🚀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
   ```

3. Start Docker (backend + Postgres):

   ```
   docker-compose up --build
   ```

4. 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)**
2. Add `/expenses` POST + GET endpoints.
3. Store in `expenses` table.

4. Test: add expense + retrieve expense.

5. **Trips (Wheels)**

6. Add `/trips` POST (log trip) + GET (get trips).
7. Store in `trips` table.

8. Test: log trip + retrieve trip summary.

9. **Community (Social)**

10. Add `/posts` POST (create post) + GET (feed).
11. Store in `posts` table.

12. Test: create post + retrieve feed.

13. **PAM AI Chat**

14. Expand `/chat` endpoint to call Gemini API.
15. Store conversations in `messages` table (add via schema migration).
16. 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):
3. Run pytest.
4. Build Docker images.
5. 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**.