Vibe Coding Playbook 001: 20+ Brutal Lessons From 6 Months of Vibe Coding

> Website: vibecodelab.co
> X:
> @vibecodelab

Thanks for grabbing this doc. If you're reading this, you're probably building with AI and trying not to lose your mind in the process.

I've spent the last 6 months building and shipping multiple products using Cursor and other tools. One is a productivity-focused voice controlled web app, another's a mobile iOS tool — all vibe-coded, all solo.

Here's what I wish someone told me before I melted through a dozen repos and rage-uninstalled Cursor three times. No hype. Just what works.

I'm not selling a prompt pack. I'm not flexing a launch. I just want to save you from wasting hundreds of hours like I did.

1. Start like a Project Manager, not a Prompt Monkey

- Write a real PRD before anything else.
- Describe what you're building, why, and what tools you're using.
- Keep it in your root directory as `product.md` or `instructions.md`.
- Reference this often — AI context evaporates quickly.

2. Add a Deployment Manual. Yesterday.

- Document how to ship: branches, env vars, servers.
- Save yourself from 2am guesswork.

3. Git or Die Trying.

- Cursor will eventually break something.
- Use version control aggressively.
- Use local changelogs per folder.
- Git saves tokens and gives your AI breadcrumbs.

4. Short Chats > Smart Chats

- Don’t hoard 400-message threads.
- Start a new chat per issue.
- Keep scope tight. Give clear commands.

5. Don’t Touch Anything Until You’ve Scoped the Feature

- Use GPT/Claude to map the full feature first.
- Choose a single approach.
- Cursor is for execution, not ideation.

6. Clean Your House Weekly

- Delete temp files and dead code.
- Reorganize folders.
- Clean codebases make better prompts.

7. Don’t Ask Cursor To Build the Whole Thing

- Use it for stubs, logic chunks, or controlled refactors.
- Full apps? You’re asking a blender to make steak.

8. Ask Before You Fix

- Debugging? Ask AI to investigate first.
- Have it suggest multiple fixes.
- Pick one before asking it to code.

9. Tech Debt Builds at AI Speed

- AI helps you MVP fast. It also digs you a mess faster.
- Pause to refactor. Keep the architecture clean.

10. Your Job is to Lead the Machine

- Cursor isn't your developer. You are.
- Use `.cursorrules` and git checkpoints.
- Do the system thinking. AI does the typing.

11. My main prompt for all the projects

💡 DO NOT GIVE ME HIGH LEVEL STUFF, IF I ASK FOR FIX OR EXPLANATION, I WANT ACTUAL CODE OR EXPLANATION!!! I DONT WANT "Here's how you can blablabla"

Be casual unless otherwise specified

Be terse

Suggest solutions that I didn't think about—anticipate my needs

Treat me as an expert

Be accurate and thorough

Give the answer immediately. Provide detailed explanations and restate my query in your own words if necessary after giving the answer

Value good arguments over authorities, the source is irrelevant

Consider new technologies and contrarian ideas, not just the conventional wisdom

You may use high levels of speculation or prediction, just flag it for me

No moral lectures

Discuss safety only when it's crucial and non-obvious

If your content policy is an issue, provide the closest acceptable response and expl

I am using macOS

12. Set Your Cursor Rules Properly

- Use https://cursor.directory/rules
- Pull rules by framework (Next.js, etc.).
- Add rules for folder structure, formatting, testing.
- It turns Cursor from a toddler into a disciplined intern.

13. Use .cursorrules Files for Global Sanity

- Write clear guidelines on what AI should and shouldn't touch.
- Helps avoid random refactors and hallucinated code cleanup.

14. Copy Structure from Open Source Projects

- Mimic their project structure, env handling, scripts.
- Don't reinvent boilerplate. Borrow it.

15. Level Up Your UI Game With Kits

- Use https://ui.shadcn.com/ and https://21st.dev/
- Prebuilt components save hours.
- Better UI = more credibility when you ship.

16. Cursor Is Dumb When You Are

- Garbage in = garbage out.
- Use parallel sessions in Claude or GPT to think better.
- Come to Cursor with answers, not questions.

17. Prompt in Layers

- Use GPT to plan.
- Use Claude to critique.
- Use Cursor to code.

18. Use YOLO Mode With Boundaries

- Enable YOLO for speed.
- But use git check-ins every hour.

- It’s chaos with a seatbelt.

19. Create Micro Prompts for Each Flow

- Split tasks into 1-3 message blocks.
- No essay dumps.
- Cursor codes better when the task is tiny.

20. Use `@` Context Loading for Scoped Memory

- Add files directly using `@filename.js`
- Cursor sees only what it needs.

21. Build a Debug Ritual

- Ask AI: “What broke? Why?”
- List 3 solutions. Choose one.
- Then fix it.

22. Don’t Wait to Automate the Boring Stuff

- Add custom scripts early (e.g. push to Vercel, run vitest).
- The 5 minutes you save adds up.

23. Ship Early, Then Fix It Live

- Use `dev.domain`, gated access, and logs.
- Let real usage shape your fixes.

24. You’re Not Building Google

- Vibe coding is about momentum.

- You’re not scaling to a billion users. Not yet.
- Focus on velocity, not perfection.

---

p.s. I’m putting together 100+ scoped prompts, debug flows, and component-level strategies in Playbook 002. If that sounds useful, keep an eye on your inbox.

Stay caffeinated. Lead the machines.