

Vibe coding prompt pack

DMILZE EdTech · Vibe Coding and AI Optimization · Weeks 1 to 7. Copy, then fill the [brackets]. Always read what AI builds before you accept it.

1. Build a one-page site (Week 1)

Paste into Cursor or Claude. Swap the brackets for your own details.

Build a one-page website for [business name], a [type of business] in [area, city]. Use Next.js and Tailwind.
Sections: hero with a clear headline and a WhatsApp button, what we offer (3 cards), prices, testimonials placeholder, contact with opening hours and a map link.
Style: [warm and friendly / clean and modern]. Mobile first.
Use placeholder text I can edit. Explain each file you create.

2. Turn a spec into steps (Week 2)

Here is my one-page spec: [paste spec].
Break it into small build steps that each take under 30 minutes.
For each step: what to build, which files change, and how I can check it works. Don't write any code yet.

3. Build one step at a time (Week 3)

We're on step [number]: [step]. Build only this step.
Keep the existing structure. After the change, tell me exactly what to click to check it works.

4. Understand the code (every week)

Explain this file to me like I'm new to coding. What does each part do? What would break if I deleted [line or function]?

5. Fix an error (Weeks 3 and 6)

I did: [steps]. I expected: [result]. Instead: [what happened].
The error message is: [paste the full error].
Explain the cause in plain words first, then suggest the smallest fix.

6. Add accounts and data (Week 4)

Add Supabase sign up, log in and log out to this app.
Each user should only see their own [notes / bookings / orders].
Use environment variables for keys and add them to .env.example.
List anything I need to set up in the Supabase dashboard.

7. Add an AI feature safely (Week 7)

Add a [summarise / chat / smart search] feature using the Claude API.
Call the API from a server route only, never from the browser.
Read the key from an environment variable. Limit each user to [20] requests a day and show a friendly message after that.

Golden rules

Small steps beat giant prompts. Commit to Git after every working step. If something breaks badly, roll back instead of patching patches. Never paste real customer data or secret keys into a chat.