Smart Financial Coach Design Documentation

1. Design Choices:

The goal of this project is to give intelligent and useful insights into the user's financial situation. Because many target users have a novice financial background, the interface was designed to be simple and easy to navigate with modern components.

- **Onboarding Screen**: A straightforward screen with disclaimers and instructions on how to use the application. Larger buttons and cards are helpful to make it obvious to the user where to go to start the flow of the application.
- **Dashboard**: After uploading their data, users are redirected to the dashboard, which provides an overview of all financial analyses and information.

    Sections:

    - **Dashboard Header**: Displays a title indicating the user is on the dashboard, along with buttons to add more transactions (green) or end the session (red).
    - **Totals Section:** Gives the user useful summary totals, providing concrete numbers about their account and an overview of their all-time balance.
    - **Spending Trends**: An impactful feature for visual users, allowing them to switch between graph and bar chart views to see where their money is being spent.
    - **My Transactions**: A table with pagination provides a streamlined way to review financial data chronologically.
    - **Insights:** In order to save space, each insight opens up into a modal to effectively display information without having to navigate out of the main screen. Users can get feedback on their transactions and take action by:
        - Overall trends
        - AI Feedback
        - Nearby Affordable alternatives

2. Technology Stack:

Frontend: JavaScript, Tailwind, React, Redux

- React provides the application with a component-based structure that supports scalability. Redux is used to improve performance and manage data that persists across the application. It also defines reusable API calls.
- Tailwind provides modern-looking components with a smooth user experience.

Backend: Python, FastAPI

- FastAPI is a light-weight backend framework. It's helpful when creating APIs and provides tools to test/visualize them. It also provides additionally security when interacting with the database to prevent web attacks like SQL Injection.
- Python is powerful for running AI/ML tasks coupled with the tools/modules it provides.

Database: SqLite

- A great light-weight solution to store persistent data but allow the user to have total control.

Technology Stack(continued):

AI/ML: Pandas, Prophet, Scikit-learn

- Pandas is used for advanced data manipulation which can then later be used to run AI/ML tasks. Data frames are created to properly establish properties for transactions.
- Prophet and Sckit-learn: These libraries helped build Isolation Tree models which detect anomalies based on unusual patterns. Scikit-learn also provides pipeline logic to transform data used in the training process. This is especially useful when dealing with non-numeric values such as category data.

External APIs: OpenAI, Overpass

- OpenAI provides a suite of LLMs that you can use for prompt engineering, which is especially useful for obtaining human-like feedback.
- Overpass is a geolocation API that gives query abilities to discover places.

3. Future Enhancements:

Technology Enhancements:

- Migrate to RTK Query within Redux. It offers enhanced state management and automatic caching, improving performance when executing costly operations.
- Transition to a Django backend to support complex data modeling and deliver stronger security solutions.
- Make tables and cards more modular to improve scalability as new features are added. Also include additional filters, such as viewing data from the last 1, 2, or 3 months.
- Migrate OpenAI API calls to on-prem LLMs using Ollama to reduce costs. Due to memory limitations in my local setup, I wasn't able to do this.
- Improve the risk model by incorporating training data that includes known fraudulent cases. This will help the system better understand risk patterns across different types of users. Additionally, adding foreign transactions in the rules to detect risk will be a helpful flag.
- Store the results of ML calls to boost performance and mitigate repetitive calls.

General Feature Enhancements:

- Develop a more comprehensive view of the user's financial background, including their credit score and creditworthiness, to enable more accurate future forecasting.
- Take things a step further by recommending financial products, such as loans and credit cards, based on the user's financial profile.
- Migrate to a more robust api like Plaid to securely connect to bank accounts to access transaction data.
- When recommending affordable places, also consider reviews to increase user satisfaction.
- Given next month's predictions, we can help budget for the future using interactive calculators.