# 🚀 Pam 2.0 – Codex Build Guide

This guide shows you **exactly** how to use Codex (or any AI code generator) to build Pam 2.0 from the scaffold. Follow the steps in order.

## 🔹 Step 1 – Setup

1. Unzip **pam2\_scaffold\_package.zip** into your local dev folder.
2. Create a new branch in GitHub:

* git checkout -b pam2.0

1. Start Docker (backend + Postgres):

* docker-compose up --build

1. Verify backend is live:

* curl http://localhost:8000/health
* Should return:
* {"status": "ok", "service": "pam-2.0-backend"}

## 🔹 Step 2 – Controlled Development with Codex

Always work **file by file**. Never ask Codex to change everything at once.

### ✅ Example Prompt to Codex:

“Open backend/main.py. Add an endpoint /expenses that accepts user\_id, amount, category and saves to Postgres using SQLAlchemy. Write tests in tests/test\_expenses.py.”

Run tests immediately:

pytest tests/

If green → commit. If red → debug with Codex.

## 🔹 Step 3 – Building Core Modules

Follow this sequence:

1. **Expenses (Wins)**
   * Add /expenses POST + GET endpoints.
   * Store in expenses table.
   * Test: add expense + retrieve expense.
2. **Trips (Wheels)**
   * Add /trips POST (log trip) + GET (get trips).
   * Store in trips table.
   * Test: log trip + retrieve trip summary.
3. **Community (Social)**
   * Add /posts POST (create post) + GET (feed).
   * Store in posts table.
   * Test: create post + retrieve feed.
4. **PAM AI Chat**
   * Expand /chat endpoint to call Gemini API.
   * Store conversations in messages table (add via schema migration).
   * Test: send message → get AI response → check DB log.

## 🔹 Step 4 – Keep It Modular

When Codex adds new logic: - Tell it: **“Create a new file”** (backend/expenses.py, backend/trips.py, etc.). - Keep main.py for **routing only**. - Example Codex instruction: > “Move expense logic into backend/expenses.py and import router into main.py.”

## 🔹 Step 5 – Continuous Testing

Run tests every time:

pytest tests/

Add new test files when you add new endpoints. Example: - tests/test\_trips.py - tests/test\_posts.py - tests/test\_chat.py

If Codex writes code without tests → always prompt: > “Write matching pytest tests for this endpoint in a new file.”

## 🔹 Step 6 – Staging Deployment

1. Push code to GitHub (pam2.0 branch).
2. Add GitHub Actions (CI/CD):
   * Run pytest.
   * Build Docker images.
   * Deploy to staging (Render backend, Netlify frontend).

## 🔹 Step 7 – Promote to Production

1. Test everything in staging with real Supabase.
2. If green → merge pam2.0 → main.
3. Deploy to production.

# ✅ Rules for Success

* **One file at a time.**
* **Run tests after each change.**
* **Keep code modular.**
* **Don’t let Codex overwrite multiple files blindly.**

Follow this and you’ll get a **clean, future-proof PAM 2.0 build**.