✅Pam 2.0 Build Recipe (Codex Step-by-Step)

Keep this checklist open. Follow it in order. Tick each box when done. Don't skip tests.

🔹 Setup

- [ ] Unzip `pam2_scaffold_package.zip` into project folder.
- [ ] Create branch: `git checkout -b pam2.0`
- [ ] Start containers: `docker-compose up --build`
- [ ] Test health: `curl http://localhost:8000/health` → ✅ `{"status": "ok"}`

🔹 Module 1: Expenses (Wins)

- [ ] Prompt Codex: *"Add* `/expenses` *POST + GET in* `backend/expenses.py` *using SQLAlchemy. Write tests in* `tests/test_expenses.py`*."*
- [ ] Run: `pytest tests/`
- [ ] Commit if ✅

🔹 Module 2: Trips (Wheels)

- [ ] Prompt Codex: *"Add* `/trips` *POST (log trip) + GET (summary) in* `backend/trips.py`*. Write tests in* `tests/test_trips.py`*."*
- [ ] Run: `pytest tests/`
- [ ] Commit if ✅

🔹 Module 3: Community (Social)

- [ ] Prompt Codex: *"Add* `/posts` *POST (create post) + GET (feed) in* `backend/posts.py`*. Write tests in* `tests/test_posts.py`*."*
- [ ] Run: `pytest tests/`
- [ ] Commit if ✅

🔹 Module 4: PAM AI Chat

- [ ] Prompt Codex: *"Expand* `/chat` *in* `backend/pam.py` *to call Gemini API. Store logs in* `messages` *table. Write tests in* `tests/test_chat.py` *(mock Gemini)."*
- [ ] Run: `pytest tests/`
- [ ] Commit if ✅

🔹 Integration

- [ ] Prompt Codex: *"Wire routers (`expenses`, `trips`, `posts`, `pam`) into `main.py`. Keep modular imports only."*
- [ ] Run: `pytest tests/`
- [ ] Commit if ✅

🔹 Staging Prep

- [ ] Push branch: `git push origin pam2.0`
- [ ] Setup GitHub Actions for pytest + build.
- [ ] Deploy staging (Render backend + Netlify frontend).

🔹 Final Merge

- [ ] Test staging manually (trips, expenses, posts, chat).
- [ ] If ✅→ merge `pam2.0` → `main`.
- [ ] Deploy production.

🔝 Rule of Thumb

- **One module at a time.**
- **Run tests after each Codex change.**
- **Commit only green builds.**

Follow this recipe → Pam 2.0 will be clean, modular, and production-ready.