versus-Android pattern table from the HIG and Material 3 so every control choice is correct
3. Sketch the navigation map (tabs, stacks, drawer) and define back behavior for every screen
4. Design the core screens on real canvases (iPhone 393 x 852 pt and Android 360 x 800 dp) using native components
5. Size every touch target to 44 pt / 48 dp, space them 8+ pt/dp apart, and respect safe areas and the thumb zone
6. Add gestures that cooperate with the OS, each with a visible tap fallback and feedback plus haptic
7. Design the loading, empty, and error state for every data screen, not just the happy path
8. Make the layout adaptive across Compact, Medium, and Expanded window size classes
9. Run the accessibility checks (contrast, dynamic type, screen-reader labels, reduced motion) and fix failures
10. Build components and tokens, export assets at the right densities, and prepare a Dev Mode handoff with a prototype and interaction notes