



Designing iOS UI without the **slop**

A practical guide to designing iOS app interfaces with AI — the mindset, the prompting techniques, the tools, and the mistakes to avoid — for anyone shipping screens with an AI in the loop.

Beginners

Never designed a screen before? This explains what "good" looks like and how to prompt for it.

Technical readers

Already shipping with AI tools? Sharpen your prompting technique and spot slop before your users do.

README · WHAT'S INSIDE

01 Foundations & mindset

02 Prompting, and negative prompting

03 Tool breakdown & MCP setup

04 Practical workflow

05 Recognizing AI slop

06 Accessibility, polish & recap



AI gets you 70% there — **taste** finishes it

AI is extremely good at producing a plausible, functional-looking screen fast. It is not good at knowing when something feels right for your brand, your users, or your platform. Treat every output as a first draft, not a final answer.

Two different workflows

AI-generated design

You prompt, the AI produces a full screen or flow, and you ship close to what came out. Fast, but generic — this is how you end up with the same purple gradient hero every other app has.

AI-assisted design

You drive the decisions — layout logic, hierarchy, brand direction — and use AI at specific steps: generating variations, styling components, critiquing a draft. You stay the author.

Aim for AI-assisted, not AI-generated, for anything that will actually ship.

DEFINE BEFORE YOU PROMPT · DESIGN LANGUAGE

ELEMENT	DEFINE
Colors	A primary, a secondary, a neutral scale, and semantic colors (success, error, warning)
Type	One or two typefaces with a defined scale (e.g., 12/14/16/20/24/32)
Spacing	A spacing unit (4px or 8px) and how it scales (4, 8, 12, 16, 24, 32...)

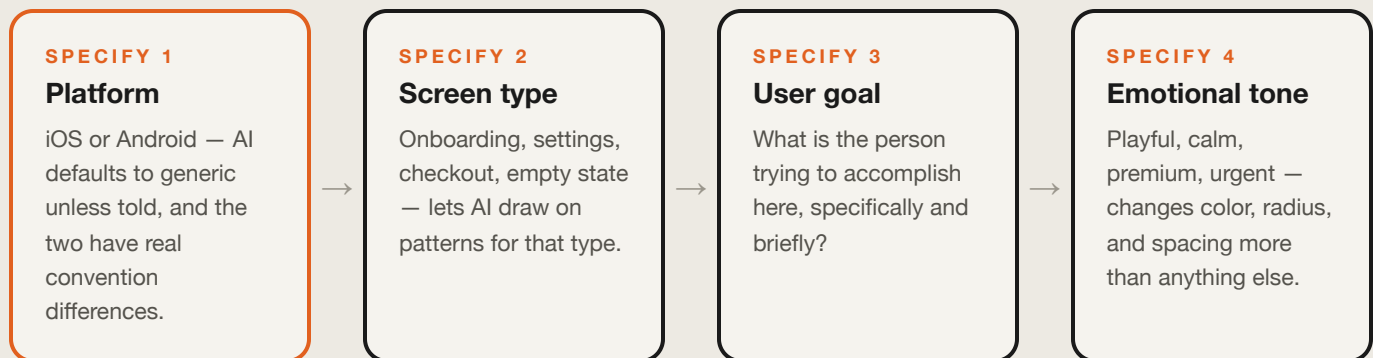
Feed this into every prompt. Without it, every AI-generated screen invents its own slightly different palette and spacing, and nothing in your app will feel like it belongs together.



02 · PROMPTING TECHNIQUES

Write prompts that **specify**

A strong UI prompt names the platform, the screen type, the user's goal, and an emotional tone. Each of those constrains the output more than any adjective pile-on.

**COPY-PASTE · STRONG PROMPT EXAMPLE**

Design an iOS settings screen for a meditation app.
The user's goal is to quickly adjust notification and reminder preferences. Tone should be calm and minimal — generous whitespace, muted colors, no visual clutter.

Notice what this prompt does *not* do: pile on adjectives. "Beautiful, modern, sleek, premium" constrains nothing — every one of those words describes ten thousand different screens.



02 · PROMPTING TECHNIQUES

Name what you don't want

Left unconstrained, a model reaches for the average of its training data. Negative prompting rules those defaults out by name — and it works better than any amount of positive description, because "not this" is unambiguous in a way "premium and modern" never is.

A starting list

- a** **No purple-to-blue gradients** — the single most recognizable AI-design tell.
- b** **No generic stock icon sets** — name a real library (SF Symbols, Lucide, Phosphor) instead.
- c** **No centered-everything layouts** — left-aligned text is the iOS default for a reason.
- d** **No lorem ipsum** — demand realistic copy, including realistic *lengths*, or the layout will break the first time real data hits it.
- e** **No glassmorphism or heavy blur** as a substitute for actual visual hierarchy.
- f** **No three-card feature grid** unless there are genuinely three things of equal weight.
- g** **No emoji as interface icons**, and no decorative sparkle/✨ motifs.
- h** **No fake data that flatters the design** — no perfectly-cropped avatars, no names that all happen to fit on one line, no empty states quietly skipped.
- i** **No invented brand elements** — no logo, no tagline, no product name you didn't ask for.

FIND YOUR OWN BAD DEFAULTS · A 10-MINUTE METHOD

The list above is generic by definition. The defaults that will actually haunt *your* app depend on your model, your prompt style, and your subject matter — so measure them instead of guessing:

- 1.** Take one screen from your app. Prompt for it three times, in three fresh sessions, with a deliberately bare prompt — no design language, no references.
- 2.** Put the three outputs side by side and mark every choice that appears in all three. Recurrence across independent generations *is* the definition of a default.
- 3.** For each one, ask: did I want that, or did the model want that? Anything in the second pile becomes a negative-prompt line.
- 4.** Keep the resulting list in your design-language file and paste it into every prompt from then on. Re-run this whenever you switch models.

You can also just ask directly: "*What visual choices did you make here that you'd make for almost any app? List them.*" Models are noticeably good at naming their own defaults when asked.



03 · TOOL BREAKDOWN

Ground every prompt in **real** screens

The single highest-leverage habit in this whole guide: before you prompt for a screen, go find two or three real apps that have solved a similar problem well, and feed those screens to your agent as reference.

TOOL	BEST FOR
Claude Design / Claude	Reasoning about UX structure, generating component code directly, critiquing existing designs
Figma Make	Fastest path from a rough idea to a visual wireframe — refine manually in Figma Design after
Mobbin MCP	Connects a library of real, live app screens into your AI workflow, grounded in shipped patterns instead of the training-data average

COPY-PASTE · TERMINAL · CONNECT MOBBIN TO CLAUDE CODE

```
claude mcp add mobbin --scope user --transport http https://api.mobbin.com/mcp
```

```
# Figma Dev Mode too, if you design there
# (the Figma desktop app must be running)
claude mcp add --transport sse figma http://127.0.0.1:3845/sse
```

```
claude mcp list # confirm both connected
```

An **MCP** (Model Context Protocol) server is a connector that lets an agent fetch from an outside service itself instead of you pasting things in. Mobbin opens a browser to authenticate on first use — no API key.

Why references matter

AI models, left to their own devices, regress to the visual mean of their training data — which is where "AI slop" comes from. A specific, real reference is the fastest way to avoid it.

Code-first approaches

Prompt Claude Code or Cursor directly for styled React Native/SwiftUI components, skipping the mockup step — works well once you have a design language and reference screens, poorly with neither.



04 · PRACTICAL WORKFLOW

From wireframe to shipped **screen**

Four steps, in order — skip straight to code with no visual direction and you get functional components with an inconsistent, undirected look.

- 1 Wireframe → AI visual direction → manual polish.** Start with rough boxes and labels, let AI propose a direction, then hand-adjust spacing and hierarchy.
- 2 Generate variations, combine the best elements.** Ask for 3–5 variations, then pick the strongest layout, color treatment, and combine them manually.
- 3 Turn mockups into real component code.** Prompt Claude Code or Cursor to convert the approved mockup into React Native/SwiftUI, referencing your design language.
- 4 A/B test with real users early.** A screen can look right and still fail in practice — watch where people hesitate or tap the wrong thing.

WHERE THE TIME ACTUALLY GOES

Steps 1 and 2 are fast — minutes each, and genuinely fun. Step 3 is where an afternoon disappears, because converting a mockup into component code is where every unstated decision has to become a real value. That's the argument for defining your design language first: it turns step 3 from a series of judgement calls into a translation job.



05 · AVOIDING AI-UI PITFALLS

Recognizing AI **slop**

Slop isn't ugly — that's what makes it dangerous. It's competent, plausible, and completely anonymous: a screen that looks like it came from no particular app, made by no particular person, for no particular user.

- a** **Purple-to-blue gradients** on every hero section or primary button, regardless of what the brand is.
- b** **Generic geometric sans-serif fonts** with no real personality — technically fine, entirely forgettable.
- c** **The same three-card, centered-icon template** regardless of content, even when there are two things or five.
- d** **Soft drop shadows and glassmorphism** used as a substitute for real hierarchy. If everything floats, nothing is emphasized.
- e** **Uniform spacing everywhere.** Real design groups related things tightly and separates unrelated things generously; slop applies the same gap to everything.
- f** **Only the happy path exists.** No empty state, no loading state, no error state, no long-name overflow — the three states you'll actually spend most of your time on.

THE THREE FIXES, IN ORDER OF LEVERAGE

1. Real references (page 5) — the highest-leverage single habit in this guide. **2. Negative prompting** (page 4) — cheap, and rules out the tells above by name. **3. A defined design language** (page 2) — the thing that keeps screen 8 looking like screen 1. None of the three is difficult; the mistake is skipping all of them because the first output looked fine.



06 · ACCESSIBILITY & POLISH

Check what AI **skips**

AI-generated mockups routinely fail basic accessibility, and it's easy to miss because the mockup looks fine on a big screen at full brightness with perfect vision. Every check below has a tool — none of this is a judgement call.

CHECK	STANDARD	HOW TO ACTUALLY CHECK IT
Contrast	WCAG AA — 4.5:1 for body text, 3:1 for large text (18pt+, or 14pt bold)	Paste the two hex values into webaim.org/resources/contrastchecker — it prints the ratio and a pass/fail for AA and AAA. Free, no signup.
Tap targets	At least 44×44pt on iOS, per the HIG	Xcode's Accessibility Inspector (Xcode → Open Developer Tool) audits a running app and flags undersized targets.
Text scaling	Layouts hold up under Dynamic Type, not just the default size	Simulator → Settings → Accessibility → Display & Text Size → Larger Text, dragged to maximum. Ten seconds, catches clipped labels immediately.
Platform conventions	Apple's HIG — nav patterns, tab bars, modals, SF Symbols	Reference the HIG explicitly in your prompt rather than assuming the AI defaults to it.

Do the contrast check on your palette *once*, when you define your design language — not per screen. Fixing a failing brand color after twenty screens use it is a much worse afternoon than fixing it on day one.

POLISH PASS · CRITIQUE YOUR OWN DESIGN

Once you have a draft, screenshot it and ask an AI a direct question: **"What's wrong with this UI?"** This flips AI from generator into reviewer — critique doesn't require inventing a direction, just spotting problems against known good patterns, which models are markedly better at. Run this before you consider any screen finished, alongside real user testing, not as a replacement for it.



RECAP

What you walk away **with**

Four things, in the order you'd put them to use.

You now have

- 1 A **design language** — colors, type, spacing — feeding every prompt.
- 2 Prompts that specify **platform, screen type, goal, and tone**, grounded in real references.
- 3 A workflow from **wireframe to tested, shipped component code**.
- 4 An eye for slop, an accessibility checklist with a tool behind every line, and an AI critique pass before shipping.

THE ONE HABIT, IF YOU ONLY KEEP ONE

Before prompting for any screen, find two or three real apps that already solved the same problem well, and hand those screens to the agent. Everything else in this guide is a refinement on top of that.

Where to go next

GUIDE	PICKS UP WITH
The AI Coding Stack	Turning these screens into a real app — the framework, the database, the MCP setup, and the AI tools that write the code
Idea to MVP in 1 Day	The compressed version of this workflow when you have hours, not weeks
Idea to App Store	The combined guide — all seven stages end to end, from idea and validation through launch and analytics