

The Plan Grader

Your AI will never tell you it is wrong. A second copy of it will. One prompt and one skill turn that into a habit, so nothing that matters ships on the word of the thing that wrote it.

Built by [@brycenwood.ai](#) · October 2026

THE PROBLEM

The AI that wrote the plan cannot grade the plan

I found this one by accident. I had spent a full day building a plan with Claude Code. Every check on it was green and I was about to run it across all three of my businesses. Then I pasted the plan into the wrong terminal window. That second window had never seen the plan. I asked it to grade the work, and it came back with sixteen problems the first one had missed.

Number sixteen was the one that scared me. The first window had told me my work was being saved. It was not. Five days of everything I had built would have quietly disappeared, and nothing would have told me.

Same model in both windows. Same brain. The difference was memory.

THE KEY IDEA

Same brain, no memory, so no trust

When the AI that built a plan checks that plan, it is checking connections it made itself. They feel right from the inside, because it knows what it meant. It remembers why each piece is there, so it never has to prove it.

The second window has none of that. It cannot lean on *I know why this is here*. It has to actually read what is on the page and test whether it holds up. That is the whole trick, and it is the same reason a good editor is never the person who wrote the draft.

THIS IS NOT ABOUT A SMARTER AI

You do not need a second tool, a second subscription, or a better model. You need the same model with no memory of writing the thing. A fresh chat gives you that for free.

STEP 1

The manual version, works in any AI today

Open a brand new chat. Not a reply in the old one, a fresh window with no history. Paste the plan in cold, the whole thing, and say this:

PASTE THIS INTO A FRESH CHAT, AFTER THE PLAN

Find everything wrong with this. Assume whoever wrote it is trying to fool you.

Read the plan itself. Do not accept any description of it.

Then:

- VERIFY every number it quotes. If you cannot verify one, say so.
- Check whether each step's PREMISE actually holds. A step can be well written and still rest on something untrue.
- Check whether step N genuinely unblocks step N+1, or only appears to.
- Say what the plan is NOT doing, and whether that is deliberate.
- Could I start step one right now without asking a question?

Report every real problem with evidence, then anything genuinely strong.

"This section is weak" is not a finding. "Step 5 assumes the report runs daily and nothing schedules it" is a finding.

That works in ChatGPT, Claude, Gemini, anything. The rule that makes it work: **give it the actual plan, never your summary of the plan**. The moment you summarise,

you have smuggled your own framing back in, and the grader inherits your blind spots.

STEP 2

Turn it into a skill in Claude Code

If you use Claude Code, you can make this a one-word command so you never have to copy and paste between windows again. Open Claude Code in your project folder and paste this in. It will build the skill and explain what it did.

PASTE THIS INTO CLAUDE CODE

I want a skill called grade-plan. When I run it on a plan file, it must grade the plan adversarially in a FRESH context, because whoever wrote a plan cannot reliably grade it.

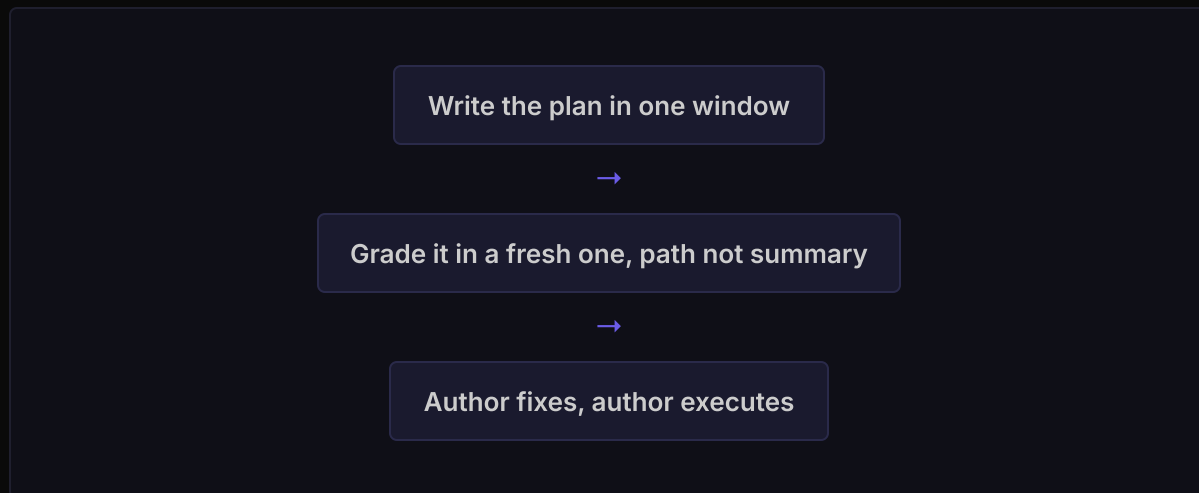
Build it like this:

1. It takes a file path. It NEVER summarises the plan for the grader. The grader must read the file itself.
2. It spawns a separate agent with a clean context and this job:
"Find everything wrong with the plan at . Assume whoever wrote it is trying to fool you. Read the plan itself, do not accept anyone's description of it. Verify every number it quotes by counting the live thing, never by trusting the plan. Check whether each step's premise actually holds. Check whether step N genuinely unblocks step N+1. Say what the plan is deliberately NOT doing. Ask whether I could start step one without a clarifying question. Report every real problem with evidence, then anything genuinely strong. Be specific."
3. When the grader reports back, relay its findings to me VERBATIM. Do not soften them and do not defend the plan. If you wrote the plan, you are the last one whose defence is worth anything.
4. Write in the skill's own notes that a spawned agent is LESS independent than a totally separate terminal, because it receives a prompt from the session that wrote the plan. So this is the cheap default. For a plan that really matters, I should still open a second terminal and paste it cold.

I am not a programmer, so explain what you are doing in plain English as you go, and tell me exactly what you changed on my machine.

STEP 3

The framework: when to grade, and what a real finding looks like



Grade anything that matters. A plan you are about to run across your business. A number you are about to put in front of a customer. A "this is fixed" you are about to believe. If being wrong would cost you a day, grade it. If it would cost you a minute, do not bother.

The author still executes. The blind spot is about checking, not doing. The window that wrote the plan knows why each step is in the order it is. Hand execution to a cold window and you throw that reasoning away. Grade cold, execute warm.

A finding names a specific thing. The grader saying "this could be stronger" is worthless. The grader saying "step 4 assumes the export runs nightly and nothing schedules it" is worth the whole exercise. If your grader gives you vague feedback, send the prompt again and add: be specific or say nothing.

THE TWO MISTAKES THAT BREAK IT

Summarising instead of pasting. The first time I tried this properly, I gave the grader a description of the plan instead of the plan. It graded my description. Give it the file.

Defending the plan. When the grader comes back with sixteen problems, the window that wrote the plan will want to explain them away. Do not let it. Read the findings cold, fix what is real, and move on.

Never trust one AI. Make it argue.

That is the entire system. Two windows, one prompt, and the discipline to hand the grader the real thing. It caught a silent failure that would have cost me a week, and it has caught something on almost every plan since.

That is the whole system.

Zero ad spend. Zero coding experience. The other playbooks live at brycenwood.com/playbook, one download each.

Built by Brycen Wood · [@brycenwood.ai](https://twitter.com/brycenwood.ai) · brycenwood.com