VIBE CODERS FIELD MANUAL

Written by: David Levine Bramante
For: Inexperienced but ambitious AI-assisted developers ("Vibe Coders")
Version: 1.0
Date: July 9, 2025
Purpose: To help you build websites and apps fast, using AI tools with structure, clarity, and confidence—without getting lost, overwhelmed, or burned out.

WHY THIS GUIDE EXISTS

If you're trying to launch a website, SaaS product, online platform, or even a personal portfolio—but you're not a "real" developer (yet), this system is for you.

I'm not a trained coder either. But I've launched real products with real revenue by using AI like Claude, GPT-4, and Cursor as my programming co-pilot. I built this guide to keep myself focused, organized, and shipping fast. Now you can use it too.

This is not a course. This is not theory. This is how I'm building actual businesses and income—from real estate tools to AI image platforms to content marketplaces—and helping others do the same.

WHO THIS IS FOR

You're a Vibe Coder if:

- You're using AI to help write code, build software, or launch products
- You're excited to move fast and learn by doing
- You want structure and sanity—not chaos, confusion, or half-built apps
- You're building for yourself or your business, not to pass a coding interview

You might be:

- A solo founder trying to launch a startup
- A creator building an app or tool around your brand
- A freelancer automating client work
- A real estate agent, artist, educator, or entrepreneur trying to scale up with AI

WHAT THIS GUIDE GIVES YOU

This setup guide gives you a clear, structured system to:

- Get started fast and stay focused
- Talk to AI coding tools the right way (no more "bad answers")
- Prevent common mistakes that waste time and money
- Keep your project clean, organized, and launch-ready
- Hand off work between tools, models, or teammates without confusion

This is your operating manual for building real software using AI, without getting lost or stuck.

THE VIBE CODING PHILOSOPHY

We build. We ship. AI is our most powerful collaborator.

Two Work Modes:

1. Promptsmithing (Think + Plan)

- Ask smart questions
- Design the architecture
- Create your database, endpoints, and layout

2. Codestreaming (Build + Ship)

- Work inside your local editor (VS Code, Cursor)
- Build one file at a time
- Ship fast with focus

The only metric that matters: Did you ship?

THE VIBE CHECK PROMPT

Before any coding session, copy/paste this into your AI assistant (Claude, GPT-4, Gemini, etc.):

Act as my expert pair programmer. Before we begin, you must read and understand these documents to get full context on my project:

- /.ai/ACTIVE_TASK.md – What we're doing now

- /.ai/AI_RULES.md and /docs/SECURITY.md – Guardrails to follow

- /docs/ARCHITECTURE.md, /docs/DB_SCHEMA.md, /docs/API_REFERENCE.md – How the system works

- /.ai/DECISIONS.md, /.ai/HANDOFF.md – History and handoffs

Let me know when you've absorbed this and I'll give you your first task.

THE 12-FILE SYSTEM (EXPLAINED)

Here's how your project folders should be set up. This structure keeps AI, code, docs, and your mental clarity in sync:

```
[project-name]/

├── .ai/                    # Your AI's private memory (DO NOT push to GitHub)

│   ├── AI_RULES.md           # What your AI must follow

│   ├── ACTIVE_TASK.md        # What you're working on right now
│   ├── DECISIONS.md          # Past choices that shouldn't change
│   └── HANDOFF.md            # Use when switching tools or teammates
├── docs/                     # Long-term project reference
(Git-tracked)
│   ├── PROJECT_SETUP_GUIDE.md
│   ├── ARCHITECTURE.md       # How your app is structured
│   ├── API_REFERENCE.md      # All backend routes explained
│   ├── DB_SCHEMA.md          # What tables you use and how they
relate
│   ├── DEPLOYMENT.md         # How to go live
│   ├── SECURITY.md           # How to protect secrets
│   └── TROUBLESHOOTING.md    # Fixes for common bugs
├── tasklog.md                # Day-by-day activity tracker
├── frontend/                 # Your React app
├── backend/                  # Your Express.js server
└── README.md                 # Project overview for GitHub
```

Important: Add this to your .gitignore file:

```
.ai/
.env
.env.local
```

DEV TOOLS & STACK (BEGINNER-FRIENDLY)

You don't need to be an expert. This is what we use to get real stuff built:

- **Frontend:** React 18, React Router 6
- **Backend:** Node.js, Express
- **Database:** PostgreSQL with UUIDs
- **AI Tools:** Claude (browser), Claude Code (Cursor), GPT-4, DeepSeek
- **Editor:** VS Code or Cursor
- **Git:** GitHub for version control
- **Deployment:** DigitalOcean, Vercel, or similar
- **Auth**, **Payments**, **Email:** Choose when needed

HOW TO TALK TO YOUR AI

Every time you ask AI for help, use this format:

Context: We're building [Project Name], a full-stack app using React, Node, and PostgreSQL.

Task: I need help fixing a bug in frontend/src/components/Signup.js

Constraints: Follow all project rules from /.ai/AI_RULES.md and use only one file at a time.

AI works best when you're clear, specific, and structured.

AI MODEL SWITCHING RULES

If you ever stop working and come back later (or switch tools), use the /.ai/HANDOFF.md file:

Last Session Summary

- Completed: [e.g. "Finished Login.js UI"]

- Working on: [e.g. "Connecting signup form to backend"]
- Next steps: [e.g. "Create POST /signup endpoint"]
- Blockers: [e.g. "Not sure if JWT token flow is correct"]
- Context: [Any unusual decisions or fixes]

LOCAL SETUP (FOR FIRST-TIMERS)

Install these tools first:

Check if Node.js is installed:

```
node -v
```

Check if PostgreSQL is installed:

```
psql --version
```

Clone and setup your project:

```
# Clone the project repository
git clone [REPO_URL]
cd [Project Name]


# Install frontend dependencies
npm install --prefix frontend


# Install backend dependencies
npm install --prefix backend
```

To run your project:

Open three terminal windows:

Terminal 1 - Frontend:

```
cd frontend

npm start
```

Terminal 2 - Backend:

```
cd backend

npm start
```

Terminal 3 - Database (optional):

```
psql "[YOUR_DATABASE_URL]"
```

COMMON BUGS & FIXES

Port already in use?

Find what's using the port:

```
lsof -i :5001
```

Kill the process (replace [PID] with the actual process ID):

```
kill -9 [PID]
```

JSX error?

Instead of this (which causes errors):

```
<><div>Hello</div></>
```

Use this:

```
<React.Fragment><div>Hello</div></React.Fragment>
```

Database connection not working?

Test your connection:

```
psql [your_connection_string] -c "SELECT 1"
```

More fixes available in /docs/TROUBLESHOOTING.md

SHIP IT: DEPLOYMENT FLOW

Step 1: Build the frontend

```
cd frontend
npm run build
```

Step 2: Create and push to staging branch

```
git checkout -b staging
git add .
git commit -m "Prepare for staging deployment"
git push origin staging
```

Step 3: Test on staging

Visit your staging URL and test all features

Step 4: Merge to main

```
git checkout main

git merge staging

git push origin main
```

See `/docs/DEPLOYMENT.md` for full deployment steps.

FINAL REMINDERS FOR VIBE CODERS

1. **Start small.** One feature at a time.
2. **Don't skip the Vibe Check Prompt**
3. **Never work on multiple files at once** unless you really know what you're doing
4. **Avoid tech debt early**—stay clean, stay lean
5. **If stuck, ask your AI**—but be specific and structured
6. **Keep shipping.** Every working feature builds momentum

ATTRIBUTION & LICENSE

Contact:
www.davidbramante.com
davidbramante@gmail.com
Mobile: (310) 906-5459