

PRACTITIONER · 6 MINUTE READ

Specs: write it down before AI builds it

A one-page spec says what you want, the rules and what "done" means, so AI builds the right thing and you can check it.

What it is

A **spec** (short for specification) is a short description of what you want, written before AI builds it. It is often one page in Markdown (see guide 06). It covers six things:

- **Purpose and users:** what the tool or task is for, and who uses it.
- **Inputs and outputs:** what goes in, and what comes out.
- **Rules:** what it must always or never do.
- **Tricky cases:** missing data, unusual values, failures.
- **Out of scope:** what it should not do.
- **Acceptance criteria:** the checks that must pass before you accept the work, in other words what "done" means.

Think of the drawing you give a builder. Without it, the builder fills the gaps with their own ideas, and you find out when the wall is in the wrong place.

Why it matters

- **AI builds what you describe and guesses the rest.** Claude, ChatGPT, Microsoft Copilot and Gemini all fill gaps with guesses. A spec turns guesses into your decisions.
- **One shared reference.** The same page guides building, review and the final check.
- **Acceptance criteria become tests.** Each criterion becomes a test case you run after every change (see guide 27).
- **It is the brief for agents.** When AI works through a task on its own, the spec tells it what to aim for and when to stop (see guide 18).
- **Someone else can pick it up.** A colleague, or you in six months, can rebuild or fix the tool from it. Writing the spec first is a recognised way of working with AI coding tools, called spec-driven development.

How to do it

1. **Write** the spec before you ask for anything to be built, using the six headings above.
2. **Ask** the AI to question the spec before building: "List any gaps, unclear rules or tricky cases I have missed. Do not build anything yet."
3. **Decide** each gap yourself and update the spec. Do not let the AI choose quietly.
4. **Ask** the AI to build from the spec, then test the result against every acceptance criterion with made-up data.
5. **Update** the spec whenever a rule or requirement changes, before you ask for the change.
6. **Save** each version with a date, so you can see what changed and when.

Example

Your team approves small expense claims by email. You want a small tool to sort them first (see guide 14).

```
# Spec: expense approval helper (version [date])

## Purpose
Sort small expense claims so managers can approve them faster.

## Users
Team members submit claims. [Manager role] approves them.

## Inputs
Date, amount, category, short description, receipt (yes/no).

## Outputs
Each claim marked "ready to approve" or "needs attention", with a reason.

## Rules
- Claims over [amount] always need attention.
- Claims without a receipt always need attention.
- Categories allowed: [travel, meals, supplies].

## Tricky cases
- Missing amount or date, zero or negative amount, future date:
  "needs attention", saying why.
- File cannot be read: show a clear error, change nothing.

## Out of scope
No payments, no approving or rejecting (a manager decides),
no names, no connection to finance systems.

## Acceptance criteria
- A claim of [amount + 1] with a receipt shows "needs attention".
- A claim with a blank amount does not crash the tool.
- Ten made-up claims are sorted as a manager sorts them by hand.
```

Before building, the AI asks about claims in other currencies. You add a rule, then ask it to build. The manager checks the ten test claims before the team uses it.

Common mistakes

- **Too vague.** Fix: fill in all six headings, even briefly.
- **Too long.** Fix: one page; put background in a separate context file.
- **No acceptance criteria.** Fix: write at least three checks you can run with made-up data.
- **Not updating it.** Fix: change the spec first, then the tool.

Try it this week

1. Pick one small tool or recurring task you hand to AI (2 minutes).
2. Write a one-page spec with the six headings (8 minutes).
3. Ask an approved assistant to list gaps and questions, without building (3 minutes).
4. Answer the questions in the spec and save it with today's date (2 minutes).

Going further

- For agent work, put the spec in the agent's instructions file and ask it to report against each acceptance criterion when it finishes.
- Keep the spec with the tool, and split larger work into several small specs, each with its own acceptance criteria.

Key words

- **Spec:** a short written description of what a tool or task must do (short for specification).
- **Acceptance criteria:** the checks that must pass before you accept the work.
- **Tricky case:** an unusual situation, such as missing data, the tool must handle.
- **Spec-driven development:** a way of working where you write the spec first, then build and check against it.