You on AI + X

Nine recipes for building useful things with AI

Authors Edo Segal and Ikhlaq Sidhu

X is your field, craft or challenge. Add AI and the right tools. Save what works so each draft, image, dataset or program makes the next step easier.

1 → 2 → 3 → 4

Folder — Give work a home

Artifacts — Make and inspect

Tools — Extend what AI does

Next version — Reuse what worked

How the approach works

Make a project folder. Keep your brief, source files, prompts, assets, drafts and outputs together. Tell the AI which files matter; a folder does not guarantee that every file has been read.

Use an agent with tools. A harness is the working environment around an AI model. It can let the model read files, run code and inspect results. Available tools and permissions determine what it can actually do.

Connect when needed. MCP, the Model Context Protocol, is one way to connect an agent to external services or local software. A connection must be installed, available and tested before you rely on it.

Think with the outputs. An image can expose a story problem. A chart can reveal a data problem. Feed the corrected artifact back into the next task. Check facts, labels and calculations yourself.

A folder you can copy

```
my_project/
  brief.md
  sources/
  prompts.md
  notes.md
  assets/
  code/
  drafts/
  outputs/
```

In notes.md, record approved choices and the next step.

Choose a recipe

01	Explain an idea with a comic	Visual direction	p 3
02	Turn feedback into an infographic	Data and evidence	p 4
03	Create a website in a folder	Build and publish	p 5
04	Publish a world clock application	Client and server	p 6
05	Create a Story Book	Story and continuity	p 7
06	Create a short instructional film	Connected production	p 8
07	Plan a room in three dimensions	Local tool control	p 9
08	Make a volunteer welcome guide	File synthesis	p 10
09	Build a workshop dashboard	Code and testing	p 11

Visual overview p 2 | **MCP toolkit appendix** p 12

The recurring habit Make a small result, inspect it, then build on it.

YOU on AI + X

The Compounding Cognition Cookbook

HOW TO MOVE FROM CHAT TO HARNESS-DRIVEN CREATION

IDEAS
TOOLS
CREATIONS
A BRIGHTER YOU

SAME
CURIOSITY
BIGGER
POSSIBILITIES

1 Start with a Folder

Treat each project like a living workspace on your machine. Keep the prompts, notes, code, assets, drafts, and outputs together.



- prompts.md
- notes.md
- code/
- assets/
- drafts/
- outputs/

A place for ideas to grow.

2 Build a Tower of Artifacts

Each step creates artifacts that become context for the next step. As the agent sees more of the work, alignment improves and the project compounds.



More context. Better alignment. Greater results.

3 Harness ≠ Chatbot

Chatbot

- Answers questions
- Predicts likely text
- Useful for ideation

VS

Harness (Codex / Claude Code)

- Writes code
- Runs code
- Uses files and context
- Solves empirical tasks

A harness does not just talk about a task — it can act on it.

4 Expand with MCPs

Harnesses can connect to external tools and APIs. They can also use local MCPs to operate software on your own machine.



Examples:

- Video rendering
- 11Labs soundtrack
- Higgsfield video
- Local software tools

5 Think with Multimodal Models

Modern image models are not only image generators. They can research, reason, cite sources, and create robust visual artifacts such as slides, infographics, and comic pages.

Images are not just outputs — they can become inputs for the next thinking step.



6 The Big Idea

You do not need to be a developer to use these systems.
If you can explain what you want, AI can amplify your ability to build.

Core mindset: everything compounds.

Examples from practice

REAL PROJECTS. REAL LEARNING. A BRIGHTER YOU.

Book creation
Map topics to URLs, shape layouts, and wrap text around images.

Storyboard to film
Create a storyboard, generate a soundtrack, then render the video.

Visual thinking
Generate an infographic or comic page and use it as the next step of the project.

A MORE CAPABLE YOU
BUILDS A BRIGHTER TOMORROW

Build Explore Compound

Original infographic reproduced intact. Capabilities depend on the model, connected tools and permissions; check facts in generated visuals.

Explain an idea with a comic

Make a process bottleneck visible in a short comic. You learn to give visual direction and revise one part without losing the rest.

You need An image-capable AI tool, a short brief and an optional portrait you may use.

What you do

1. **Choose one message.** Write the audience, the problem and the change you want them to understand. Make clear when the story is fictional.
2. **Plan the panels.** Start with four beats: the obstacle, a new approach, a check and the result. Approve the dialogue before making the image.
3. **Generate and inspect.** Specify the visual style, panel order and exact names. Read every speech bubble and check that the sequence makes sense.
4. **Revise precisely.** Provide the current image and ask for one change, such as the face. Preserve the layout, lettering, clothes and story. Save both versions.

Sample output



From *The Dean vs. The Delay*: a seven-panel fictional satire. The two-hour build belongs to the story; it is not a general time promise.

Try this prompt

Create a four-panel fictional comic for colleagues: a delayed request, a small AI-assisted prototype, a user test and an improved result. Use short dialogue and a consistent protagonist. Show the script first; then make one comic page.

Tips and mental model

- Use a reference for appearance and another for style; say which controls what.
- Request a targeted edit: “Change the face; keep the composition and text.”

Mental model Act as the art director. Clear constraints make revisions easier.

Turn feedback into an infographic

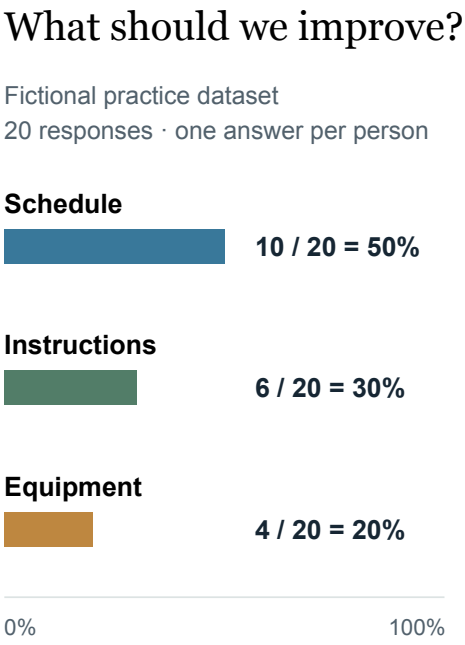
Use a small survey to explain one useful finding. This adapts the AI awareness infographic into a new topic: improving a community workshop.

You need An agent that can calculate, a CSV of anonymized responses and a chart or design tool.

What you do

1. **Define the question.** Ask what participants most want improved. Use one coded answer per person for this exercise. Keep the original responses unchanged.
2. **Check the data.** Count valid responses, missing answers and duplicates. Document how free-text answers become categories. Review ambiguous answers yourself.
3. **Calculate before designing.** Have the agent run code to count categories and compute percentages. Reconcile the counts to the total, then choose one message.
4. **Make the infographic.** Use a zero-based bar chart with direct labels. Add the question, response count and limits. Check the designed chart against the calculation file.

Sample output



Headline Half of these respondents chose scheduling as their top improvement.

A small workshop sample cannot represent every participant or explain why they answered this way.

Try this prompt

Analyze feedback.csv. Report the question, valid response count, exclusions and category counts. Run and save the calculation. Make one labeled bar chart and a plain-language takeaway. Separate observed results from explanations that the data cannot establish.

Tips and mental model

- If people can select several answers, say so; percentages may exceed 100%.
- Use code or a chart tool for exact values. Recheck numbers after visual styling.

Mental model Evidence first, visual explanation second. A polished chart still needs a valid denominator.

New example parallel to the AI awareness infographic in chat-summary.pdf, p 1. All workshop numbers above are fictional.

Create a website in a folder

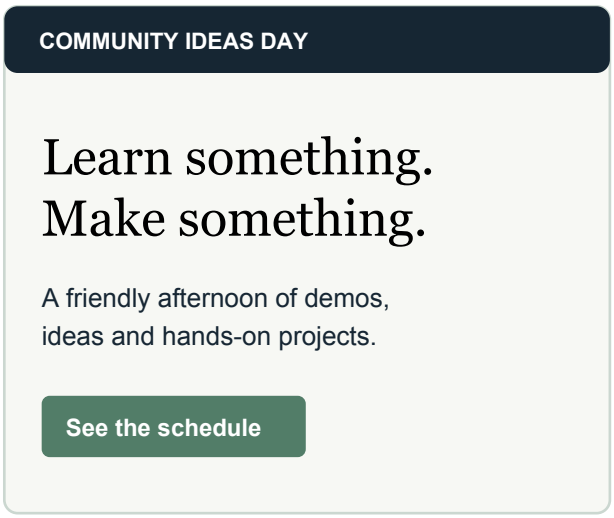
Build a simple event website on your computer, then publish the same files at a public address. You learn how a local artifact becomes a shareable website.

You need An agent with file access, a browser, your content and a hosting account.

What you do

1. **Give a small brief.** Ask for one page with an event description, schedule and a contact link. Specify plain HTML, CSS and optional JavaScript, with no build step.
2. **Make a portable folder.** Create website/ with index.html, styles.css and assets/. Use relative file paths so images and links work when the folder moves.
3. **Preview and check.** Open index.html locally. Check phone-sized layouts, keyboard navigation and every link. Ask the agent to fix what you see.
4. **Publish and revisit.** Upload the website/ folder using one route below. Open the live URL on another device. For an update, edit locally and deploy again.

Sample output



Illustrative website preview

website/

- index.html — the page
- styles.css — the appearance
- assets/ — images and files

Three ways to put the folder online

Netlify	Sign in, open Netlify Drop, then drag website/ into the drop zone.
Cloudflare	Choose Pages Direct Upload, give the project a name, upload the folder and deploy.
Vercel	Use Vercel Drop: upload the folder, choose your team and project name, then deploy.

Try this prompt

Create a one-page event website in website/ using plain HTML and CSS. Include index.html, relative asset paths and a short README. Make it responsive and check all links. Explain how to upload this folder to a static host.

Tips and mental model

• Keep private notes and credentials outside the upload folder. A contact form needs a service to receive submissions.
• **Mental model** A static website is a portable package of files; hosting gives that package a public address.

Publish a world clock application

Meridian combines a React client with a local Node server in one project folder. Learn how to move this working desktop project into a hosted web application.

You need The Meridian folder, Node and npm, a browser, and a Render or Vercel account.

What you do

1. **Run the project.** Run npm install, npm run build, then npm run serve. Open http://127.0.0.1:4318. npm start launches the Electron desktop app.
2. **Prepare a hosting copy.** Keep src/, server/, package.json and the lockfile. Adapt the client and API for one host below.
3. **Check the result.** Run npm test. Verify /api/health, city selection, daylight saving and unavailable-rate states. Show the provider date beside rates.

Try this prompt: browser app

Create a browser-based React web app backed by a Node server. Show world clocks, local currencies and exchange rates against a chosen base currency. Start with New York, London, Paris, Tokyo and Singapore. Add a plus button to search for more cities. Save city choices, handle daylight saving, and make the layout work on phones and computers.

Sample output

MERIDIAN

New York	London
09:00	14:00

1 Oct 2026 · illustrative snapshot

How the pieces fit

React client → local Node server
Clocks and city choices → data and rates

Try this prompt: Mac app

Create a React-based Electron app for macOS with a local Node server. Show a world clock and the local currency for each city, with exchange rates against a chosen base currency. Start with five global hubs: New York, London, Paris, Tokyo and Singapore. Add a plus button to search for and add more cities. Save city choices and handle daylight saving time.

Move the folder into a live application

Render — closest to the current architecture. In the hosting copy, make the server listen on 0.0.0.0 and process.env.PORT. Connect its Git repository as a Node Web Service. Build with **npm install --include=dev && npm run build**; start with **npm run serve**. The Node server serves both dist/ and the API.

Vercel — adapt the API first. Ask the agent to move the /api routes into Vercel Node functions and deploy the Vite client with dist/ as its output. Keep /api calls on the same domain. Upload the adapted project with Vercel Drop, or use Git/CLI; a raw desktop folder is not yet deployment-ready.

Try this prompt: publish the web app

Prepare a Render hosting copy of Meridian. Keep the desktop original. Set server binding, build/start commands and health check. Test client and API; write deployment instructions.

Tips and mental model

- Browser preferences stay on each device. An in-memory rate cache can reset.
- **Mental model** The client is the interface; the server is the service behind it. A host must support both.

Basis: Meridian README, package.json and server/index.js. Hosting: Render · Vercel Functions · Vercel Drop. Deployment is a recipe, not a verified live release.

Create a Story Book

Turn a spoken idea into an illustrated Story Book: a script, paired storyboard frames and production notes. Little Green reimagines Red Riding Hood in modern Manhattan.

You need An image-capable AI tool, your story brief, character references and a PDF layout tool.

What you do

1. **Start with a story change.** The original prompt changed red to emerald green and the forest to Central Park. Ask for a 90-second script with dialogue, voice-over, music and natural sound cues.
2. **Turn beats into shots.** The answer proposed eight story beats. A follow-up requested a visual PDF; the plan expanded to 12 shots and 24 first/last frames to account for cuts.
3. **Lock character continuity.** Approve Olive, the wolf and Grandma as references. Reuse the same faces, clothing and props. Ask for the first and last image of each shot separately.
4. **Assemble and review.** Make a Story Book PDF with a sequence map, paired frames, timing, dialogue and sound notes. Check the reveal, then correct inconsistent frames before exporting.

Sample output



Little Green - supplied character illustration. Emerald hood, ivory top and cream food tote establish Olive's visual identity.

One shot in the Story Book

FIRST FRAME		LAST FRAME
Olive lifts her emerald hood.	→	Olive approaches the park.

Voice-over: “Once upon a rush hour, a girl in green took dinner across the city.”
Sound: door click, footsteps, distant traffic.

Try this prompt

Let’s create a Story Book based on Little Red Riding Hood, but set in modern New York. Her route to Grandma’s home goes through Central Park instead of a forest. Give her an emerald hood and call this version Little Green. Adapt the familiar story into a 90-second script with dialogue, voice-over, music and sound effects. Plan about 12 shots, with separate first and last frames using consistent character references. Create a PDF with a sequence map, paired frames, timing and dialogue. Review continuity before exporting.

Tips and mental model

- Check story logic as well as appearance: Grandma must not appear before her reveal; the wolf must not wear his disguise in the park.
- Review phone inserts for the intended screen. Keep lettering editable and save the script, reference images and approved frames together.

Mental model The Story Book is a visual plan you can inspect and revise before making a film.

Source: supplied Little Green prompt-and-answer sequence (30 September) and character illustration. Shot targets are summarized from the sequence; this is a Story Book recipe, not a finished film.

Create a short instructional film

Explain how a visitor uses a fictional library checkout station in 30 seconds. This transfers Little Green's story beats and continuity checks to practical learning.

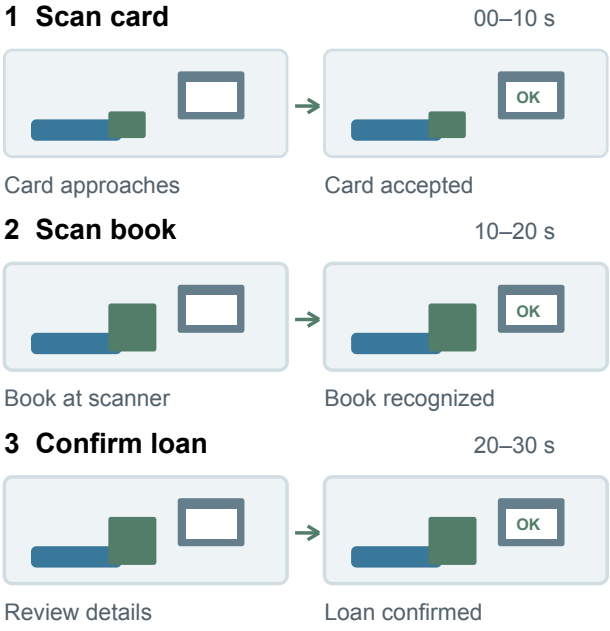
You need An approved process, image/video tools, voice or audio tools and a working video editor.

What you do

1. **Approve the sequence.** Write three ten-second shots: scan a library card, scan a book, then confirm the loan. Have someone check the actual library process before production.
2. **Lock visual continuity.** Choose one adult visitor, blue sleeve, green book and the same station. Make a first and last frame for every shot. Check all six frames together.
3. **Produce one test shot.** Generate a short clip and narration. Inspect hand movements, labels, pacing and voice clarity. Fix the test before paying for the remaining shots.
4. **Assemble and verify.** Generate the remaining clips, add captions and align audio in the editor. Watch the exported file from beginning to end and check that it teaches the approved steps.

Sample output

Three shots with a clear handoff



Six keyframes define the transitions. This is a storyboard schematic, not generated video.

Try this prompt

Using the approved checkout instructions, plan a 30-second film in three shots. Keep the visitor, book and station consistent. Draft narration and first/last-frame descriptions. Make the storyboard first. After review, produce one test shot before generating the rest.

Tips and mental model

- Keep editable scripts, reference images, audio and exports in the project folder.
- Confirm tool access and costs first. Use file exports when no connector is available.

Mental model Direct a production in stages. A good storyboard is the handoff between tools.

New example parallel to little-green-chat-summary.pdf, pp 1–3, and the conversation, p 1. Little Green had images and a plan, not a finished film.

Plan a room in three dimensions

Use a connected local design tool to compare workshop layouts. You learn to turn a precise brief into inspectable actions in software.

You need Blender, an installed and tested compatible MCP bridge, and room measurements.

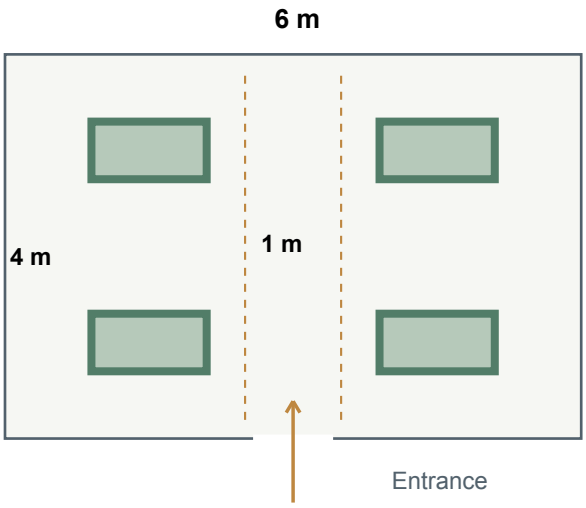
What you do

1. **Write a measurable brief.** For practice, use a 6 m by 4 m room and four 1.2 m by 0.6 m tables. Mark the entrance and a central aisle in a simple sketch.
2. **Verify the connection.** Keep Blender open. Ask the agent to list the scene objects before changing anything. Save a copy of an existing scene first.
3. **Build a rough scene.** Create the floor and tables at the specified scale. Name objects clearly. Request a top view and one perspective view before adding decorative detail.
4. **Compare and save.** Check dimensions and door access. Change one layout feature, compare both views and save the chosen .blend file plus labeled preview images.

Sample output

A rough plan before rendering

Concept sketch · dimensions in meters



Four tables, each 1.2 × 0.6 m.
Inspect scale and clearance before requesting a polished render.

Practice geometry only; room access requirements need a separate check for the real venue.

Try this prompt

First query the open Blender scene and confirm that the bridge responds. In a new scene, model the attached room sketch in meters with named objects. Save a top view and a perspective preview for review. Keep the geometry simple until I approve the layout.

Tips and mental model

- An installed app and a working tool connection are separate checks.
- When a tool call fails, inspect the app and connection before repeating it.

Mental model The AI is operating a workbench. Verify the tool, then inspect what it made.

Make a volunteer welcome guide

Turn a folder of scattered notes into a guide a new volunteer can use. This extends the source discussion of building a book from reusable project files.

You need An agent with file access, an event brief, a role list and approved instructions.

What you do

1. **Gather the sources.** Put the event brief, schedule and role descriptions in sources/. In brief.md, name the reader and the five questions the guide must answer.
2. **Map facts to files.** Ask for a source list and an outline. Have the AI flag conflicting times, missing contacts and unclear responsibilities before drafting.
3. **Make one useful page.** Draft arrival instructions, a role checklist and where to get help. Keep a source filename beside each operational detail.
4. **Test and reuse.** Ask a new reader to follow the page. Correct gaps, then save the approved guide, source map and notes for the next event.

Sample output

VOLUNTEER WELCOME

Your first 15 minutes

Illustrative content for a fictional event

1. **Arrive**
   Collect your badge.
2. **Get oriented**
   Meet your shift lead.
3. **Begin**
   Check your supplies.

Source map
Arrival → event_brief.md
Responsibilities → roles.md

Keep the source map with the editable guide so the next revision has a clear starting point.

Try this prompt

Read all files in sources/ and list what you used. Create a one-page welcome guide for first-time volunteers. Trace each instruction to its source. Flag contradictions and missing facts; do not invent them. Save the editable guide and a source map.

Tips and mental model

- Give exact file paths and say which source takes priority when versions disagree.
- Keep approved decisions in notes.md; ask the agent to read it when resuming.

Mental model The folder is shared working memory that you actively maintain.

Build a workshop dashboard

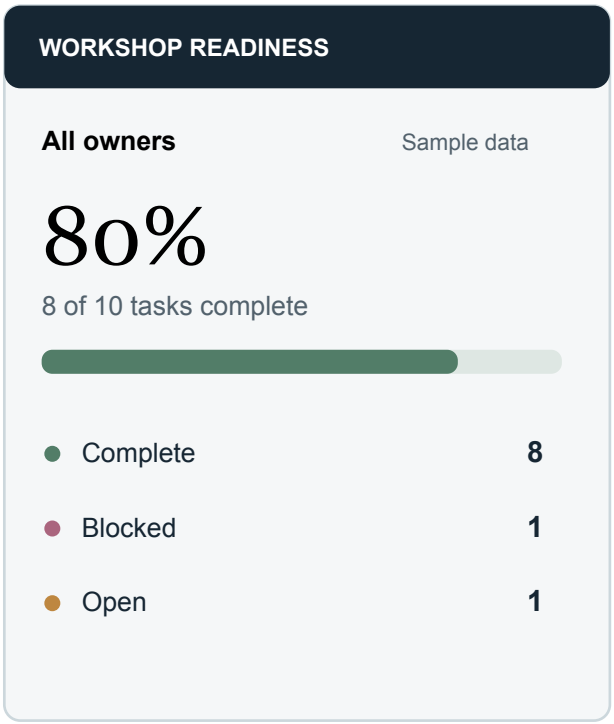
Turn a task list into a small working app. This takes the saved preferences, data states and testing lessons from Meridian into event preparation.

You need An agent with code execution, a browser and a CSV with task, owner, due date and status.

What you do

1. **Specify one screen.** Show completed tasks, blocked tasks and an owner filter. Define the allowed statuses and what counts as complete before asking for code.
2. **Build with sample data.** Use ten fictional tasks first. Ask for a local app and clear launch instructions. Add file import only after the first screen works.
3. **Check behavior.** Test a normal file, a missing column, an empty file and an unknown status. Confirm the counts after filtering and after reloading.
4. **Keep it maintainable.** Save the code, sample data and test results. Add a short README explaining how to start the app and where it stores any saved settings.

Sample output



Expected test: an empty file shows “No tasks yet” instead of a misleading 0% completion score.

Try this prompt

Build a local workshop dashboard from tasks.csv. Use complete, blocked and open as the allowed statuses. Show completion as complete divided by all displayed tasks. Add an owner filter and clear empty/error states. Run behavior tests and provide launch instructions.

Tips and mental model

- Build a tiny working version before adding live services or more screens.
- Ask for observable test results. “The code looks right” does not verify the app.

Mental model Manage a small engineering project: define behavior, run it, inspect it and improve it.

New example parallel to Meridian in chat-summary.pdf, p 1. Dashboard illustration uses fictional data; it is not a running app.

Ten useful MCPs to know

A practical shortlist of established integrations and useful specialist tools. Selected for relevance to these recipes and clear documentation, rather than a universal usage ranking.

Your brief → **AI agent** → **MCP connection** → **App or files**

Choose the capability you need, connect the tool, then inspect what it does.

1 Filesystem

Local · MCP reference server

Read, organize and edit project files.

"List the source files for my guide."

Documentation and setup

2 Notion

Remote · official

Find knowledge and update pages.

"Turn approved notes into a project wiki."

Documentation and setup

3 Slack

Remote · official

Search conversations and gather context.

"Summarize decisions in this channel."

Documentation and setup

4 Linear

Remote · official

Find, create and update project issues.

"Turn this plan into a set of tasks."

Documentation and setup

5 GitHub

Remote or local · official

Work with code, issues and pull requests.

"Explain the world-clock repository."

Documentation and setup

6 Context7

Remote or local · Upstash

Retrieve library documentation for code.

"Find docs for the version we use."

Documentation and setup

7 Playwright

Local · Microsoft

Operate a browser and inspect pages.

"Check every link on my local website."

Documentation and setup

8 Sentry

Remote · official

Inspect application errors and context.

"Investigate the latest API failure."

Documentation and setup

9 Figma

Remote or desktop · official

Bring design context into a build.

"Match this page to the selected design."

Documentation and setup

10 MCP for Blender

Local · community

Inspect and manipulate a 3D scene.

"Make a rough room from this sketch."

Documentation and setup

Start with one connection

Use the documentation links above to check client compatibility, account access and setup. Try one read-only task first. Limit local file access to the project you need, and review actions that publish, send or delete.

Sources: provider documentation and maintainer repositories linked above; checked 1 October 2026. MCP is a connection standard, not an AI model.