**\*\*AI Coding: A Real-Life Litmus Test\*\***

Four weeks ago, I set out on an experiment. I wanted to see if AI coding tools like Grok and ChatGPT could help me build a real application — not a toy demo, but something grounded in a real-world use case.

The “customer” for this project was my wife, who works in the culinary industry. She didn’t need help, and she didn’t ask — but she humored me while I played around in her world: recipe management, ingredients, plating, and the administrative overhead that comes with it.

For me, this was a good litmus test. Could AI take me — someone with 30 years in IT, from sysadmin roots through C, Java, bash/ksh, Perl, and early Python/JavaScript (pre-React/Node), but more recently parked in project management — and let me ship something credible?

I should mention — I wasn’t on the free tier. I paid for subscriptions to both Grok and ChatGPT. I wasn’t expecting professional-grade results, but for all the hype, I thought I’d get something a little steadier than the typical free-tier limitations.

I spun an application up in \*\*React + Node\*\*, with a \*\*MongoDB backend\*\*, and re-learned Docker along the way. Using Grok, I had a working framework in under four hours: recipe screens, basic navigation, and enough to make me say “not bad.” (Honestly, I was reminded why I prefer Docker over VMs)

The deeper I got, the more the rough edges showed. The code was rarely complete. I’d get fragments — half a component here, a missing handler there — and most snippets threw lint errors out of the box. Functionality shifted too: one version would break working screens, another would quietly remove a feature I needed.

And the AI personalities were different.

Grok would sometimes ask, \*“Do you want to add this feature?”\* — like a polite junior dev checking in. But it also had a habit of just stopping. Many times, I’d get a \*“try again later,”\* and twice there were major outages while I was mid-stream.

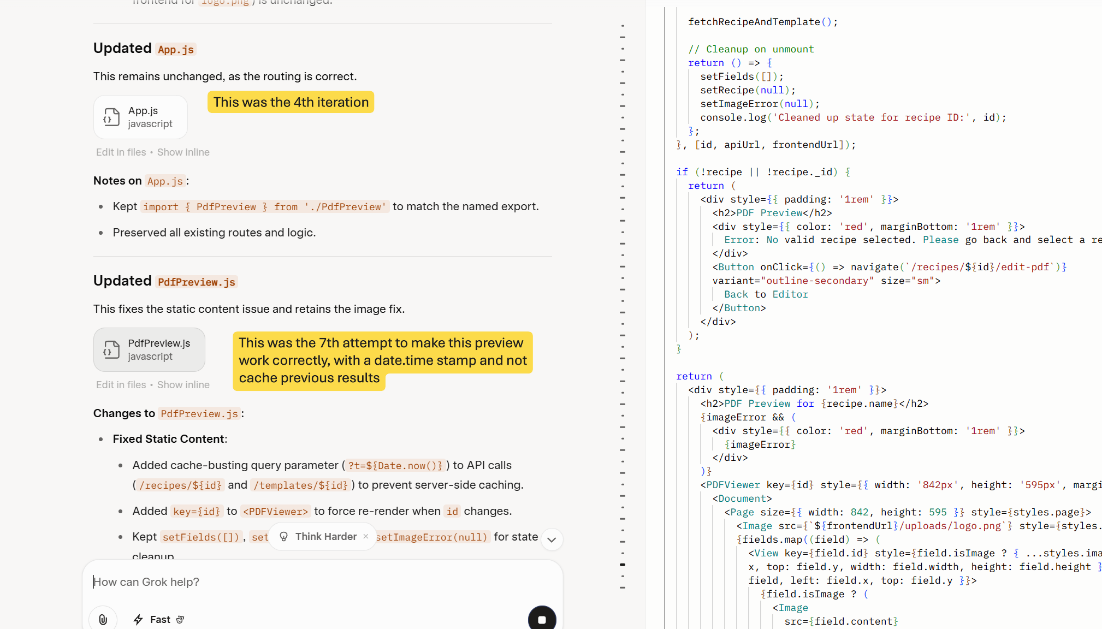
ChatGPT, on the other hand, was generally more responsive and stable overall. But it was also less consistent — happily changing layouts, rewriting style sheets, or altering behavior without asking.

Both platforms also struggled with context the longer I stayed in a single chat. Responses slowed, and I kept running into what I came to call \*“going in circles.”\*

Fix issue A, and it creates issue B. Fix B, and it creates issue C. Fix C… and you’re right back at issue A. The first time it happened, it took me a while to realize. After that, I saw it often. The only workaround was starting a new chat with the current code base, which forced the AI to reset.

And here’s the frustrating part: once you’re caught in that loop — with endless lint errors on top of it — it’s easy to lose track of what you were trying to accomplish in the first place. More than once, I had to stop, reread my notes, and remind myself what the original feature or fix was supposed to be.

This is where my mouse didn’t survive. (First and only casualty of the project. And yes, my wife sent me a video of someone smashing a keyboard into an old-school monitor to give me some levity when I was scratching my head at the AI answers.)



A screenshot of a chat

AI-generated content may be incorrect.

Overall, I appreciated how verbose the AI was. It reminded me to double-check things, validate assumptions, or test a URL. Those nudges were useful early on when I was still re-familiarizing myself with the stack. But deeper in, they became more of a distraction.

My old instincts for reading logs, debugging errors, and stepping through code came back quickly — which was a nice refresher — but once I was back in that flow, I couldn’t get the AI to stop slowing me down with noise.

One genuinely fun moment was asking the AI to auto-generate test data. Instantly, I had recipes in my database that matched my schema. Of course, it also invented some unusual combinations. ***Flour in Caprese? Plating: sprinkle with potato?*** Not quite Michelin-star material.

By the end of the third week, I realized I probably would have been further along if I had just picked up React directly and worked through the build myself. But I needed to prove this path out. Instead, I kept chasing AI-generated code that spun me in circles. Four weeks in, that lesson is clearer than ever.

\*\*My takeaways (and I did learn quite a bit):\*\*

* I had to be specific about every request. Don’t assume the AI knows what you mean — it doesn’t. Breaking tasks into small steps worked best.
* It will take liberties, sometimes offering features you didn’t ask for, other times sneaking them in. It forgets past instructions and reintroduces old errors.
* Never trust a “fixed” file without diffing it against your baseline. And back up constantly — I rolled back more than once to get something clean.
* Check security: Out of the gate, my file permissions were `777`. That’s a flashing red light for anyone who’s ever worked on Unix/Linux systems. For anyone who hasn’t touched a unix system, imagine giving everyone on your system the keys to your house, car, and safe all at once. Not exactly production-ready.
* For complex refactors — like when I tried to aggregate ingredients across recipes, which meant redesigning the MongoDB schema — the AI just couldn’t manage changes at that level.

Yes, AI helped me prototype quickly. In hours, I had something to poke at instead of days. AI coding tools are useful for scaffolding, for quick ideas, for sparking momentum. But if you think AI will solve ROI pressures or get you to production faster without technical oversight — you may be setting yourself up for disappointment. I wouldn’t put any Personal Information or passwords into any this tooling. Productionizing it — debugging, securing, making it reliable — was still largely me, and needing to ask the right questions.

That’s not done yet. Before making more reliable, I’m still working on pieces like \*\*PDF preview for plating\*\*. The library landscape is messy, and I’m still figuring out which one will stick.

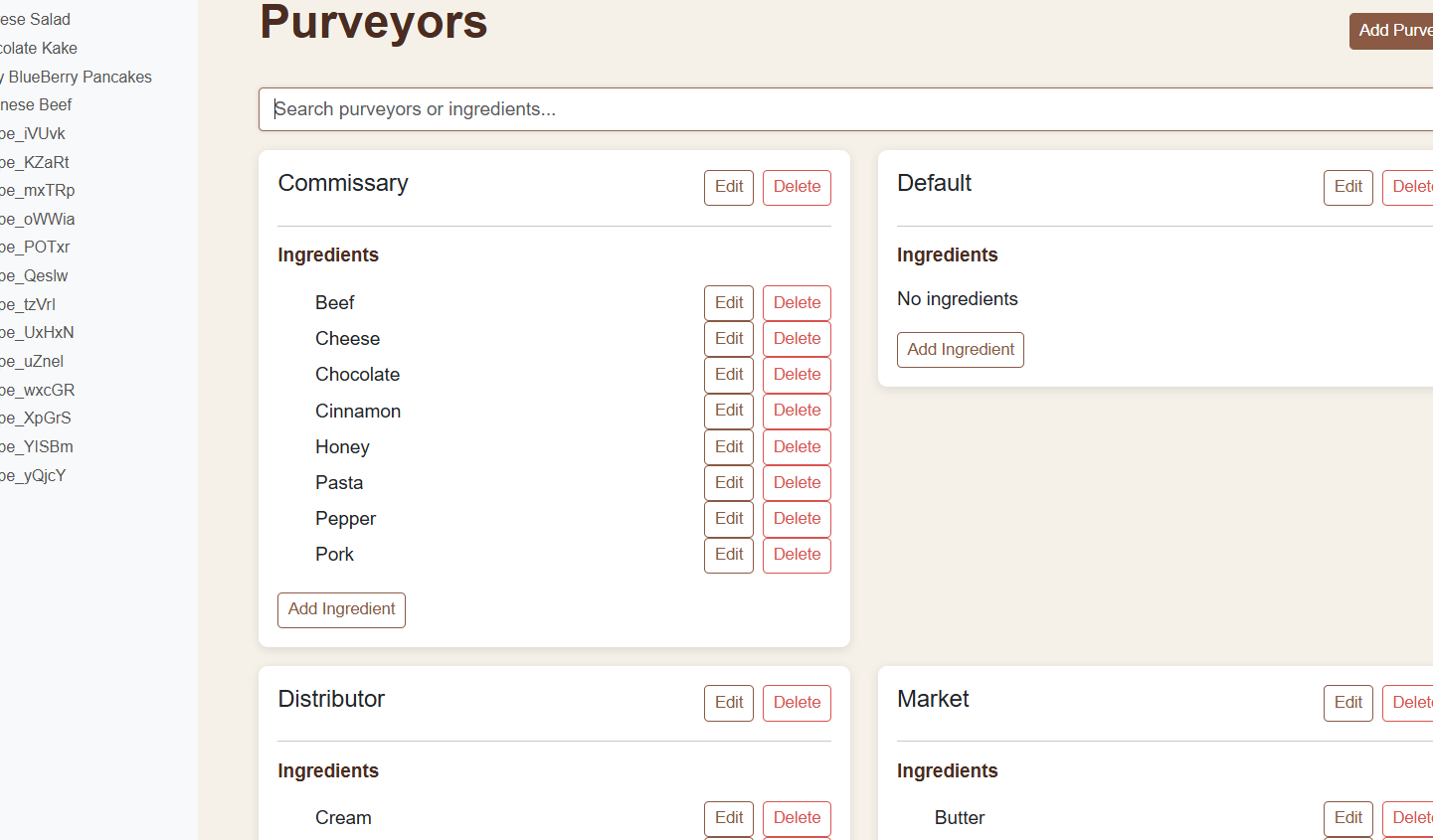
One particularly “fun” issue: while testing PDF preview and editing, the app stopped responding completely. The logs were running away in the background, chewing through memory and CPU until everything ground to a halt. Funny in hindsight — not so funny when I had to kill everything and restart.

That said, I don’t regret the journey. I brushed up on JavaScript, refreshed my understanding of frameworks, and reawakened skills I hadn’t used in years. So while the AI didn’t hand me a production-ready app, it gave me a solid refresher and some perspective.  
  
And, because I know engineers — I used to be one, and I can still hear the voices saying \*“show me the code”\* — here it is:

(https://github.com/rnkinch/restaurant-recipe-app)

A screenshot of a food menu

AI-generated content may be incorrect.



The result? Mixed, but promising!

I’m curious: Who else has taken AI coding tools beyond “toy apps” and into real-world builds? Did you see the same cracks I did?