

Why Your Vibe-Coded App Isn't Production-Ready (And What to Do About It)

You had an idea. You opened Lovable, Bolt, v0, or whatever AI tool you prefer, and within a few hours you had something that actually works. A real interface, real data flowing, something you could show to people and say "this is what I'm building."

That's genuinely impressive. A year ago, that would have taken a team of developers weeks to pull off.

But now you're trying to take it further, and things are getting sticky. The app is slow. It crashes when more than a handful of people use it. You're scared to touch anything because every change seems to break something else. And somewhere in the back of your mind, you know the security situation is... not great.

You're not alone. This is the reality for nearly everyone building with AI coding tools right now. The prototype-to-production gap is real, and it's wider than most people think.

What AI Coding Tools Are Great At

Let's be clear: these tools are extraordinary for certain things.

- **Speed of iteration.** You can go from concept to clickable prototype in hours.
- **Visual output.** The UIs these tools generate are often surprisingly polished.
- **Breadth of features.** You can bolt on authentication, payments, databases, and more with simple prompts.
- **Validation.** You can test whether an idea has legs before investing serious money.

If you're using these tools for prototyping and validation, you're doing it right. The problems start when you try to run a prototype as a product.

Where Things Break Down

1. Architecture

AI tools generate code that works for the prompt you gave them, not code that's designed for what comes next. There's no thought given to separation of concerns, data flow patterns, or how the system will evolve. Everything tends to be tightly coupled, which means changing one thing risks breaking three others.

2. Security

This is the big one. AI-generated apps routinely ship with:

- API keys and secrets visible in frontend code
- No input validation or sanitisation
- Basic or no authentication
- No rate limiting
- Direct database access from the client
- No encryption for sensitive data

Any one of these would fail a basic security review. Together, they're a data breach waiting to happen.

3. Performance

The code works, but it's not efficient. Common issues include:

- Database queries that fetch everything instead of what's needed
- No caching at any layer
- No connection pooling
- Synchronous operations that should be asynchronous
- No pagination for large datasets
- Unoptimised images and assets

These don't matter when you're the only user. They matter a lot when you have 100.

4. Cost

Unoptimised code running on unplanned infrastructure gets expensive fast. We've seen AI-generated apps with cloud bills 10x what they should be, simply because nobody thought about query efficiency, caching, or right-sizing the infrastructure.

5. Maintainability

AI-generated code has no consistent patterns, no documentation, no tests, and no clear structure. The AI doesn't think about the developer who comes next. Every change becomes archaeology, trying to understand why something was done a certain way before daring to modify it.

What "Production-Ready" Actually Means

A production-ready application isn't just one that works. It's one that:

- **Stays up** when real users hit it concurrently
- **Stays secure** against common attack vectors
- **Stays fast** as your data and user base grow
- **Stays affordable** as you scale
- **Stays maintainable** as your team and requirements change

Getting there from an AI-generated prototype typically requires work in these areas:

Authentication and Authorisation

Proper user management, role-based access, session handling, and secure token management. Not just "can they log in" but "can they only see and do what they're supposed to."

Database Design

Normalised schemas, proper indexing, efficient queries, migration strategies, and backup procedures. The difference between a database that works and one that works at scale.

API Design

Consistent endpoints, proper error handling, validation, rate limiting, versioning, and documentation. The backbone of a reliable application.

Error Handling

Graceful degradation, user-friendly error messages, proper logging, and alerting. When something goes wrong (and it will), you need to know about it before your users tell you.

Testing

Unit tests, integration tests, and end-to-end tests that give you confidence to ship changes without fear. Not every line needs a test, but the critical paths absolutely do.

Infrastructure

Proper hosting, CDN configuration, environment management, CI/CD pipelines, monitoring, and alerting. The unglamorous work that keeps everything running.

The Path Forward

If you've built something with an AI tool and you're hitting these walls, you broadly have three options:

- 1. Learn to do it yourself.** This is viable if you have the time and inclination to become a software engineer. It's a months-to-years journey, but it's valuable.
- 2. Hire a developer or team.** This works, but finding good developers is hard, expensive, and slow. And many will want to throw away your prototype and start from scratch.
- 3. Work with a team that specialises in this exact problem.** This is what we do at SlantedBox. We take AI-generated prototypes and make them production-ready. We don't start over. We understand what you've built, keep what works, and rebuild the parts that need engineering rigour.

What to Look For in a Partner

If you go the partnership route, look for:

- **Experience with AI-generated codebases.** Not all developers understand the specific patterns and problems these tools create.
- **Honest assessment.** A good partner will tell you what actually needs to change, not try to rebuild everything to pad the invoice.
- **Clear process.** Audit first, then plan, then build. No surprises.
- **Focus on handoff.** The goal should be a codebase you own and can continue to develop, not a dependency on the partner.

The Bottom Line

AI coding tools have democratised the ability to build software prototypes. That's a genuine revolution. But the gap between prototype and product is still very real, and it requires different skills to bridge.

If you've built something you believe in and you're ready to take it to the next level, that's exactly the kind of challenge we love.

SlantedBox Launch is our service for founders and builders who've validated their idea with AI tools and need engineering expertise to go live. [Learn more about Launch](#) or [get in touch](#) to discuss your project.