From Sceptic to Convert:

How Vibe Coding Quietly Rewired the Way I Build Software

A trip to Shenzhen, an unfamiliar startup, and three months of late nights turned a buzzword I had been dodging into the most productive tool I have ever used.

01

For most of my career, the gap between having an idea for software and actually building it has felt enormous. I have spent years working around the edges of products that frustrated me, asking developers to bend platforms into shapes they were never designed to take, and quietly accepting that some ideas would simply never leave my notebook.

02

Then I went to China. On the last day of the trip, I sat down with a startup I had never heard of called YouWare, and that meeting did something I did not expect. It turned vibe coding from a piece of online noise I had been comfortably ignoring into the most useful skill I have picked up in the past decade.

The meeting that broke the mystique

Like most people who spend any time on tech social media, I had encountered the term vibe coding before. I had seen the screenshots, the threads, and the bold claims. I had also assumed, without really investigating, that it was probably overhyped and that anything serious would still require a proper engineering team.

A few hours in a Shenzhen office shifted that view permanently. The YouWare team was not a group of hobbyists. It included former product managers from major companies, and the founder, Leon Ming, came out of Moonshot AI and CapCut. Watching the team move from a sentence of intent to a working interface in front of me made the abstraction collapse. There was no mystery left. It was just a faster way to make software.

Twenty years of ideas finally getting built

Once I got home and actually tried it, the dam broke. The first thing I built was a small events platform for a series of House of Tech events I had been planning around major trade shows. That was supposed to be a one-off experiment. Instead, it became the on-ramp to something much bigger.

01

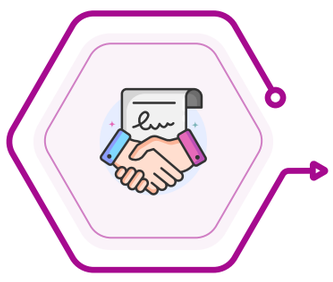
Within weeks I had vibe coded multiple platforms, including a new content management system I am calling Cora CMS. Every feature I had ever tried to graft onto WordPress, every plugin I had paid developers to build, every aborted attempt at a custom newsletter pipeline, all of it now lives inside one product that I actually own. It pulls workflows from Substack, Asana, and Mailchimp, and it is designed around how I have actually worked as a reporter for the past twenty years, not how a generic CMS thinks I should work.

02

That is the part of vibe coding that nobody quite captures in a hot take. Disruption in any industry tends to follow the same pattern. A newer player studies what the incumbents got wrong and builds a product that quietly refuses to repeat those mistakes. The barrier was never the idea. The barrier was assembling the team. With a sharp opinion, a paid AI subscription, and a willingness to put in the hours, that barrier is gone.

The trick is making the AI argue with you

There is one technique that has made the difference between toy projects and software I would actually ship. I tell the agent to push back.



Out of the box, most coding agents lean towards agreement. You ask, they comply. That is fine for a quick script, but it is dangerous when you are designing something that will need to live for years. The version of vibe coding that works for serious products is the one where the agent challenges the request, points out the trade-offs, and sometimes talks you out of an idea entirely. In that mode it stops feeling like an autocomplete and starts feeling like a thoughtful collaborator with access to a huge amount of context.

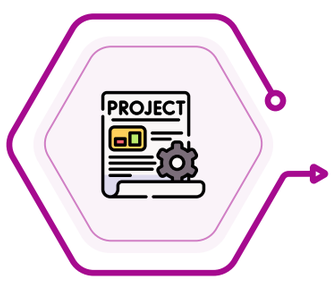
The honest cost: it still eats your evenings

I want to be careful not to oversell this. There is a quiet myth around vibe coding that suggests anyone can produce serious software in an afternoon. That has not been my experience, and I think anyone selling that version is doing the rest of us a disservice.

Building real products still demands real sacrifice. In one three-week stretch I pulled at least one all-nighter every week. At some point my sister and niece mentioned that they had not seen me in a while, which is the kind of feedback you cannot brush off. The difference between vibe coding and traditional development is not that the work disappears. The difference is that the bottleneck moves. You are no longer waiting on a developer's calendar. You are now the bottleneck, and your time is the resource you are spending.

Old gadgets, new jobs

One unexpected side effect of all this has been the way my hardware has shifted in usefulness. I owned a stack of tablets for vague reasons that I struggled to defend. Now they are part of a working setup.



With Astropad's Workbench app on my iPad Pro M5, I can drop into my Mac from anywhere in the house and keep iterating on a project. On the Android side, Parsec and Splashtop do something similar. The result is that tablets which used to sit in a drawer now live in most rooms of the house, charged and ready to pick up wherever the last thought left off. It is a small thing, but the parallel agent workflow that vibe coding encourages turns these devices from indulgences into endpoints.

The trap nobody warns you about

If there is one warning I would attach to all of this, it is the temptation to build everything. Once the friction drops, the urge to spin up a new project for every minor irritation in your day becomes almost overwhelming. You start vibe coding solutions to problems that did not really need solving.

The old engineering wisdom still applies. Just because you can build it does not mean you should. Ask Claude or any other capable model to be honest about a project idea and it will frequently tell you not to bother. That is a feature, not a flaw. The most valuable skill in this new world might turn out to be the discipline to walk away from the keyboard when the idea is not actually worth shipping.

Why this is not a phase

Three months in, I do not think vibe coding is a fad that the tech industry will quietly retire. The economics are too clear. Anyone with strong opinions about a product category, a clear sense of what is broken in the incumbent tools, and the patience to iterate can now produce software that would once have required a funded team. That is a structural change, not a trend.

Will every vibe coded product survive contact with real users? Almost certainly not. But the ones that do will tend to be the ones built by people who actually understand the problem they are solving, rather than by teams that won the funding round. For anyone who has been holding back because the term sounded faintly embarrassing, my advice is straightforward. Stop ignoring it. Sit down with the tools for a weekend and see what happens. You may end up, like me, with a backlog of twenty-year-old ideas suddenly becoming things you can ship.