STRETCHLEARN · WORKBOOK

Icon Design — Workbook

This workbook turns the Icon Design course into a finished, shippable 30-icon set. Each section maps to one course module and moves you from grid setup, through precise drawing and optical correction, to a systematized, exported, and documented icon library. Work through the exercises in order and keep all source files on a single 24px grid.

Foundations of Icon Design

Establish your set's concept, build the reusable 24px grid with keylines, and lock your core style rules.

Exercise: The Blur Test for Metaphors

Pick three icon concepts from your planned set (for example settings, search, profile). Sketch each concept three different ways. Shrink every sketch to roughly 20 pixels tall on screen and squint. Circle the version of each that stays readable when blurred, and note why it survived.

• Which metaphor was instantly recognizable at 20px and which needed a caption?

• Where did you reach for cleverness when a conventional symbol would read faster?

• What single concept is hardest to represent literally, and what is your fallback?

Worksheet: Grid and Keyline Setup Sheet

Define your grid specification before drawing anything, then build it as a reusable template in your tool. Fill in each value and confirm the template is saved.

Base canvas size (px)

Outer padding (px)

Live area size (px)

Square keyline dimension (px)

Circle keyline diameter (px)

Vertical rectangle keyline (w x h px)

Horizontal rectangle keyline (w x h px)

Tool used (Figma / Illustrator / Affinity / Sketch)

Template file name and location

Checklist: Core Style Rules Locked

[ ] Chosen a primary style family (line or filled) and written it down
[ ] Set a single stroke weight for the whole set (for example 2px)
[ ] Chosen one corner radius and one line cap and one line join
[ ] Written a banned list (for example no diagonals below 30 degrees)
[ ] Saved a reusable grid template with all four keylines
[ ] Picked a loose theme so all metaphors share a context

Drawing Precise, Pixel-Perfect Icons

Build icons from primitives, master optical correction, and balance internal spacing and detail density.

Exercise: Boolean Build Drill

Using only rectangles, circles, and rounded rectangles plus Boolean operations (union, subtract, intersect, exclude), construct five icons on your grid: plus, minus, check, close (X), and a search magnifier. Keep all strokes live and all anchor points on whole pixels. Do not draw any path freehand.

• Which Boolean operation did each icon require, and why?

• How few anchor points could you use per shape?

• Where did pixel snapping change the look versus a free placement?

Exercise: Optical Correction Calibration

Place a square, a circle, and a triangle side by side, each at a 20px bounding box. Adjust the circle and triangle until all three read as equal visual weight, then shift the triangle horizontally and vertically until it looks centered. Compare your corrected shapes to Material or Feather equivalents.

• How many pixels did the circle and triangle need to overshoot the square?

• How far did the triangle move to look optically centered?

• Where does your eye still catch an imbalance, and what fixed it?

Worksheet: Per-Icon QA Sheet

For each icon you draw, fill one row of this sheet to confirm it obeys the rules before you call it done.

Icon name

Aligned to correct keyline (yes/no)

Stroke weight matches set spec (yes/no)

Caps, joins, corner radius consistent (yes/no)

Optical correction applied (yes/no)

Minimum internal gap respected (yes/no)

Detail density matches the set (yes/no)

Readable at 16px (yes/no)

Checklist: Density and Alignment Review

[ ] Laid all current icons in one row at 24px to compare weight
[ ] Reduced any over-detailed icon (for example gear teeth) to match the set
[ ] Confirmed no internal strokes merge at 16px
[ ] Reused shared motifs (arrowhead, dot) rather than redrawing
[ ] Checked horizontal elements align across icons

Building a Cohesive Icon System

Standardize shared components, plan multi-state icons, and document everything in an icon specification.

Exercise: Shared Component Kit

Identify every part that repeats across your set and build each one once as a reusable component or symbol: arrowhead, plus and minus marks, notification dot, container rectangle, and user silhouette. Rebuild any existing icon that should use these so it pulls from the shared part.

• Which shared parts appear in three or more of your icons?

• Which icons currently use a slightly different version that should be unified?

• How much faster is your next icon once the kit exists?

Worksheet: Metaphor Dictionary

Record the single agreed symbol for each concept in your set so you never contradict yourself. Add a row for every concept you need to represent.

Concept (e.g. settings)

Chosen metaphor (e.g. gear)

Rejected alternatives (e.g. wrench, sliders)

Shared components it uses

Notes on size or orientation

Worksheet: Multi-State Pair Plan

List the icons that need both an outline (inactive) and filled (active) state, typically navigation destinations, and confirm each pair shares one silhouette.

Icon name

Needs filled state (yes/no)

Outline version done (yes/no)

Filled version shares silhouette (yes/no)

Meaning still readable without color (yes/no)

Checklist: Icon Specification Complete

[ ] Documented grid, padding, live area, and keylines
[ ] Documented stroke weight, caps, joins, and corner radius
[ ] Documented style family and when filled versus outline is used
[ ] Listed all shared components with their dimensions
[ ] Included the metaphor dictionary
[ ] Added at least one do and one do-not example
[ ] Stored the spec next to the source files and versioned it

Exporting and Managing an Icon Library

Produce clean uniform SVG, name and version the library, and assemble a complete developer handoff package.

Exercise: SVG Cleanup Pass

Export every icon to SVG, then run each one through SVGO or SVGOMG. Standardize the viewBox to 0 0 24 24, set fill or stroke to currentColor, and confirm each file has the same structure. Compare the raw and optimized file sizes.

• What was the average file-size reduction after optimization?

• Did you commit to stroke-based or fill-based SVG, and why?

• Which icons exported with stray points or extra groups that needed fixing?

Worksheet: Library Naming Registry

Create one row per icon to lock in kebab-case, meaning-based names and catch duplicates before release.

Final file name (kebab-case)

Concept it represents

State suffix if any (e.g. -fill)

Group prefix if any (e.g. arrow-)

viewBox confirmed (yes/no)

currentColor set (yes/no)

Exercise: Final Set Audit at Size

Place all 30 icons on a single board and view them at both 24px and 16px. Hunt for outliers in weight, density, alignment, and legibility, and drop a few icons into a real interface mockup to sanity-check them in context.

• Which icons broke down or turned to mush at 16px?

• Did any icon look like it belonged to a different set?

• What three fixes most improved overall consistency?

Checklist: Handoff Package Ready

[ ] Optimized SVG for every icon with consistent viewBox and currentColor
[ ] PNG exports at 1x, 2x, and 3x
[ ] Visual index sheet showing every icon with its name
[ ] Icon specification document included
[ ] Semantic version number and changelog written
[ ] Source masters and exports stored in separate, clear folders
[ ] Filled-state pairs verified to share silhouettes

Your Action Plan

1. Choose a theme and draft a list of the 30 icon concepts your set will cover
2. Build and save the reusable 24px grid template with all four keylines
3. Write your core style rules and one-page metaphor approach before drawing
4. Draw icons from primitives in batches of five, applying optical correction as you go
5. Review each batch at 24px and 16px and fix outliers before continuing
6. Build the shared component kit and rebuild earlier icons to use it
7. Produce filled-state pairs for navigation icons
8. Run every icon through SVGO, set currentColor, and standardize the viewBox
9. Apply the kebab-case naming registry and version the library with a changelog
10. Assemble and audit the final handoff package of SVG, PNG, index sheet, and spec