SP-26 PURPLE BUDGET APP

THE FINAL REPORT

Course: Senior Project Section 01 CS 4850

Semester: Spring 2024

Professor Perry

Date: 4/6/2024

Team Members:

Ann Nguyen

Cameron Lowry

Alexandra Clark

Number of lines in the project: about 1800 lines

Number of components in the project: about 30-50 components

Website: https://sp-26-purple-budget-app.github.io/

Github: https://github.com/SP-26-PURPLE-BUDGET-APP

Over this Spring Semester we have been designing and implementing our Senior project. For this project we decided we would build a budgeting app by the name of MoneyFlow. This app is designed to help users manage their income by tacking daily expenses, categorizing them and providing insight. This will allow users to make better financial decisions and avoid overspending. The app was designed using JavaScript through React Native and integrates several APIs for backend communication with a database and with banks.

As with all projects MoneyFlow began simply as an idea for a simple budgeting app that would allow users to keep better track of their spending. This came with our first project goals which included expense tracking, visualization of expenses and privacy. To accomplish these goals, we understood that there would be a variety of different constraints we would have to deal with. One of the biggest constraints we had to deal with was budget limitations as we had no funding for this project. Budget limitations affected a large part of our project as we had to use free resources in order to build MoneyFlow. Many of our design choices such as using react native for our implementation and Firestore database would stem from this limitation. Another large limitation we would face would be the time constraint that all senior projects face. This means we would only have one semester to design and implement this app, meaning some features would have to be not fully realized or placed on the back burner. Other smaller constraints that we had to deal with were that the app would have to run on both IOS and Android and that the app would need to be connected to the internet for some of its features to work. During this planning phase we also had to figure out who our target audience was. After some discussion, we decided to target people between the ages 18-35 with stable internet access and a cellphone. This allowed us to make some assumptions about who these people are such as that they had a moderate proficiency with their device and that their financial literacy could vary heavily. This second assumption would lead us to provide clear explanations in the app to help those with low financial literacy better understand the app.

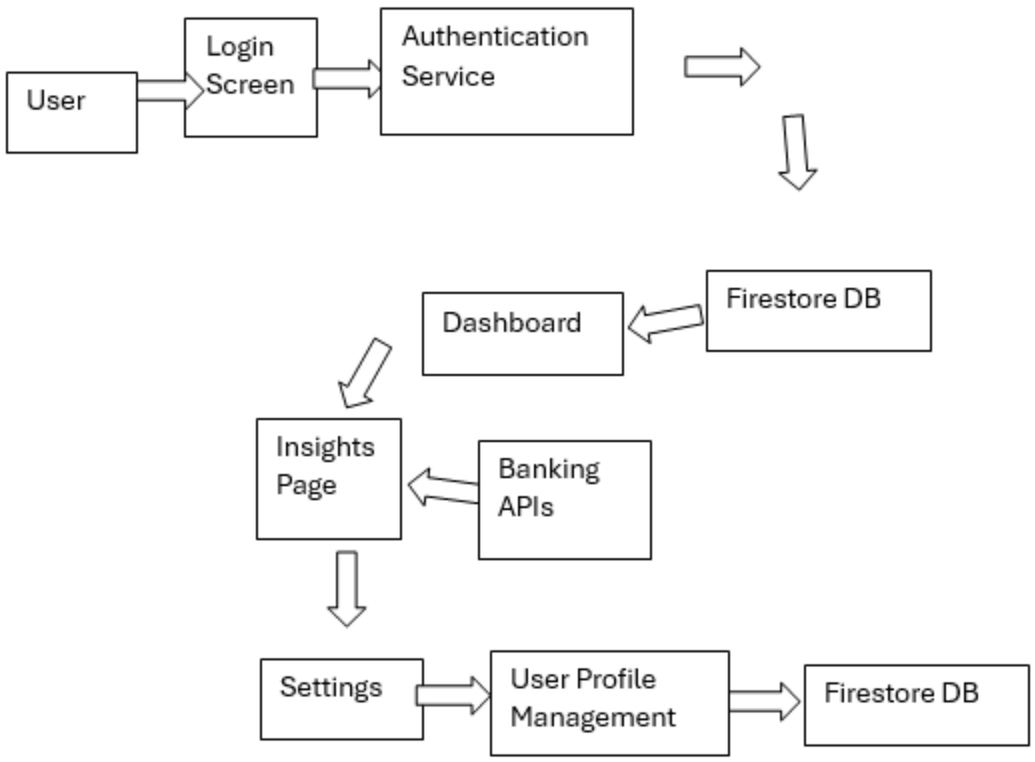
Based on our goals, assumptions and restraints we decided on some fundamental requirements that we would have to meet. The first of these was a login screen with username and password functionality. This would both protect the user's information with a password and allow for each individual user to have their own account. In the design we wanted to make it quick and easy for users to see their data, so we also decided to add a Homepage that would provide an overview of the user's information. In addition to this we had the idea of a page that would show more information such as the relationship between income and expenses. Also, we had the idea of a page that would show more information like the relationship between income and expenses. For MoneyFlow it was also required that a user would be able to easily input and edit their expenses and income. For this we designed a balance sheet that would allow the user to see their total budget and input new expenses as they are made. We also had some non-functional requirements that we were shooting for but didn't know if we would be able to include. We split these non-functional requirements into four different categories, Security, Capacity, Usability and Others. The security category held all the ideas we wanted to add to make the app more secure for users but were not entirely necessary for the app to function. Capacity spanned issues such as performance and scalability across the app when it was experiencing heavy traffic. Usability was a category we put ideas into that would help users more easily navigate the app and understand their financial information. Finally, the other category held ideas that didn't necessarily fit in with the other categories such as improvements to code readability and app communication.

To implement these ideas, we decided to use several different programs and APIs. The first program we decided on using was React Native. As we were discussing different programming languages and programs to help us build MoneyFlow the idea of React Native was thrown out. React Native had a lot of things going for it such as being free and providing the framework for app integration for both IOS and Android. One of our group members also had prior experience using this program so we decided to use this to build our app. When first using React we made a few mistakes as we were learning how to use the app better. However, these were overcome rather quickly and helped us get to implementing our design. Later into development as we were creating the database we had to agree upon which database system to use. We knew that building our own database would simply take too long and more effort for less results than using a premade cloud database. After some discussion within our group and with other project groups we decided it would be best to use Firebase. Firebase has access to Firestore which is a NoSQL cloud database built by google that is used for web, mobile and server development. Using Firebase and Firestore we were able to design a database to hold all the user's information. This also came with its own data transfer security, taking a lot of work off our shoulders.

Once we decided on the programs we would use, we began to design the UI and implement a test version. One of the first things that we did was build a mock-up of the UI that would later be added into MoneyFlow. This mockup was made using various images pasted together to make a "fake image" of what our UI would look like. At first this design included only a few pictures and certain icons didn't quite fit the rest of the aesthetic. Over about a week and a half we went through several iterations of this UI with icons being changed and different color schemes applied. During this time, we also discussed which font would be best to use in this app. We wanted one that would be easy to read and look sleek and professional. Once the UI was designed, we added functionality by implementing it inside of React Native. Using React Native we added buttons and swipe features to allow for easy navigation of the app. Finally, we implemented the database using Firebase. This took some time to get working as none of us has ever used firebase or designed a database. After creating each of the core features, we went through testing each section. For testing we mainly tested the app by running through it ourselves as we don't have the funding to hire a team of testers to thoroughly test every part of the application.

System Overview

The software is a budget management application designed to help users track and manage their finances effectively. The app provides various features to assist users in budget and expenses tracking. It is developed by React Native, allowing users to access the app both on Android and iOS:



Conceptualization:

MoneyFlow app creates a tool that provides better financial management. Our goal was to build an app that not only functions as a budget tracker but also educates its users on financial literacy.

Functionalities:

- Expenses tracking allows users to track their expenses by linking to the bank account or enter by their own.
- Income tracking: users can also track their income sources and view their total income over a specified period.
- User authentication and Security: to ensure the security of user data. The app includes user authentication features such as login/logout functionality and data encryption.

Tech Specifications:

- Programming Language and Framework: JavaScript with React Native was chosen because of its cross-platform mobile application development.
- User Interface: A mobile app interface using React Native.
- Data Management: Firestore is used for the database.
- Communication: The application includes API integration for backend communication and banking data synchronization
- Debugging: Uses react native's debugging tools and libraries to identify and fix the issue during development.

Design approach:

- The system is divided into two subsystems: budget management and user management subsystem. Each subsystem is responsible for handling the specific aspect of the application functionality.
- The overall system focuses on a user-friendly interface, with intuitive navigation and clear visualization of financial data. The app design also emphasizes security to protect user database and profile.
- The system is structured to allow scalability and flexibility. New features can be added easily, and an existing feature can be modified.
- Overall, the app is designed to provide users with comprehensive tools for managing their finances effectively, with a focus on security, usability and scalability.

Design and Development

- Constraints: Our project faced some constraints, primarily budget limitations and the tight timeframe typical of a semester-long project. These constraints influenced our choice to use only free and open-source tools and technologies.

Version Control:

We used Git as our version control system, with the repository hosted on GitHub. This setup allowed for a way to manage the project's development lifecycle.

- Branch Strategy: We used a streamlined branch strategy to keep our development process organized and manageable. This approach allowed us to focus on developing features without the overhead of managing multiple branches.
- Commit Protocol: The commit process was kept straightforward, allowing us to maintain a steady pace of development.

Testing

- Local Testing: All new features and changes were tested locally by the coder to ensure that they worked correctly before being committed to the development branch.
- Unit Testing: Tested the individual components to make sure there were no issues.
- Integration Testing: Ensured that different parts of the app worked together without errors.

Conclusion

In conclusion, the development of the MoneyFlow app during our senior project has been a rewarding experience. We successfully created a functional and secure budgeting tool that addresses the financial management needs of young adults. Throughout this project, we improved our understanding of project management and user-centered design and our technical skills in React Native and API integration. This project has given us practical experience and insights into the complexities of building real-world applications.

Appendix

Appendix A: PROJECT PLAN

Appendix B: SOFTWARE REQUIREMENTS SPECIFICATION

Appendix C: SOFTWARE DESIGN DOCUMENT