

PRACTITIONER · 6 MINUTE READ

Building small apps with AI, without coding

Describe a small tool in plain words, let AI build it, test with made-up data, and hand anything risky or holding real data to IT.

What it is

Some AI assistants, such as Claude, ChatGPT, Microsoft Copilot and Gemini, can build small working tools from a plain description. You type "make me a tracker for team holiday requests" and the assistant writes the code and shows you a working page: a form, a table, a calculator or a simple dashboard. Depending on the assistant, this happens in a side panel, a shared canvas, a preview window or a code-writing assistant.

You do not need to read the code: you try what it builds and ask for changes in plain words.

Think of a fast carpenter. Say "a shelf" and you get a shelf. Say "a shelf 80 centimetres wide for heavy files" and you get the one you need.

Why it matters

- Small daily jobs (a calculator, a checklist, a tracker) can sometimes get a working first version the same day, instead of waiting in a queue.
- You can show a working example instead of describing an idea in a meeting.
- Where the data is stored, and who can see it, is easy to overlook.
- A tool that looks finished can still give wrong results, so testing is essential.
- Some tools must be built and maintained properly by IT or engineers.

How to do it

1. **Write** a short spec first: who uses it, what goes in (inputs), what comes out (outputs), and the rules it must follow (see guide 15).
2. **Ask** the assistant to build the simplest version that meets the spec.
3. **Test** it with made-up data, including awkward cases: blanks, very large numbers, wrong dates.
4. **Iterate** by asking for one change at a time, and retest after each change.
5. **Find out** where the data is stored (only in your browser, in your account, or shared online) and who can open the tool.
6. **Save** a copy of each working version with a date, so you can go back if a change breaks something.
7. **Hand over** to IT or engineers before the tool touches customer data, connects to live business systems, handles payments, or needs security controls.

Example

Your team works out mileage claims by hand each month.

The spec:

Build a simple mileage calculator page.

Who uses it: staff submitting monthly travel claims.

Inputs: date, start location, end location, kilometres, purpose.

Rules:

- Rate is [amount] per kilometre.
- Claims over [number] kilometres in one trip need a note.
- Do not allow future dates.

Outputs: a table of trips, a monthly total, and a button to copy the table as text for the expenses form.

No names or employee numbers. Store nothing online.

Explain in two sentences where any data is kept.

The first version works but allows future dates. You say "block future dates and show a clear message", and the assistant fixes it. You test five made-up trips and check the total against a calculator by hand.

Your manager reviews it before the team uses it. The tool helps fill in the official form; it does not replace the approval process.

Common mistakes

- **Starting without a spec.** Fix: spend five minutes writing who, inputs, outputs and rules.
- **Testing with real data.** Fix: use made-up data until the tool is approved for real use.
- **Asking for many changes at once.** Fix: one change per message, then retest.
- **Not knowing where data goes.** Fix: ask the assistant, and check your organisation's rules for the tool you used.
- **Letting a quick tool become a business system.** Fix: when people depend on it daily, or it holds customer or personal data, hand it to IT.

Try it this week

1. Pick a small calculation or list you do by hand at least weekly (2 minutes).
2. Write a spec: who, inputs, outputs, rules (5 minutes).
3. Ask an approved assistant to build it, then test it with three made-up entries, including one awkward case (6 minutes).
4. Ask for one improvement, retest, and save a copy of the working version (2 minutes).

Going further

- Ask the assistant to write a short "how to use" note and a list of known limits to keep alongside the tool.
- Keep the spec with the tool and update it with each change, so anyone can rebuild or review it.
- Small tools can sometimes be shared with colleagues; check what the sharing setting allows, and whether viewers can see each other's entries.
- A one-click action inside your normal software may suit better (see guide 12); first ask whether to build at all (see guide 16).

Key words

- **App:** a small software tool, such as a form, tracker or calculator.
- **Spec:** a short written description of what a tool must do (short for specification).
- **Input:** information the user puts into the tool.
- **Output:** the result the tool gives back.
- **Iterate:** improve something step by step through small changes.
- **Version:** a saved copy of the tool at a point in time.