

What I hand over to AI — and what it would break without me

1
MCP

2
brief

3
interview
guide

4
synthesis

5
reviews

6
references

7
decisions
before code

8
AI feature

9
build check

10
5 questions

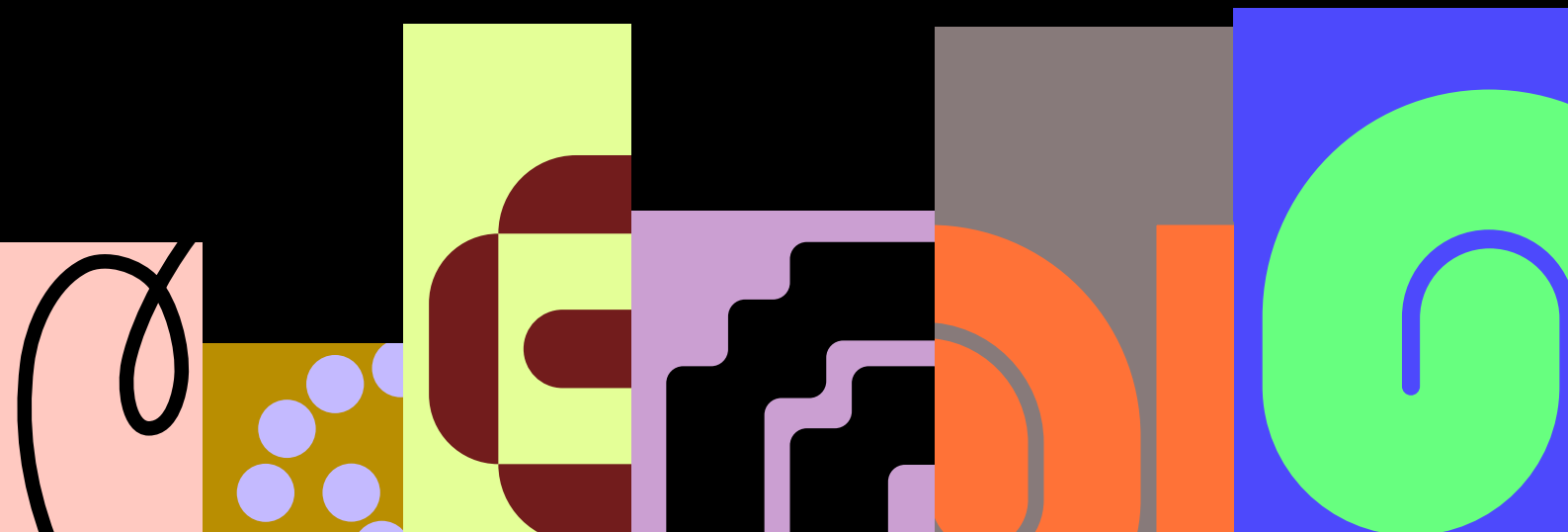
how to use

Take 1 task and 1 technique. You don't need to copy my whole toolset.

“chat” — works in a regular chat with screenshots and links. **“MCP”** — the assistant is connected to a service and can read the material directly; you still need to check the result.

[In square brackets] — what you replace with your own.

Examples come from 2 of my apps: Tunto (a planner for people with ADHD) and Ennakko (a cycle forecast without daily logging).



10 techniques from the talk

Each technique shows what I hand over to AI, where it would let me down and a prompt you can take. Add a tail to every prompt: source, date, don't guess.

1 I don't carry screenshots to the assistant – it reads the source itself MCP

prompt

Read the board [FigJam link] via Figma MCP and build a table of [N] apps: [columns]. Add a separate section on method and limits: what you saw directly, what came only from the App Store, what didn't open. Don't change the board. If there is no material, say so.

without MCP screenshots of the board in the chat, 1 app at a time – you get the same table, just slower.

My case: the Tunto competitor board – 15 apps, about 2,300 screens. 3 sections didn't render – and the table says so.

2 I don't explain the product in every chat chat

prompt

Read brief.md. State the product's main rule in 1 line – and only then answer my question.

My case: the brief template and the Tunto example – 7 lines from the product context file – are in the section "[brief.md](#)".

3 Interview guide in 1 prompt, with a reason to drop each hypothesis chat

prompt

Here are my hypotheses: [list]. For each one – 2 questions about past behavior ("when did you last...", "what did you do then...") with no "would you like" and no hint at the answer. Plus the answer that would make us drop the hypothesis. Order: from easy questions to sensitive ones.

My case: hypothesis 3 in Ennakko was "people will give 1–2 minutes a day to logging". In 6 interviews only 1 person logged every day. The conclusion isn't "nobody needs diaries" but a more precise one: don't build the whole product on mandatory daily input.

4 Synthesis by AI, cross-checks and "limitations" by me chat

prompt

Group the answers by hypothesis. For each finding – the source and N. Quotes – word for word, with the participant code. A separate section, "limitations": sample bias, intent vs behavior, what wasn't checked.

after 3 checks are mine: percentages vs the raw charts · participant codes exist · quotes can be found by searching the transcripts. Before you give the data to a service, remove identifiers; a participant code alone isn't enough for anonymity.

My case: in interviews people named €5–7 a month, but that's intent in a conversation, not a purchase. Of the 29 people surveyed, 8 had ever paid for an app like this – and they paid €25–40 a year, which is €2–3.5 a month. That goes into "limitations", not into a conclusion about price.

5 Why people leave a competitor — from reviews, not from my head chat · MCP

prompt

Group the 1-2★ reviews of [app] from the last 12 months into complaint themes. For each theme — verbatim quotes, dates and links. Mark the reasons people leave as hypotheses.

prompt · once a quarter

Find what [big players in the niche] released in the last 6 months: feature, date, source. Rumors and plans without a date go into a separate list.

without MCP give a link to the reviews page; if it isn't available, paste selected reviews with dates and sources.

My case: this is how I learned that in March 2026 Whoop launched AI tips by cycle phase and in February Oura launched its own AI model for women's health in Oura Advisor. It's a signal for positioning, not proof that they built the same product.

6 Reference → rule, not “make it look similar” chat · MCP

prompt

Compare [3] references for [screen]: take / don't take / why. Show a link to each screen; if you didn't open a screen, mark it — don't describe it from memory. Write 1 change for my screen.

before layout reference lock: 1 rule from each reference and a line “what we won't have”.

without MCP 5 screenshots from Mobbin in the chat — the same “take / don't take” table.

My case: the Ennakko payroll — 3 rules from 4 references: hero → value → plan choice; a big hero and compact plans; no one else's mascot or promises.

7 Decisions live in a dated file, not in the chat chat

prompt · spec

Write a spec for [feature]: 1) why — with a link to the research; 2) the decision with reasons; 3) what we don't do; 4) flow and states; 5) risks. First ask me about everything that is unclear.

prompt · before implementation

List every product decision this feature forces us to make, with the options and the cost of each. Don't implement anything until I answer.

My case: sync in Tunto is 1 “technical” task with 5 product decisions inside: whether it's on by default, what “deleted” means, what to do with demo data, what happens to the counters, what we promise in the copy. The status is part of the decision too: decided, implemented, checked, not checked.

8 AI feature: the fallback comes first chat

prompt

First describe what the person sees when the model is unavailable, slow or wrong. Then the happy path. Add 2 checks outside the screen: the model is available and a real call works — so you can tell “broken” from “slow”.

My case: the model worked, but the assistant chat waited only 5 seconds. The 1st generation after launch on the simulator often takes longer — the 1st tap gave an offline placeholder and nothing on the screen said whose answer it was.

9 Numbers, not eyeballing: 4 kinds of audit chat · MCP

audit 1 · full review of a live build

Walk through the live build of [the app] in the simulator along this route: [steps]. At every step – a screenshot saved to the project folder, not a temp folder. Record each finding in 5 fields (format below). In the header: this is an expert review, not a user test; what you didn't check goes into a separate list.

audit 2 · Nielsen with an "evidence" column · an easy place to start in a chat

Rate [screenshot of the screen] against Nielsen's 10 heuristics. For each one – what exactly on the screenshot supports it; severity 0-4.

audit 3 · motivation

Walk through [scenario] state by state. For each screen state – does it add motivation or take it away and why. Mark the states where the scenario loses the person.

audit 4 · accessibility and design QA (design quality check) in numbers

Compare [reference] and [implementation] across 5 surfaces: fonts, spacing, colors, assets, copy and product truth. Contrast of every text/background pair – as a number against 4.5:1. Check the largest system font size and the longest language: what gets cut off or breaks. Save the evidence to the project folder, not a temp folder.

more edge states · add to audit 1 or 4

Also go through the empty screen, an error, offline and a late response. For each state – a simulator screenshot saved to the project folder.

more UX laws · add to audit 2 or 3

Also check Jakob's, Fitts's, Hick's and Miller's laws and the peak-end rule: what on the screenshot breaks them.

why Give AI criteria instead of "it looks bad": Jakob's, Fitts's, Hick's and Miller's laws and the peak-end rule. In Tunto there is 1 main action on the screen – that's Hick's law: choice doesn't slow down the start.

finding format · for all 4

Every issue goes into 5 fields: observation and confidence · impact · fix · acceptance criterion · priority. Separate observation from assumption.

My case: the longest language at the largest system font size – 4 labels broke, 3 of them only in Polish. English hides broken line wraps.

My case with UX laws: Miller – in Ennakko the week is folded into 1 glance: energy, phase and real meetings, not 7 separate cards. Jakob's law – in Ennakko the calendar looks like a regular calendar, so there is nothing new to learn.

10 Don't take AI's answer on faith chat

prompt

Here is my prompt: [text]. What in it lets you guess or embellish? Rewrite it so the result can be checked: add the source, the date, the format and a list of don'ts.

My case: the prompt tail and 5 questions after the result are in the section "[The tail of every prompt](#)".

+ Bonus: "what should we remove?" chat

prompt

Here is a list of everything in the product: [feature list]. For each feature, answer the main rule from the brief. Suggest 3 things to remove completely – with the reason and what we lose. Don't suggest adding anything.

My case: in Tunto 2 questions like this turned 5 tabs into 3.

Prompts and templates from the talk

Tick what you already do. The prompt for each technique is above, under the same number.

-
- | | | | |
|---|--------------------------|---|-----|
| 1 | ☐ | I don't carry screenshots and links to the assistant | MCP |
| I connect libraries via MCP — the assistant goes to Mobbin, Refero and Figma itself | | | |
-
- | | | | |
|--|--------------------------|--|------|
| 2 | ☐ | I don't explain the product in every chat | chat |
| I keep a brief — every new chat reads it first | | | |
-
- | | | | |
|---|--------------------------|--|------|
| 3 | ☐ | I ask AI for an interview guide in 1 prompt | chat |
| For each hypothesis — questions about the past and the answer that makes me drop it | | | |
-
- | | | | |
|--|--------------------------|--|------|
| 4 | ☐ | I don't synthesize surveys myself — but I cross-check | chat |
| AI does the synthesis; I do the cross-checks and the "limitations" section | | | |
-
- | | | | |
|--|--------------------------|--|------------|
| 5 | ☐ | I don't guess why people leave a competitor | chat · MCP |
| "Why they leave" comes from 1–2★ reviews with dates and links;
the reasons are hypotheses | | | |
-
- | | | | |
|---|--------------------------|---|------------|
| 6 | ☐ | I don't copy references — I borrow rules | chat · MCP |
| Reference → rule: take / don't take / why; before layout — "what we won't have" | | | |
-
- | | | | |
|--|--------------------------|---|------|
| 7 | ☐ | I don't keep decisions in the chat | chat |
| Decisions go into a dated spec; a "technical" task = a list of product decisions | | | |
-
- | | | | |
|--|--------------------------|---|------|
| 8 | ☐ | I don't trust a check through the screen | chat |
| I start an AI feature with a fallback and an honest note; the proof is 2 checks outside the screen | | | |
-
- | | | | |
|--|--------------------------|-----------------------------|------------|
| 9 | ☐ | I don't check by eye | chat · MCP |
| 4 kinds of audit, a finding in 5 fields, contrast and font size in numbers | | | |
-
- | | | | |
|--|--------------------------|--|------|
| 10 | ☐ | I don't take AI's answer on faith | chat |
| The prompt tail "source · date · don't guess" and 5 questions after the result | | | |
-

The tail of every prompt

Add it to the end of any request — a report, a table, a spec, code.

1 • Source

Every number, quote or reference comes with a link, a participant code or a file path.

2 • Date

The header of the result has the date and how long the data stays valid.

3 • Don't guess

If there is no data or the tool isn't available, say so. Separate "saw" from "assume".

4 • Format and don'ts

A table, a list or a file — and what not to do: don't edit mockups, don't change numbers, don't add features.

`prompt • copy as is`

Every percentage comes with a link to the source. Every quote comes with a date. In the header — the date and how long the data stays valid. If you don't know something, write "I don't know". Separate "saw" from "assume". Answer `[as a table]; [don't touch the mockups]`.

short version — 3 rules at the end of every prompt

source. date. don't guess.

These rules make checking easier but don't guarantee a true answer. The checking is still yours.

5 questions after any answer

- 1 What did you assume rather than take from the data?
- 2 What didn't you check — didn't open, didn't run?
- 3 What could be out of date here — prices, numbers, links?
- 4 Where is the data from and how much of it is there?
- 5 Rewrite my prompt so the result can be checked.

Answers to 1–4 go into the file header as a "not checked" section. Question 5 is the cheapest way to learn to write prompts.

brief.md – a 7-line brief every chat reads first

Create brief.md in the project folder and add 1 line to CLAUDE.md, the Claude Code instructions file: “Read brief.md before you start”. After that your prompts get short – the context is already in the file.

```
template · brief.md
Product: [1 sentence – what it is and who it's for] · [constraints: platform, languages,
budget, deadline, accessibility]
User: [who, in what state they open the product, what they need to do]
Main rule: [1 question every screen and line of copy is checked against]
Tone risks: [3-5 drifts the product must not slide into]
Sources of truth: [where research, decisions, mockups and code live – and which one wins
if they conflict]
Don't: Don't make up data, don't add features without a spec, don't edit without permission.
Language and format: [document language]; numbers as digits; a date
in the header of every file.
```

example · Tunto – the product context file in 7 lines

Product	A planner for people with ADHD, built around the hardest part – getting started.
User	Mia, a freelance illustrator; she opens the app when she has already been avoiding work for 60 minutes.
Main rule	“Does this help the person start or does it give them a reason to stay in the app?”
Tone risks	Mindfulness, therapy, “cozy productivity”, replacing a planner, gamification pressure.
Sources of truth	The build and the code win over documents; documents are dated.
Don't	Don't make up data, don't add features without a spec, don't edit without permission.
Language and format	Ukrainian, numbers as digits, a date in the header of every file.

prompt · at the start of a new chat

Read brief.md. State the product's main rule in 1 line – and only then answer my question.

The line “don't add features without a spec” is there for a reason: without it AI suggests adding more often than removing.

mine · a design system in code – instead of Figma

Same as the brief: 1 file in the project and 1 rule in CLAUDE.md. Only this time the file holds tokens: colors, fonts, spacing.

prompt · design tokens

Describe the design system of [the product] in 3 layers: tokens (colors, fonts, spacing, radii) → roles → components. Move the values from [the prototype] into 1 token file with a light and a dark theme, replace hard-coded values on the screens with tokens and add to CLAUDE.md: “Colors, fonts and spacing come only from this file”. Where the contrast of text against its background is below 4.5:1, give a darker shade of the same color.

My case: in Tunto the tokens live in Theme.swift. The dot on the focus timer arc was once hard-coded to 1 color: the accent changed – and a purple dot stayed on the blue arc. With a token, all screens change together.

Decision card

Take 1 AI result — a report, a table, a spec or a screen — and fill in the 3 highlighted rows before you accept it. Use the full card — with the date and status, options, the decision and what isn't checked — for sync, a payroll, an AI feature, removing a feature or any place where “just do it” hides a product choice.

3 decisions I don't leave implicit

Benefit and the limit of the promise
What exactly the person should get — and what the product doesn't know or doesn't do.

Trade-off and cost
Which option we choose, what we lose and why that's acceptable.

Enough evidence
What has to happen to accept the result: a scenario, a build, a reopen or an honest “not checked”.

Date and status	[date] · [observed / implemented / checked / not checked]
What we promise the person	[what has to be true for the user after this decision]
Options	[option A] · [option B] · [option C], with a short cost for each
Decision	[what we choose and why]
What cost of a mistake we accept	[what the person loses if we are wrong]
What evidence shows it works	[which scenario, build, source or reopen confirms the result]
Not checked	[what is still unknown and where to see it]

prompt

Before implementing, list every product decision this feature forces us to make, with the options and the cost of each. Don't implement anything until I answer. After the decision, fill in the decision card using the template above.

The card doesn't prove that the decision is right. It makes visible the promise, the cost of a mistake and the evidence you need before you accept the result.

AI mistakes · from my projects

Where AI got it wrong — and what I require now

where it went wrong	what I require now
Ready to describe a screen it never opened (reference search)	→ a “viewed” mark + a link, or an honest “the tool isn't available”.
Cited a part of a board that doesn't exist (Tunto competitor analysis)	→ a tool checks the link, I don't take its word for it; the file has a date.
In my projects it suggested adding more often than removing	→ ask “what should we remove?”: in Tunto 5 tabs → 3.
Said “done”, but the “What's New” field in App Store Connect stayed empty	→ reread after saving, don't trust the report.

MCP: the assistant fetches the material itself

MCP (Model Context Protocol) is a standard the assistant uses to connect to tools. It isn't a separate model. Start with 1 task you repeat often and connect only what that task needs.

Mobbin

app screens

<https://api.mobbin.com/mcp>

Screens, flows and sections of mobile and web apps — collect 5 versions of 1 pattern with links. Paid subscription; you sign in through the browser on the 1st request.

Refero

web and styles

<https://api.refero.design/mcp>

Web pages and iOS screens, flows by pattern, “styles” of real sites — references for type and color. Paid subscription; you need an API key from refero.design/mcp.

ScreensDesign

flows and reviews

<https://api.screensdesign.com/v1/mcp>

Recorded app flows (onboarding → paywall), App Store reviews by country and rating, store creatives.

You sign in through the browser; there is a request limit per period — if it runs out, the assistant should say so.

Figma / FigJam

boards and mockups

<https://mcp.figma.com/mcp>

Reads boards and mockups by link and can build in the file. The Claude app has a ready-made Figma connector.

Magnific

images

<https://mcp.magnific.com>

Upscale and retouch, remove the background, extend to another format, generate an illustration or an asset.

You pay in credits for each generation.

Higgsfield

optional • video

<https://mcp.higgsfield.ai/mcp>

Video and visual concepts: App Store previews from finished screens, an animated hero, promos. Credits; you sign in through the browser on the 1st request.

Claude Code • 1 line in the terminal for each connection

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

```
claude mcp add --transport http refero https://api.refero.design/mcp + the key from your Refero account
```

```
claude mcp add --transport http screensdesign https://api.screensdesign.com/v1/mcp
```

```
claude mcp add --transport http figma https://mcp.figma.com/mcp
```

```
claude mcp add --transport http magnific https://mcp.magnific.com
```

```
claude mcp add --transport http higgsfield https://mcp.higgsfield.ai/mcp
```

For Refero, add the key: `--header "Authorization: Bearer [key]"`

Claude app

Settings → Connectors → Add custom connector → paste the address.

Cursor and similar

In mcp.json — the line `"url": "[address]"`.

check • 1 request

Find 5 onboardings with 1 question in Mobbin and give me the links.

No links is a signal to investigate, but not the only proof that something is broken. Check the connector status, sign-in, whether the tool is available and what it actually read.

works without MCP too

5 screenshots in the chat — the same table, just by hand. A connection removes the copying, not the checking: still verify the sources.

What I have installed in Claude

You don't need all of this by default — add a tool when you have a task for it.

Plugins 23

1	design-engineer volomydyr.com	Runs a design project end to end: research, spec, prototype, review, implementation from Figma and design system checks. Includes launch, discovery, review and prototype commands plus a set of agents.	Paid
2	impeccable impeccable.style	Designs and refines interfaces: audit, polish, typography, color, motion, responsive layout and accessibility. Can edit UI live in the browser.	Free
3	21st 21st.dev	Generates UI components from a prompt via 21st.dev, finds ready-made components in the catalog and publishes your own.	Free tier available
4	typesafe typesafe.ai	Skills for working with the TypeSafe System One API.	Free tier available
5	superpowers github.com	Development workflows: brainstorming before a task, plans, step-by-step debugging, TDD, code review and verification before handoff.	Free
6	figma github.com	Works with Figma: design to code, code to design, component libraries, Code Connect, FigJam, Slides, motion and shaders.	Free tier available
7	vercel github.com	Deploys and configures projects on Vercel: Next.js, environment variables, domains, caching, functions, storage and AI SDK.	Free tier available
8	frontend-design github.com	Builds frontends with a considered visual style instead of a template look.	Free
9	code-review github.com	Reviews code changes for bugs and leaves comments, including on PRs.	Free
10	code-simplifier github.com	Simplifies and cleans up recently changed code without changing behavior.	Free
11	code-modernization github.com	Helps update legacy code to modern approaches and versions.	Free
12	feature-dev github.com	Step-by-step feature development: codebase exploration, architecture, implementation and review.	Free
13	commit-commands github.com	Commands for commits, pushes and opening PRs in git.	Free
14	skill-creator github.com	Creates, edits and tests custom skills.	Free
15	claude-md-management github.com	Keeps CLAUDE.md project instruction files up to date.	Free
16	claude-code-setup github.com	Analyzes a project and recommends which automations, hooks and skills to add.	Free
17	context7 context7.com	Pulls up-to-date documentation for libraries and frameworks.	Free tier available
18	github github.com	Works with GitHub: repositories, issues and PRs.	Free

19	playwright github.com	Drives a browser via Playwright: opens pages, clicks, takes screenshots and runs tests.	Free
20	anthropic-skills github.com	Works with documents and files: PDF, Word, PowerPoint and Excel; posters, generative art, themes and Slack GIFs.	Free
21	data github.com	Data analysis: SQL queries, charts, dashboards, statistics and data validation. Connects to Amplitude, Atlassian, BigQuery, Definite and Hex.	Free tier available
22	pdf-viewer github.com	Opens PDFs and helps annotate, fill in forms and sign them.	Free
23	cowork-plugin-management github.com	Creates and customizes your own plugins.	Free

Skills 19

24	refero-design github.com	Designs interfaces grounded in real references from Refero: research first, then design.	Free
25	design-taste-frontend github.com	Builds landing pages, portfolios and redesigns that don't look like AI templates.	Free
26	high-end-visual-design github.com	Sets fonts, spacing, shadows, cards and motion at the level of a high-end agency.	Free
27	minimalist-ui github.com	Clean editorial-style interfaces: warm monochrome, bento grids, no gradients.	Free
28	industrial-brutalist-ui github.com	Interfaces in the style of Swiss typography and military terminals for dashboards and portfolios.	Free
29	redesign-existing-projects github.com	Audits an existing site or app and brings it to a premium level without breaking functionality.	Free
30	brandkit github.com	Generates brand guideline boards: logo, identity, mockups and the brand's visual world.	Free
31	imagegen-frontend-web github.com	Generates concept images of web interfaces.	Free
32	imagegen-frontend-mobile github.com	Generates concept images of mobile app screens in a phone mockup.	Free
33	higgsfield-generate github.com	Generates images, video, 3D models, sound, music and ad videos via Higgsfield.	Paid
34	higgsfield-websites github.com	Builds, edits and publishes websites and web apps via Higgsfield.	Paid
35	higgsfield-game-generation github.com	Makes browser games and game assets: sprites, textures, 3D, music and sound effects.	Paid
36	higgsfield-product-photoshoot github.com	Product photos: studio, lifestyle, banners, ad creatives and virtual try-on.	Paid
37	higgsfield-marketplace-cards github.com	Marketplace product cards: main image, infographics and A+ content.	Paid
38	higgsfield-soul-id github.com	Trains a model on face photos to generate images and video of that person.	Paid
39	higgsfield-video-explainer github.com	Makes an animated narrated video that explains a topic or tells a story.	Paid

40	screensdesign-data screensdesign.com	Mobile app research: competitors, screens, onboarding, paywalls, App Store reviews and revenue.	Paid
41	auditing-figma-slide-decks ~/claude/skills	Checks a Figma Slides deck before a talk: text, links, QR codes and styles.	Free
42	full-output-enforcement github.com	Forces complete code output with no shortcuts or placeholders.	Free

Connectors 9

43	Refero refero.design	Reference search: screens, flows and styles of real web and mobile products.	Paid
44	Mobbin mobbin.com	Searches screens, flows and sections of real apps in the Mobbin library.	Paid
45	ScreensDesign screensdesign.com	Mobile app data: screens, flows, store screenshots, reviews, downloads and revenue.	Paid
46	Magnific magnific.ai	Generates and processes images, video, 3D and audio: upscaling, background removal, relighting and stock photos.	Paid
47	Higgsfield higgsfield.ai	Access to Higgsfield generation; powers the higgsfield-* skills.	Paid
48	harness-notes github.com	Reads and resolves comments left on pages via comments-harness.	Free
49	Figma figma.com	Reads and edits Figma, FigJam and Slides files, takes screenshots and creates files.	Free tier available
50	Vercel vercel.com	Manages Vercel projects: deployments, logs, domains and environment variables.	Free tier available
51	Supabase supabase.com	Manages Supabase databases: tables, SQL, migrations, edge functions and logs.	Free tier available

Extensions 5

52	Claude in Chrome claude.com	Lets Claude control your Chrome: open pages, click and fill in forms.	Free
53	Control Chrome claude.ai	Opens, switches and closes Chrome tabs and reads their content.	Free
54	Figma Dev Mode help.figma.com	Pulls design data from the open Figma app: code, variables and screenshots.	Paid
55	Shadcn UI github.com	Provides code, demos and themes for shadcn/ui components.	Free
56	Filesystem github.com	Lets Claude in the desktop app read and write files on the computer.	Free

[olenadiomina.com ↘](#)
[LinkedIn ↘](#)
[Tunto on the App Store ↘](#)