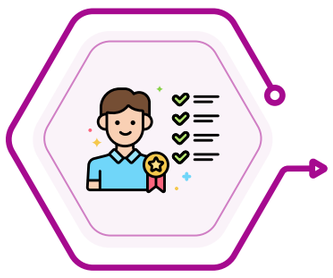
From Vibe-Coding Skeptic to Antigravity Convert:
How AI Tools Got Me Building Again

I'll confess to something that probably ought to embarrass me more than it does. For a long stretch, I treated AI coding assistants as little more than a punchline. The phrase “vibe coding” alone was enough to make me roll my eyes. These days I open one of these tools almost every week to put together something small and genuinely useful, and I'd have a hard time going back to life without them.

A degree left to gather dust

I picked up a computer science degree in the early 2010s and then quietly let most of what I learned slip away. That's on me and nobody else. The honest version of the story is that I never really enjoyed the act of programming itself. Dreaming up what something could become was always far more fun than the slow, grinding work of actually building it. Implementation felt like an obstacle parked between a good idea and a finished product, and I rarely had the patience to push through it.



So the skills sat idle. I tinkered with the occasional side project, abandoned most of them, and made peace with the idea that some of my concepts were destined to stay concepts. If you're a developer reading this and thinking those projects were better off unbuilt, that's a fair shot, and I won't argue.

The offhand suggestion that flipped a switch

My default solution to repetitive digital chores was always to go shopping. I'd hunt for a plugin, dig through extensions, or shell out for an app that did roughly what I wanted, even when none of them fit quite right. It never occurred to me that the better move was to simply have the software built to my exact specification.

01

That changed during a casual conversation with a colleague about workflow headaches. He asked, almost as a throwaway, why I didn't just point an AI tool at the problem and have it build the thing I kept griping about. I was a little stunned that the idea had never landed for me. I'd assumed I was reasonably plugged into what modern tools could do, yet I'd never seriously tested the waters here. My earlier attempts at getting a chatbot to spit out scripts had been clunky and frustrating, mostly because a plain chat window is a miserable place to do anything code-related.

02

A purpose-built environment that helps you map out a plan before turning the model loose was a completely different experience. A few hours of experimenting in, it felt like a door had swung open. Suddenly I could commission a tool tailored to my own needs instead of paying a premium for a generic add-on that solved eighty percent of the problem at best.

Tools that erase the busywork

The tasks I'm describing aren't glamorous, and that's exactly the point. Stamping watermarks onto a batch of images. Converting and compressing files in bulk. Making the same small color tweak across a dozen assets. Running the same tedious export over and over. On their own, each one is trivial. Stacked up across a workweek, they quietly eat hours and patience in equal measure.

01

Over the past several months I've assembled a small collection of personal utilities that quietly handle exactly these annoyances. None of them are impressive in isolation. Together, they've handed me back real time, smoothed out processes I used to dread, and let me spread my attention more evenly across the parts of my job that actually matter.

02

I want to be clear about one thing: I would never claim I “built” these tools in any traditional sense. What I really did was hand the concept off to a capable assistant and guide it toward the result I had in mind. Think of it as delegation. You're briefing a developer who needs clear, structured direction to do good work, and who will hand you a mess if you're vague. Some engineers I know compare these tools to a junior developer, and that framing feels about right.

Where personal use is the sweet spot

Here's the line I draw firmly in the sand: personal use. I'm entirely with the developers who caution against shipping vibe-coded software out into the world. If you're going to publish something, you really do need to understand how it works, well enough to troubleshoot and fix it when it breaks. You don't have to know every line by heart, but a grasp of the inner workings is non-negotiable once other people depend on your code.

You can throw together a prototype in minutes, sure. Turning that prototype into something that actually works reliably can take hours, days, or weeks. Plenty of people lack the patience to carry a project all the way through, and that's a genuine trap. Getting a project ninety percent of the way there is the easy part; the final ten percent can swallow more time than everything before it combined. That gap is precisely what soured me on conventional software work years ago, and guiding an AI through those last stubborn issues is still hit-or-miss.

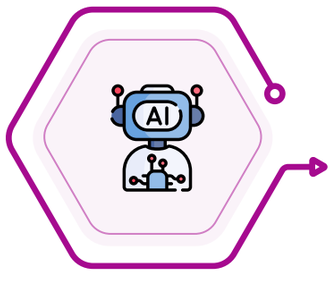
Respect the guardrails

None of this comes free of risk. Putting powerful generation tools in the hands of people who can't read what they produce is asking for trouble, because these projects can spiral into chaos without firm direction. Messy code tends to breed more messy code. That matters far less when whatever you've made never leaves your own machine, which is exactly why every tool I've built stays strictly local.

Security is the part that deserves the most caution. When you can't fully see what a piece of software is doing, you can't be confident about the vulnerabilities hiding inside it. Anyone tempted to scale a vibe-coded project beyond personal use should treat that uncertainty as a serious red flag and think hard before going any further.

Worth a real shot

Even with the pitfalls and the rough edges, I can't help but nudge anyone with an idea to try a little of this for themselves. As more lightweight, AI-assisted tools fill the gaps that off-the-shelf software leaves behind, the case for rolling your own gets stronger. The result might be a touch hacky, but a patchwork of small custom tools can end up greater than the sum of its parts.



If you've been holding out, this is a good moment to give Antigravity, Codex, or Claude Code an honest try. I came in as a skeptic and walked out a convert, and I genuinely can't recommend the experiment highly enough.