Case Study: Building a Modern Banking Application with AI-Driven Development

Overview

As a Senior Full-Stack Software Engineer, I led the design and implementation of a production-grade banking application prototype (codename: Banksy). The project leveraged Next.js, FastAPI, PostgreSQL/SQLite, Drizzle ORM, Docker, Clerk authentication, Payload CMS, Vitest, and pnpm workspaces to deliver a modern, scalable, and secure developer-friendly platform.

Problem Context

The client required a secure, AI-driven banking prototype that would: 1) Support account creation, transactions, money transfers, and statements, 2) Provide a developer-flexible CMS with PostgreSQL for backend control, 3) Include a test suite with frontend and backend coverage, 4) Run reliably in Docker environments for both local dev and production, 5) Integrate role-based authentication with Clerk.

My Role & Contributions

- Established a pnpm monorepo with apps/frontend and apps/backend.
- Configured Next.js frontend with Tailwind CSS v4 and React 19.
- Set up FastAPI backend with async SQLAlchemy and Alembic for migrations.
- Integrated Vitest + Testing Library for frontend tests with Clerk and API mocks.
- Built Docker Compose files with dev/prod overrides.
- Implemented Makefile commands for consistent developer experience.
- Enhanced security by removing .env copying in Docker, replacing it with dynamic Makefile injection.
- Designed modern account management forms with validation, dynamic loading states, and error handling.

Key Technical Challenges & Solutions

- Async driver errors with SQLAlchemy → Configured correct async drivers and ORM bindings.
- Vitest failing to resolve @/ imports → Added resolve.alias to Vite config.
- Clerk auth breaking in tests → Created global mocks for Clerk in __mocks__/clerk.ts.
- pnpm frozen lockfile errors in Docker builds → Isolated lockfile per workspace and rebuilt.

- Windows/Unix line ending issues in Docker scripts → Normalized with LF endings and enforced via .gitattributes.

Results

- Delivered a fully containerized monorepo banking app prototype.
- Achieved working frontend/backend integration with Clerk authentication.
- Established test coverage for account management with mock APIs.
- Produced a repeatable, secure dev environment with Makefile automation.
- Documented project setup and workflows for onboarding and deployment.

Professional Takeaways

- Hands-on experience with Next.js, FastAPI, pnpm, and Docker.
- Designed scalable monorepo architectures with microservice-style separation.
- Developed testing strategies for React in complex auth/data environments.
- Enforced security best practices in containerized environments.
- Led project setup from architecture through implementation, testing, and deployment.