Project Reflection: Kurve Kiosk Intern Task - PAVAN BARGOTTA

This project was a valuable exercise in building a full-stack customer management application, featuring a Python/FastAPI backend and a React/TypeScript frontend, with data persisted in a MySQL database.

Familiar Ground:

My recent seven-month experience with Legal-Pythia on a web application meant I was already well-acquainted with much of the core stack.

Backend: I was already extremely familiar with Python, FastAPI (and Uvicorn). Furthermore, as a result of working closely with the technical lead at Legal-Pythia, I learnt the importance of robust validation and error handling, so I easily transferred my skills of leveraging Pydantic to help.

Frontend: Similar to the backend, I used React, TypeScript, and Tailwind CSS in my previous project. This made developing with React's component model, TypeScript's type safety and Tailwind's styling very comfortable, which allowed me to efficiently structure the UI and implement features. Connecting both the front and backend (via the api.services.ts) was also comfortable, as I took a similar approach to how I had done it previously and was able to reuse helper functions I had made in the past with little modifications.

New Challenges & Learning:

Going into this task, I had a one-week deadline and knew I wouldn't have much free time during that period. Despite this, I still wanted to learn at least one part of the tech stack you use at Kurve, so I decided to focus on the database. I used MySQL with SQLAlchemy, which presented several challenges - particularly around the asynchronous implementation.

SQLAlchemy & ORM Concepts: While databases weren't entirely new conceptually, working directly with SQLAlchemy was. Understanding how to define models, use the Engine and AsyncSession, and translate CRUD operations into ORM queries required a lot of focused learning in (what I found) very confusing documentation - to remedy this, I used lots of external resources (like video tutorials etc).

Asynchronous Database Calls: Understanding how async/await works with AsyncSession and create_async_engine was crucial. Managing asynchronous database sessions as FastAPI dependencies (async def get_db(): ... yield session) was a practical application of combining these technologies. Ensuring that all database calls were properly awaited to prevent blocking operations was a constant point of attention.

Practical Database Considerations: Implementing unique constraints (like for email) and then handling them by, for example, pre-fetching existing emails before populating test data, offered practical insights into real-world database interactions.

Final Reflection

If I had more time during the week, I would have enjoyed exploring PHP for this task or adding extra features to the application, such as filtering or search functionality. That said, I'm pleased with what I have been able to achieve in a short period, particularly in developing my understanding of MySQL and SQLAlchemy. I hope you're equally satisfied with the outcome, and I would be excited to continue contributing and growing with your team!