Designing Real UI with Mobbin MCP

How to stop your AI from inventing generic screens, and start building from patterns real apps already proved

README · WHO THIS GUIDE IS FOR

This guide covers the UI design workflow I use as a vibe coder: connecting Mobbin to Claude Code (or Codex), pulling real screens and flows as references, and using them to design my app and research what converts. It's part of the *AI Coding Guides* series and slots into the UI design stage, alongside *Designing UI with AI*.

Beginners

you'll learn how to connect Mobbin to your AI tool in a few minutes and run your first search.

Technical readers

you'll get prompt patterns for turning real-app research into onboarding, paywall, waitlist and landing page decisions.

01 · THE PROBLEM

1. The problem: random UI that screams "AI-generated"

If you ask an AI to design a screen with no reference, it falls back on the average of everything it has seen. That's where the purple gradients, the identical three-card layouts and the default icon sets come from.

The fix is to give the AI something real to work from. Instead of "make a nice onboarding screen", you point it at specific navigation bars, cards and entire screens from apps that already ship and convert.

02 · WHAT MOBBIN IS

2. What Mobbin is (and why the MCP matters)

Mobbin is a huge library of real mobile and web app screens, flows and website sections. You can browse it yourself at **mobbin.com/discover/apps/ios/latest** to see the latest iOS apps.

The **Mobbin MCP** lets your coding agent search that library directly, instead of you screenshotting things by hand. Once connected, your agent can:

TOOL	WHAT IT FINDS
`search_screens`	Individual UI screens: logins, paywalls, empty states, dialogs, forms, navigation
`search_flows`	Ordered multi-step flows: onboarding, checkout, account setup, subscription, cancellation
`search_sections`	Website sections: heroes, pricing, testimonials, navigation, footers
`list_collections` / `get_collection_contents`	Read your saved content and collections
`create_collection` / `save_content`	Save screens, flows or sections into collections
`add_comment`	Add a note to a saved screen, flow or section

02 · WHAT MOBBIN IS · CONTINUED

REQUIREMENTS

- A **Mobbin Pro, Team or Enterprise** plan (MCP access isn't included on the free plan).
- An MCP-capable AI tool: Claude Code, Claude Desktop/Web, Codex, Cursor and others.
- **Rate limit:** 60 requests per 60 seconds per user. If you hit a `429`, wait and retry.

03 · CONNECTING

3. Connecting the Mobbin MCP

The server endpoint is `https://api.mobbin.com/mcp` (Streamable HTTP). It's a hosted server with browser sign-in (OAuth), so there's nothing to install locally and **no API key to copy**. Pick the setup that matches your tool.

Option A: Claude Code (terminal)

Quickest method. Register the server and authenticate:

COPY-PASTE · TERMINAL

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

Then start a new Claude Code session and open the MCP menu:

COPY-PASTE · TERMINAL

```
claude
/mcp
```

Select **mobbin**, choose **Authenticate**, and sign in to your Mobbin account in the browser window that opens.

Alternative (uses your Claude account connectors). Mobbin's docs recommend this one:

1. Connect the Mobbin connector in Claude Desktop or Web (Option B below).
2. Sign in to Claude Code with the same Claude account: `claude /login`
3. Start a new session and check Mobbin appears: `/mcp`



NOTE

Connectors won't load if the CLI uses an API key, `claude setup-token`, or a provider like Bedrock or Vertex AI. Run `claude /status` to see how you're signed in.

Option B: Claude Desktop or Claude on the web

1. Go to **Customize > Connectors > Add > Browse connectors** and select **Mobbin**.
2. Click **Connect**.
3. Sign in with your Mobbin account to authorize.

On a **Team or Enterprise** Claude plan, an admin may need to add Mobbin first under **Organization settings > Connectors > Add > All available > Mobbin > Add to your team**.

Bonus: in Claude Desktop and Web, results can show as an interactive gallery.

Option C: Codex

Recommended: set up Mobbin in ChatGPT first, then sign in to Codex with the same ChatGPT account:

COPY-PASTE · TERMINAL

```
codex login
```

Codex then picks up Mobbin automatically.

CLI-only alternative:

COPY-PASTE · TERMINAL

```
codex plugin marketplace add mobbin/skills
codex plugin add mobbin@mobbin
codex mcp login mobbin
```

A browser window opens for you to sign in.

03 · CONNECTING · CONTINUED

Option D: Cursor

1. Open the Mobbin plugin page in the Cursor Marketplace and click **Add to Cursor**, then **Add Plugin**.
2. In Cursor, go to **Customize > Installed** and select **Mobbin**.
3. Click **Authenticate** and sign in to Mobbin.
4. Confirm the **mobbin** MCP server shows a status badge.

Test it

Ask your agent:

COPY-PASTE · PROMPT

Search Mobbin for onboarding screens from banking apps.

If screens come back, you're connected. If authorization fails, reconnect and complete the sign-in again. To remove access later, revoke the client in your Mobbin account settings.

04 · WORKFLOW 1

4. Workflow 1: Save references for your coding agent

This is the "browse like a designer, build like an engineer" loop.

STEP 1

Browse Mobbin

and find screens or components you like: tab bars, cards, settings lists, empty states, full screens.

→

STEP 2

Save them into a collection

for your app (e.g. "My App: Home Screen"). You can do this yourself in Mobbin, or ask the agent to use `create_collection` and `save_content`.

→

STEP 3

Point your coding agent at the collection

when you ask for UI.

Example prompt:

COPY-PASTE · PROMPT

Read my "My App: Home Screen" collection on Mobbin. Build the home screen in SwiftUI using the card layout and spacing from those references. Match the navigation bar style from the second screen. Use my design tokens from `DesignSystem.swift`.

Tips

1. Reference **specific components**, not "make it look like Airbnb". "The bottom tab bar from this screen" beats a vibe.
2. Combine references: layout from one app, card style from another.
3. Keep your own design language (colors, type, spacing) defined, so references guide structure without overriding your brand. See *Designing UI with AI*.

05 · WORKFLOW 2

5. Workflow 2: Research onboarding flows and paywalls

This is the most valuable use. Instead of guessing what converts, you study what the best apps in your category already do.

Step 1: Pull the flows

Search flows and screens in your exact category.

COPY-PASTE · PROMPT

Search Mobbin for onboarding flows from [your category] apps on iOS. Then search for paywall screens from the same apps.

Be specific: name the platform, product category, user goal or pattern. If results are too narrow, remove constraints or search screens, flows and sections separately.

Step 2: Ask for an analysis report

COPY-PASTE · PROMPT

Analyse the onboarding flows and paywalls you found. Write a report covering:

1. What the best apps do well, step by step
2. Common patterns across them (number of steps, when they ask for permissions, where the paywall sits, how they frame pricing)
3. Where they differ, and what that might mean
4. Best practices I should apply to my app, [one-line description]

Reference specific apps and screens as evidence.

Step 3: Turn the report into decisions

COPY-PASTE · PROMPT

Based on that report, propose an onboarding flow and paywall for my app. List each screen, its goal and what it should contain. Flag anything you're unsure about.

05 · WORKFLOW 2 · CONTINUED

Useful research prompts to reuse

- "Find mobile paywalls that explain quarterly and yearly billing clearly."
- "Show notification-permission priming screens and identify recurring patterns."
- "Compare how two apps in my category handle the first-run experience."
- "Find empty states used by [category] apps."

Reality check

The report tells you what other apps *do*. It doesn't tell you what *converts*. Treat the patterns as strong starting hypotheses, then A/B test with real users.

06 · WORKFLOW 3

6. Workflow 3: Research waitlists and landing pages

The same approach works before you launch. Use `search_sections` to pull website sections:

COPY-PASTE · PROMPT

Search Mobbin for hero sections, pricing sections and testimonial sections from high-converting waitlist and landing pages for [your category] apps. Identify the common structure, headline styles, social proof placement and call-to-action patterns. Then recommend a section order for my landing page.

Then hand the result to whatever builds your page. See *Framer MCP Landing Page Guide* in this series.

07 · QUICK REFERENCE

7. Quick reference

GOAL	TOOL	EXAMPLE PROMPT
Match a component style	`search_screens`	"Find bottom tab bars with 5 items from fitness apps"
Study a journey	`search_flows`	"Find onboarding flows from budgeting apps"
Compare apps	`search_flows`	"Compare how two apps handle checkout"
Landing page research	`search_sections`	"Find waitlist hero sections for mobile apps"
Keep references organised	`create_collection`, `save_content`	"Save these screens to a collection called X"

08 · COMMON MISTAKES

8. Common mistakes

1. **Copying a screen wholesale.** Use references for structure and patterns. Make it yours with your own brand, copy and assets.
2. **Vague searches.** "Nice app UI" returns noise. Name the platform, category and screen type.
3. **Skipping the analysis step.** Pulling screens isn't research. The report that spots patterns is where the value is.
4. **Ignoring rate limits.** 60 requests per minute is plenty, but a broad "search everything" prompt can burn through it. Be targeted.
5. **Forgetting accessibility.** Real apps aren't always accessible. Still check contrast, tap targets and text scaling.

SUMMARY

Summary: the workflow at a glance

STAGE	WHAT TO DO
Setup	Get a paid Mobbin plan, connect the MCP, run a test search
Components	Save real screens to a collection and point your coding agent at it
Onboarding & paywalls	Pull flows, ask for a patterns report, apply best practices
Waitlist & landing page	Pull sections, analyse structure, recommend a section order
Validate	A/B test with real users

This guide is part of the AI Coding Guides series, covering the full path from ideation to App Store launch using AI-assisted development.

SOURCES

Mobbin MCP · **Mobbin MCP reference** · **Claude Code setup** · **Codex CLI setup** · **Claude Desktop/Web setup** · **Cursor setup**