Idea To Prototype Pipeline

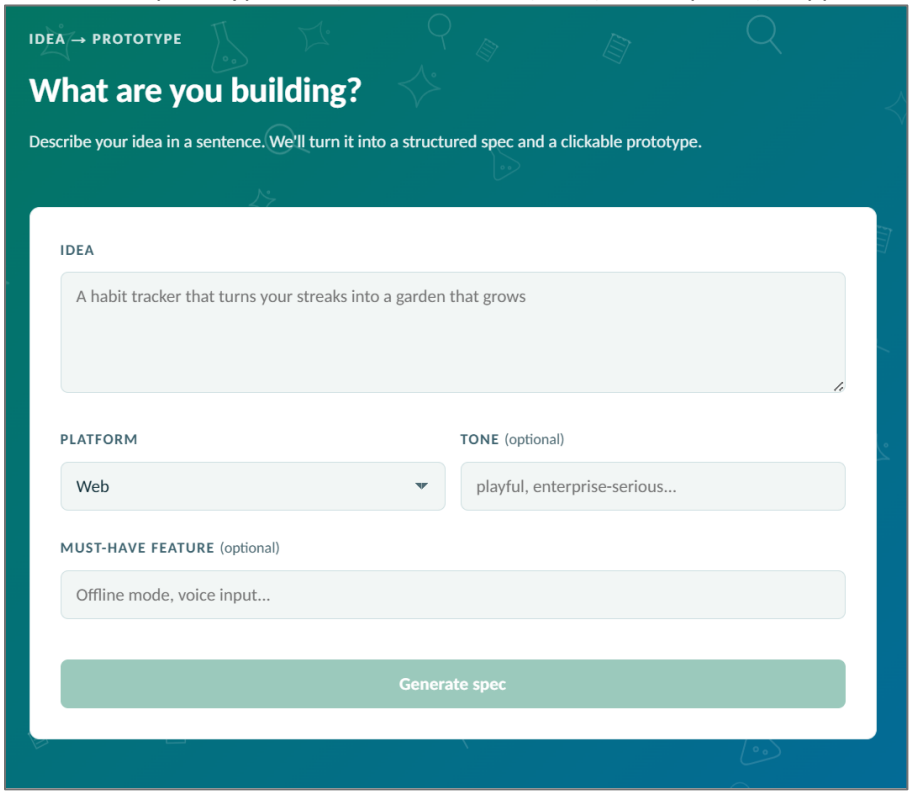
Using chained AI tools to go from a one-line idea to a working prototype

Meghna Bhagwat | Product Manager | LinkedIn | Live tool: Click here

Why I Built This

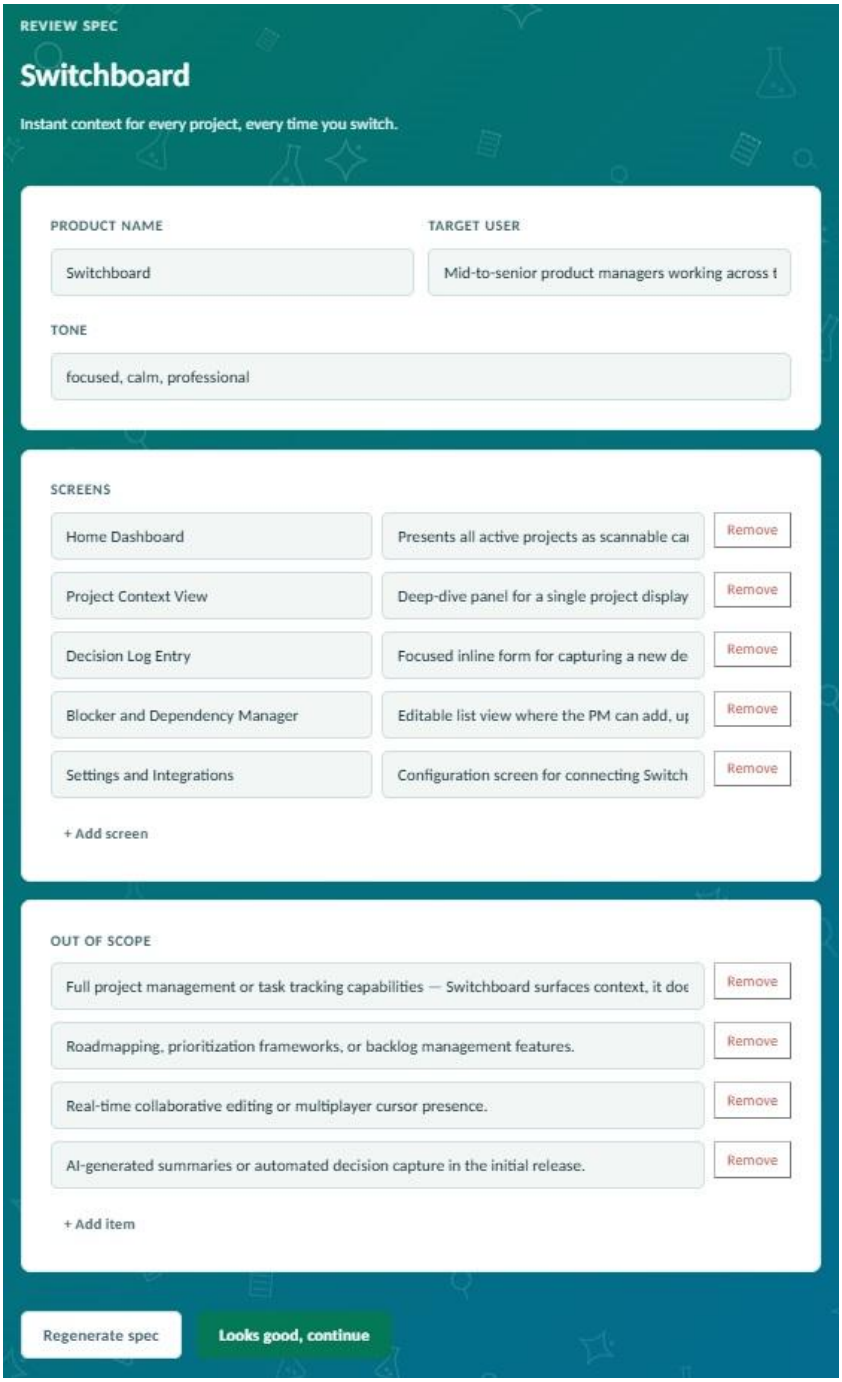
As a PM, I regularly need to turn a rough idea into something a stakeholder or engineer can respond to. Normally that means writing a spec, then waiting on design or engineering time to see it as a real screen.
I wanted to test whether AI could compress that loop, turning an idea into a clickable prototype in minutes, without needing to pull anyone else in first. This project is that test. It produced a working pipeline, along with what I learned about where AI tools help and where they don't.

Tools: Claude (spec generation and prototype code) + Claude Code (build) + Sandpack (in-app rendering)



What I Built

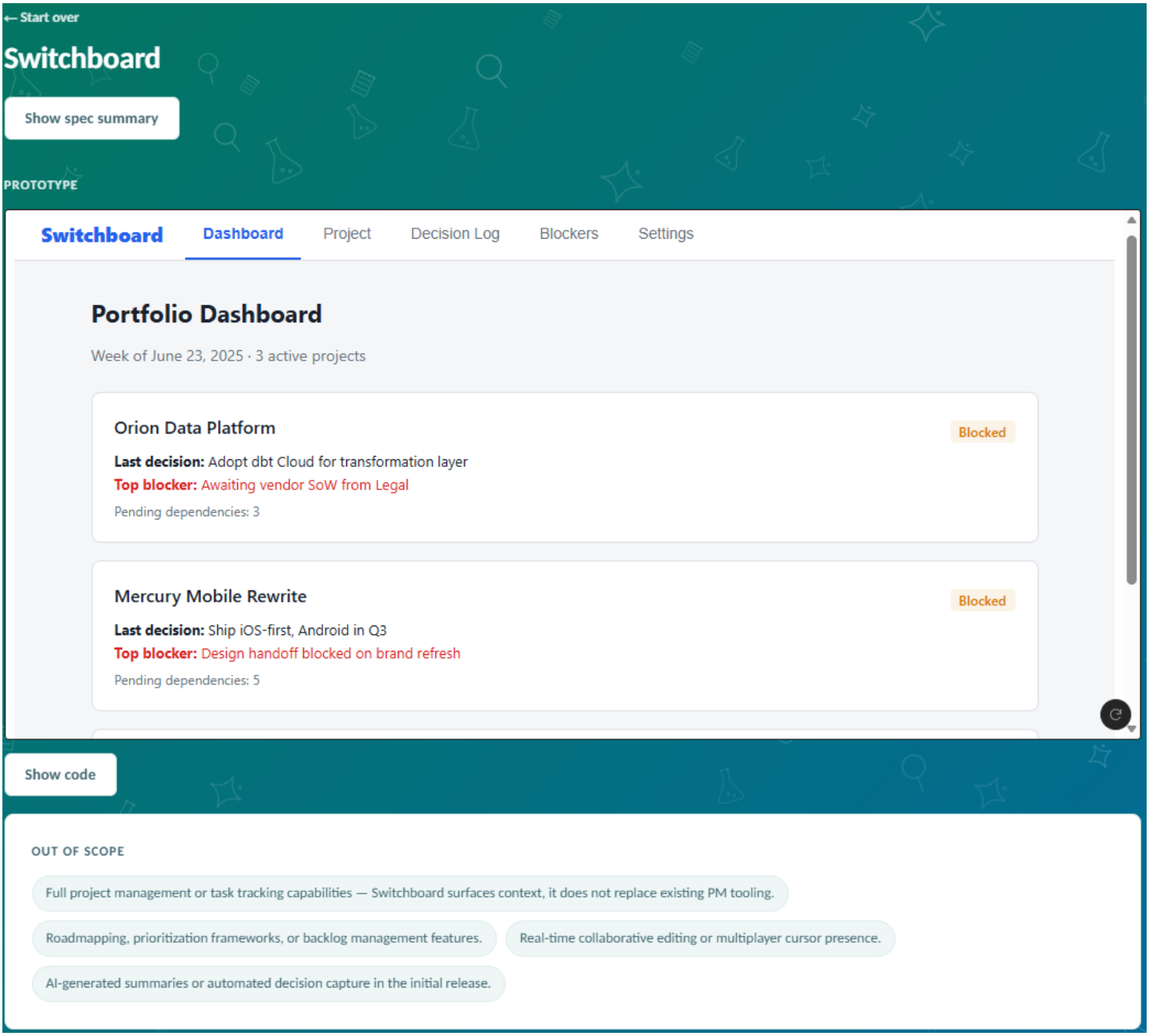
This is a three-stage tool:
1. **Gather** - a short input describing the idea, plus a few optional details (platform, must-have feature, tone).
2. **Spec** - Claude turns the idea into a structured spec covering the problem, target user, screens, and what's explicitly out of scope. I review and edit this before continuing. This is the one checkpoint in the pipeline, so I always know what got sent forward.
3. **Prototype** - the approved spec is used to generate a working, clickable prototype, rendered live inside the app.



Working Example

I tested the pipeline against a real PM pain point. I wanted a dashboard that helps PMs quickly re-orient when switching between projects, showing the last decision made, what's blocking progress, and who's waiting on what for each workstream.

Claude's spec correctly scoped this into a small set of screens, an overview and a per-project detail view, and flagged full project management and real-time collaboration and notifications as explicitly out of scope for a first version, without me having to ask for that scoping.



What I learned

Negative prompts don't reliably work in image generation

While testing an image-generation stage for visual moodboards, I ran a prompt based on my Sentiment Analyser project (also featured in my portfolio), a tool that compares user opinions about competing products. The prompt named that domain directly and included an explicit instruction to avoid charts and text, but the image still came out with garbled text and chart-like shapes. Naming a concept, even to forbid it, seems to make the model generate it anyway. Removing domain-specific nouns and describing the tone, color, and shapes instead fixed this reliably.

Cut what doesn't earn its place

I tested the moodboard stage twice, first as a standalone artifact, then redesigned to feed the prototype stage as a visual style reference. The generated prototype showed no visible influence from it either time. Since it added no functional value to the pipeline, I removed it.

Hosted platforms carry real, ongoing risk, not just a one-time restriction

The first prototyping tool I used, v0 by Vercel, generated real, working screens, but its hosted preview could not be embedded inline in the results page, a standard security restriction most platforms set on their own domains. I worked around it with a plain link out to the live prototype instead of an inline view. Weeks later, that same link stopped opening entirely, independent of anything in my own code. I looked at other hosted prototyping platforms, including Bolt, Replit, and Lovable, and found the same underlying pattern, each depends on its own hosted preview infrastructure, with no guarantee it stays embeddable or even reachable over time. I replaced v0 with Claude generating the prototype code directly, rendered live inside the app using Sandpack, an open source library. This removed the dependency entirely, since nothing after generation relies on an external service staying available.

How I Tested It

I timed the pipeline on real ideas, from typing the idea to seeing the finished prototype. Average time was about 2 minutes, ranging from roughly 1 to 3 minutes.

Seven people from my network reviewed the generated prototypes, 3 PMs, 2 designers, 1 product owner, and 1 developer. The PMs and the product owner said the specs were generally good but usually needed some edits before approval. They specifically liked the out of scope list, useful either to keep as-is or to show in a stakeholder review to make clear a feature isn't planned for the MVP. Designers said it sped up their work, but the screens still needed edits before moving forward. Depending on urgency, getting a design pass normally takes anywhere from same-day to a full iteration, and this shortened that starting point considerably. The developer was glad to see working code generated alongside the clickable screens instead of just a visual mockup.

One clear limitation came up across reviewers, prototype quality depends heavily on the tone input. Left for the AI to decide, the output is noticeably less creative, so I almost always set the tone myself to get a good look and feel.

Most reviewers said they'd show the prototype to a stakeholder after editing the spec first, not as-is. Overall feedback was positive.

Conclusion

The real value here is compressing the starting point of the idea-to-something-clickable loop, not the whole thing. Reviewers still edited specs and screens before using them, but got there in minutes instead of waiting on a full design or engineering cycle. For a PM, that means testing more ideas before committing real resources to any one of them, with something concrete to react to earlier in the process.

The build also surfaced a broader lesson about relying on other platforms inside a pipeline. Each stage needs to be tested in isolation before trusting it, and any dependency on someone else's hosted service is a real risk worth weighing before committing to a stage. The most durable version of this pipeline turned out to be the leanest one, one model handling both the thinking and the code, with the result rendered directly inside the app instead of handed off elsewhere to display.