SP26 – Red Budget App

CS 4850 – Section 03 – Fall 2024

October 3rd

Josh Wilmath Developer	Garrett Heffner Developer	Avi Reale Documentation

Project Team

Name	Role	Cell Phone / Alt Email
Josh Wilmath	Developer	630-850-0565 jrwilmath@gmail.com
Garrett Heffner	Developer	678.894.6317 garretttheffner27@gmail.com
Avi Reale	Documentation	678.378.1507 akreale2000@gmail.com
Sharon Perry	Project Owner or Advisor	770.329.3895 Sperry46@kennesaw.edu

Total Man Hours: 258 Hours

Lines of Code: 1,015

Number of Project Components/Technologies: 14

Website: https://sp26groupagj.github.io/SP26RedBudget.github.io/

Github: https://github.com/SP26GroupAGJ/SP26RedBudgetApp

Table of Contents

1 Introduction

1.1 Overview

This Mobile Budgeting app is being developed to provide users with an easy and effective way to manage their finances. The app will include features for tracking income, expenses, and savings, setting financial goals, and analyzing spending habits through detailed visual aids like graphs and charts. The project's scope includes designing an interface that is easy to use, implementing data storage that is secure, and making sure that it's compatible with both iOS and Android platforms. This project's objective is to deliver an efficient and reliable solution that enables users to make informed financial decisions easily.

1.2 Project Goals

The main objective of this project is to develop an easy-to-use mobile application that simplifies the way users manage their finances. Along with the development of the application itself, we have goals to ensure secure integration with financial accounts using Plaid and to create cross-platform compatibility with iOS and Android.

1.3 Definitions and Acronyms

- **API: Application Programming Interface** - A set of protocols, tools, and definitions that allows different software applications to communicate with each other.
- **Plaid**: A financial services API that connects users' bank accounts to the app securely, allowing access to financial data.
- **Figma**: A design tool used for creating user interface mockups and prototypes.
- **SDK**: Software Development Kit - A collection of software development tools that allows the creation of applications for a specific platform.
- **UI/UX**: User Interface/User Experience - UI refers to the design of the application's interface, while UX refers to the overall experience a user has while interacting with the app.
- **Flutter**: An open-source UI software development toolkit created by Google for building natively compiled applications for mobile, web, and desktop from a single codebase.
- **Dart**: A programming language optimized for building user interfaces, primarily used with the Flutter SDK.

1.4 Assumptions

It is assumed that the users have some basic knowledge of financial terms and smartphone usage.

It is assumed that there will be continuous availability of external APIs like Plaid for seamless financial account integration.

2 Design Constraints

2.1 Environment

The mobile budgeting app will operate on both iOS and Android platforms which will ensure a wide range of accessibility. The app will require an internet connection to ensure real-time data synchronization and integration with Plaid.com for secure financial data access. Our backend services will include user authentication and database management to create a reliable and scalable environment.

2.2 User Characteristics

The end users of this mobile budgeting app will be individuals seeking an easy and straightforward way to manage their finances. The users will likely range from college students to even working professionals. These end-user types are expected to have some sort of experience with mobile apps and a decent familiarity with basic budgeting concepts. Since we are dealing with a range of novices to some financial knowledge, we aim to create a design that prioritizes ease of use and simple navigation to cater to a broad audience of users.

2.3 System

At a high level, our application will consist of three main components which are the mobile application, the backend, and the database. These will all be integrated together using external services. The mobile application will serve as our interface for end users. This will communicate with our backend server to synchronize data, retrieve financial insights, and authenticate users. The backend will also handle requests and interface with Plaid API to access user data securely. Lastly, our database is responsible for storing user data. We aim to have the database support real-time synchronization with our mobile app so users can access updated financial information. Overall, our system is designed to offer a reliable and stress-free user experience.

3 Requirements

3.1 Functional Requirements

2.0 Login/Launch Page

- R 1.1 The "Login" button redirects the user to the home page if the correct password is input into the text box.
- R 1.2 The "Register" button redirects the user to the create account page.
- R 2.2 The user can log in to a local account using a secure password.

3.0 Register - Create Account Page

- R 3.1 The user must provide an email address (which will serve as their username).
- R 3.2 The user must create a secure password.

4.0 Home Page

- R 4.1 The app displays an overview of the user's financial status, including total balance, recent transactions, and budget tracking overview in a scrolling menu.
- R 4.2 The home page provides navigation options to access different sections of the app: transactions, goals, and settings.
- R 4.3 The app offers quick actions from the home page, such as adding a new transaction or updating financial goals.

5.0 Expense and Income Tracking

- R 5.1 The user can log a new expense and income by entering the amount and date.
- R 5.2 The user can categorize expenses and income under predefined or custom categories.
- R 5.3 The user can view a summary of expenses and income by category, date, or amount.

6.0 Budgeting Tools

- R 7.1 The user can set monthly budgets for specific user made categories
- R 7.2 The app tracks spending against the budget and notifies the user if they are approaching or exceeding their budget.
- R 7.3 The user can adjust total spending amounts, budget time intervals, and deleting budgets in a screen allowing for the editing of existing user created budgets.

7.0 Financial Visualization

- R 8.1 The app displays spending and income patterns through visual aids such as pie charts, bar graphs, and line charts.
- R 8.2 A dropdown date menu allows users to customize the time range for graphs (e.g., daily, weekly, monthly).

8.0 Notifications

- R 9.1 The app sends transaction alerts for significant spending or income events.
- R 9.2 The app sends reminders to update or review financial goals once each week by default unless configured otherwise by the user.
- R 9.3 The user can customize notification settings in the settings screen (e.g., enable/disable specific alerts).

9.0 Plaid API Page

- R 10.1 A screen displays the user's current Plaid API key if configured or prompts the user to connect a bank account if not set up.
- R 10.2 The user can select the banking organization from a dropdown menu of options.
- R 10.3 The user can open a link to Plaid to complete the process of connecting to their bank account.
- R 10.4 The user can refresh the API call to Plaid to receive updated transaction information by pressing a sync button.

10.0 Settings Page

- R 10.1 The user can update personal information such as name, email, and phone number.
- R 10.2 The user can change their password after completing and email verification
- R 10.3 The user can manage notification preferences.
- R 10.4 The user can log out of the app, returning the user to the login screen.

3.2 Non-Functional Requirements

3.2.1 Security

Security is important when it comes to our mobile application since it handles sensitive financial data. Security measures must be implemented to ensure that all user data is protected from unauthorized access and potential breaches. We will use features like two-factor authentication as a layer of security.

*** more about password storage and possible DB connections and security

3.2.2 Capacity

The amount of people using the application could turn into a problem with connected applications, such as Plaid, generating errors if API limits are reached. However, for our scope, Plaid's limits should be more than enough to handle a small number of users. Also, the choice of Flutter's ecosystem which uses an efficient layered architecture which operates close to the native hardware, allowing for efficient performance.

3.2.3 Usability

The useability of our application is incredibly important since it is intended for a wide range of users. The app user interface must be easy to navigate and allow users to quickly access their financial data. Our UI will utilize screens with minimal buttons and elements to cater to a mobile audience and ensure that users can reliably navigate through the app.

3.2.4 Other

Another important factor for our application will be its maintainability. Ideally, the application's back-end code will be designed to maximize readability so that any future updates are bogged down by difficult to read code. Reducing the initial level of tech debt will help users enjoy a stream of updates that don't take an unreasonable amount of time to deliver desired features.

3.3 External Interface Requirements

3.3.1 User Interface Requirements

The mobile app will have a user-friendly interface that's designed for both iOS and Android devices. The interface will be made simple and easy to navigate. It will include features like tracking income, expenses, and savings. It will also be designed to ensure responsiveness and clear visuals like charts and graphs.

3.3.2 Hardware Interface Requirements

The application will be designed to work with smartphones and tablets that support iOS and Android operating systems. This will be the only specialized hardware requirement for our application. Other requirements are more standard like internal storage and internet connectivity.

3.3.3 Software Interface Requirements

The Mobile Budgeting app will integrate with external APIs, such as Plaid. This will securely connect user accounts with their financial accounts. The app will require our team to install the latest version of Flutter and Dart SDKs for development and maintenance. The software will also need to support secure communication protocols for data transfer between the app and the financial institutions through APIs.

3.3.4 Communication Interface Requirements

The application will rely on internet connection for certain features. Some of these features include integrating financial account data through Plaid and accessing real time data.

4 Design

4.1 UI Design

The user interface is responsible for all user components, which includes the login and registration page, the home screen, and various other financial tools. The UI design is to ensure the app allows easy navigation and access to all the features. Mocked-up screens were used to get a feel for how the user will interact with the control flow, which helped decide to combine the launch and home page and balance functionality with ease of use on the home screen. We plan to keep the design simple and minimalistic to focus on functionality and keep the UX intuitive.

The app will interface with Plaid APIs to access users' financial data securely, and the design will be focused on providing an easy-to-use interface that will include visual aids like charts and graphs. The FL Chart library supports visual elements such as line charts, bar graphs, and pie charts, which will be used to display the user's budget and financial tracking data.

4.2 System Design

The mobile budgeting app's system design is classified into several key modules. Each module will serve its own purpose. The user interface is responsible for all user components which includes the login and registration page, the home screen, and various other financial tools. It's designed to ensure the app allows easy navigation and access to all the features.

There will also be a synchronization module that manages the integration of financial data through the Plaid API. This module will include the logic required to make API calls. It will also handle data updates and provide users with a manual sync button that will allow them to refresh their financial data as needed.

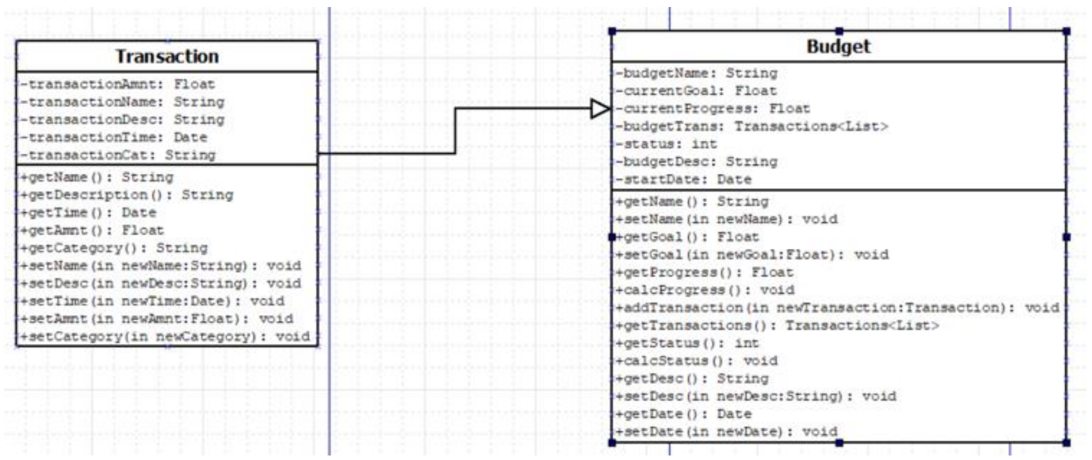


Figure 2: *UML Diagram of Transaction and Budget classes to demonstrate initial plans for their composition.*

The primary classes that will be utilized within the application are classes for budgets and transactions. Transactions feature different variables for the attributes each transaction will need such as an amount, name, description, time, and category. These variables are private, so getter and setter functions are used to access and modify these variables.

Transaction objects are going to be utilized further by any budget classes, as each budget class will contain a list of any relevant transactions affecting the specific budget. Other variables within budget objects will be the name, goal, progress, status, startDate, and description. Similarly, to the transaction class these variables are private and require getters and setters for most of these variables. Notable exceptions are for progress and status which will run a calculation to determine the value of these variables before returning the variable

5 Development

5.1 System Architecture and Backend

The system architecture for the mobile budgeting app will be designed with a multi-tiered approach. The architecture will follow the Model View Controller (MVC) approach with each screen being a view connecting to a controller using Flutter to navigate with buttons. The models will define the data structures with the Budget and Transaction classes driving the data displayed on the screen.

The mobile app will be developed using Flutter and Dart to ensure iOS and Android platform compatibility. The “mobile client” application will serve as the user interface. It will provide the end user tools to manage their finances. They will be able to track expenses, income, and savings. The app will interface with Plaid APIs to access users’ financial data securely. The design will be focused on providing an easy-to-use interface that will include visual aids like charts and graphs.

We plan to store the budgets and transactions using classes to simplify storage and storing instances of those classes in JSON files allowing for ease of exporting and importing for possible future functionality. Architecturally, this will serve as the model manipulated by the Dart code and displayed on Budget, Transactions, and Visuals screens.

5.2 Software Architecture

Main Architecture: The main architecture of our application will be using Flutter and its programming language Dart. Flutter’s layered architecture allows for low-level hardware control for snappy performance while also using abstraction for a straightforward development process. This approach sacrifices using a more popular programming language such as C# but supports an SDK that allows native compatibility for running our application on iOS and android devices.

Financial Integration: Integrating information from financial institutions for transaction information and necessary budgetary information will be handled by the Plaid API. This will require managing the API key from Plaid which will be done in the following process:

1. Call to plaid to get link_token
2. Open link for user to authenticate/connect to bank
3. onSuccess callback returns public_token
4. exchange public_token for permeant access_token and item_id
5. Store access_token permanently in app.

   https://plaid.com/docs/quickstart/#how-it-works

Plaid was chosen as our method of accessing financial information because it offers superior convenience for users in the ability to access banking information rather than manually inputting information. However, some notable drawbacks of adding complexity and reliance on an external system and may limit scalability due to Plaid API rate limits here.

Rendering Graphs/Charts: Using FL Chart, free and open-source graphing library for Flutter. Free for personal and commercial use. Cross platform support across iOS, Android, Web. Supports line charts, bar graphs, pie charts, and more.
https://fluttergems.dev/packages/fl_chart/

https://flchart.dev/

Storing Transaction/Income data: Transactions will be represented by a class with attributes for value, name, category, date, and income/expense.

Budget tracking: Budgets are also represented in a class with attributes for the name, goal, progress, status, relevant transactions, description, and start date.

Storing Plaid API keys: Use Plaid, a user will connect their bank through Plaid and return an API key that can be used to get later updates. Our application will need to store the API key and allow the user to delete or change the API key.

Control structure:

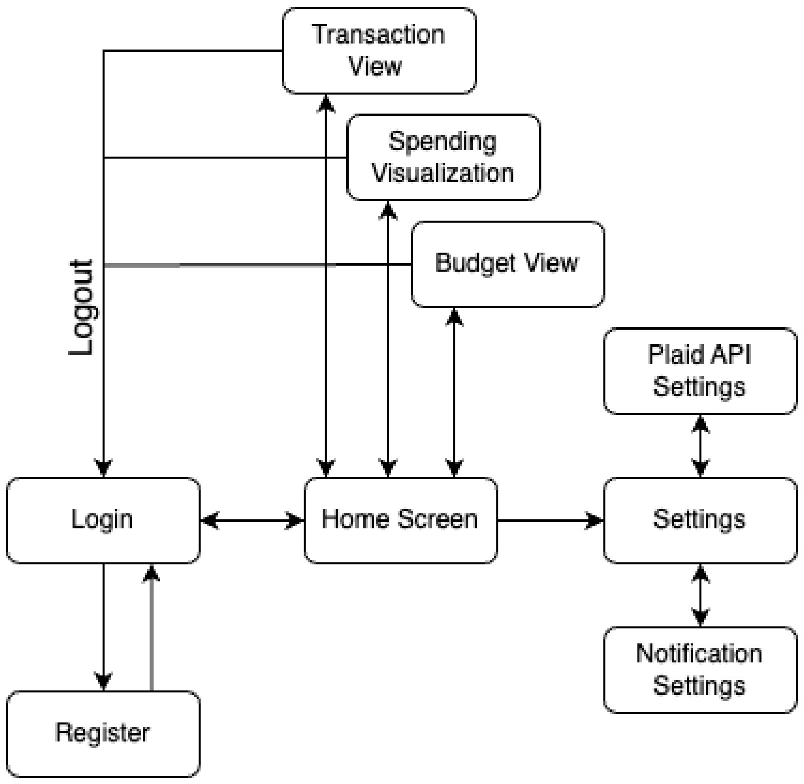


Figure 1: *A user flow diagram showing the paths the user could take within the application*

User Login: Users access the main functionality of the app by logging in with a password. Passwords are stored locally on the device with encryption. Currently we only plan to have one user for each app, so only a password is used to access the app.

Future plans: Outside of the current scope of the project, we had ideas to create an online service where users can log in, authenticate with 2FA, and reset their password with an email. Multiple users could log in, switching from user to user on one device. We also had the idea that users could possibly log into multiple bank accounts and import/export transaction records which would be explored more in the future.

5.3 Information Flow of Mobile App

The information flow of the app is designed to be seamless and intuitive for users. The process starts when a user logs into the app by providing their credentials for authentication. Once authenticated, the app connects to the user's financial accounts through Plaid. Plaid is a service that allows us to pull important data like transaction history, account balances, and more. This connection happens securely to ensure privacy and data protection.

After the app successfully retrieves the financial data, it processes the information to organize it into categories, calculate spending, and update the user's financial overview in real time. The data is then displayed in a simple format on the screen. Users can view their spending, track budgets, and make financial decisions based on the most up-to-date information. This flow ensures a smooth user experience. Users can interact with their financial data, all while maintaining the integrity and privacy of their personal information.

5.4 Analysis and Limitations of Technology

The technologies we've chosen for the app have worked well, but they come with their own set of challenges. Flutter allows us to build a single app for both iOS and Android. However, it isn't perfect. It can struggle with complex tasks that require high performance, and since the framework is still evolving, some third-party packages we rely on aren't always fully supported. When it comes to Plaid, which we use to pull financial data, it's generally reliable, but it can have downtime or issues pulling data from smaller banks. We also have to be mindful of Plaid's strict API rate limits, as exceeding them can disrupt the app. Device performance can vary, especially on older models, which may struggle with processing large amounts of data or handling multiple tasks. And of course, security is always a concern when dealing with sensitive financial data. While we've implemented solid encryption, staying ahead of potential vulnerabilities is an ongoing effort.

6 Testing

6.1 Test Plan

Our test plan is designed to verify that all app features work correctly. We'll start with functional testing, checking that each core feature (including the login, Plaid integration, expense tracking, and budget updates) works smoothly without issues. Then we'll conduct usability testing to ensure users can easily navigate and access key functions. Security testing will focus on data protection and making sure user information is safe during storage and transfer. We'll do performance testing. We'll evaluate how the app performs on different devices to confirm it loads quickly and operates smoothly. Finally, we'll do compatibility testing to check that the app works consistently on both iOS and Android.

6.2 Test Report

Requirement	Pass	Fail	Severity	Comments
Login/Launch Page				
"Login" button redirects to home page on correct password input	X		Major	
"Register" button redirects to create account page	X		Major	Button labeled create account instead of register
User can log in with secure password	X		Major	More of a non-functional requirement for security, more

				security measures could be implemented later
Register - Create Account Page				
User provides an email address as username	X		Medium	
User creates a secure password	X		Minor	*Not listed are requirements to create a new user profile that can be logged in with
Home Page				
App displays financial overview (total balance, recent transactions, budget)	X		Major	Total account balance not currently listed
Navigation options to transactions, goals, and settings	X		Major	Navigation options are labeled differently
Quick actions to add transaction or update goals		X	Medium	Buttons to add new transactions are currently not functional
Expense and Income Tracking				
Log new expense/income with amount and date		X	Medium	New transactions can currently only be added via editing the JSON file
Categorize expenses/income (predefined or custom)		X	Medium	No custom category support
View summary of expenses/income by category, date, or amount	X		Major	Changing the sorting is currently not functional but transactions are broken down by category
Budgeting Tools				
Set monthly budgets for categories	X		Major	
App notifies user when approaching/exceeding budget		X	Minor	
Adjust spending, budget intervals, and delete budgets		X	Medium	Budget information can only be edited in a JSON file
Financial Visualization				

Display spending/income patterns in charts	X		Major	
Dropdown to customize graph date range		X	Medium	Transactions list dropdown is not functional
Notifications				
Transaction alerts for significant spending/income		X	Minor	
Weekly reminders for goal updates by default		X	Minor	
Customizable notification settings		X	Minor	Notification settings button is not functional
Plaid API Page				
Display Plaid API key or prompt to connect bank		X	Major	
Dropdown to select banking organization		X	Minor	Might not be necessary, Plaid should handle this
Link to Plaid to connect bank		X	Major	
Sync button for updated transaction info		X	Medium	Sync button currently links to API setting screen
Settings Page				
Update personal information (name, email, phone)		X	Minor	
Change password with email verification		X	Minor	
Manage notification preferences		X	Minor	
Logout returns to login screen	X		Major	

Severity Key:

- **Minor:** These are small issues that don't really get in the way of using the app. They might make things look a little off or feel less polished, but they don't stop anything important from working
- **Medium:** These are problems that cause some inconvenience or make the app harder to use, but they aren't dealbreakers. There's usually a way to work around them, and they don't affect the app's most important features
- **Major:** These are big problems that really hurt the app's ability to do what it's supposed to do. They can make key features unusable or stop users from completing essential tasks, like logging in or navigating the app. Fixing these is a top priority because they impact the overall experience in a big way

7 Version Control

7.1 GitHub Repository

The main method of version control we will be utilizing is a GitHub repository. This repository will enable us to maintain the ability for different versions of the app to be managed by each member of the team. Furthermore, this will allow for past versions to be accessed in the event of an update causing a catastrophic bug that needs to be immediately reverted.

7.2 Version Control Plan

We're using Git for version control to keep the app's development organized and make sure every change is tracked. Each update, fix, or new feature is recorded, so it makes it easier for us to go back to an earlier version if anything breaks.

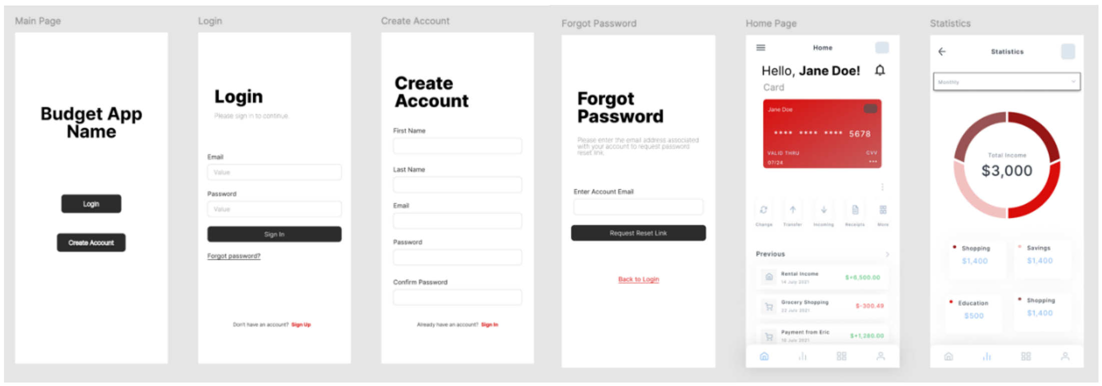
8 Conclusion

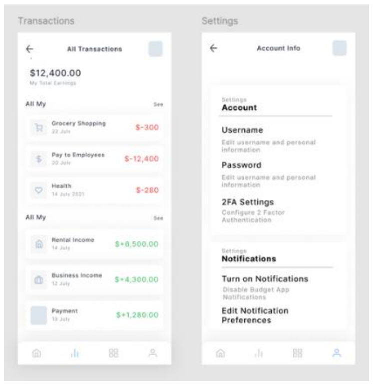
Creating this budgeting app has been all about making personal finance easier and more manageable for users. We wanted a tool that anyone could pick up and use effortlessly. While there were challenges, like making sure the app worked well on both iOS and Android and securely connecting to financial data, each step brought us closer to our goal. Now we're excited to put the finishing touches on and see the positive impact it will have for people looking to take control of their finances.

Appendix

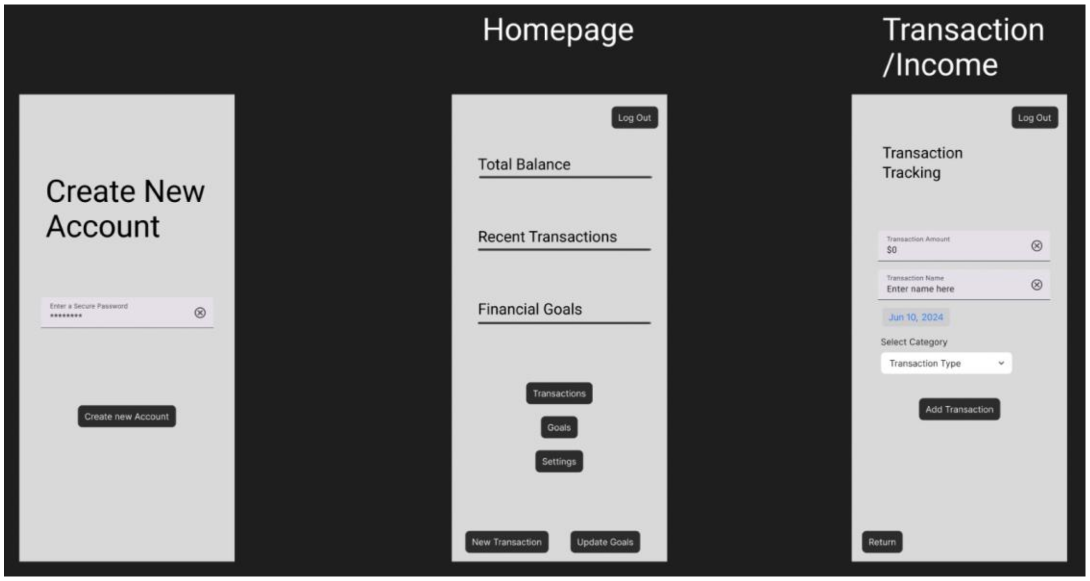
Appendix A - Figma App Mockups

- Avi:

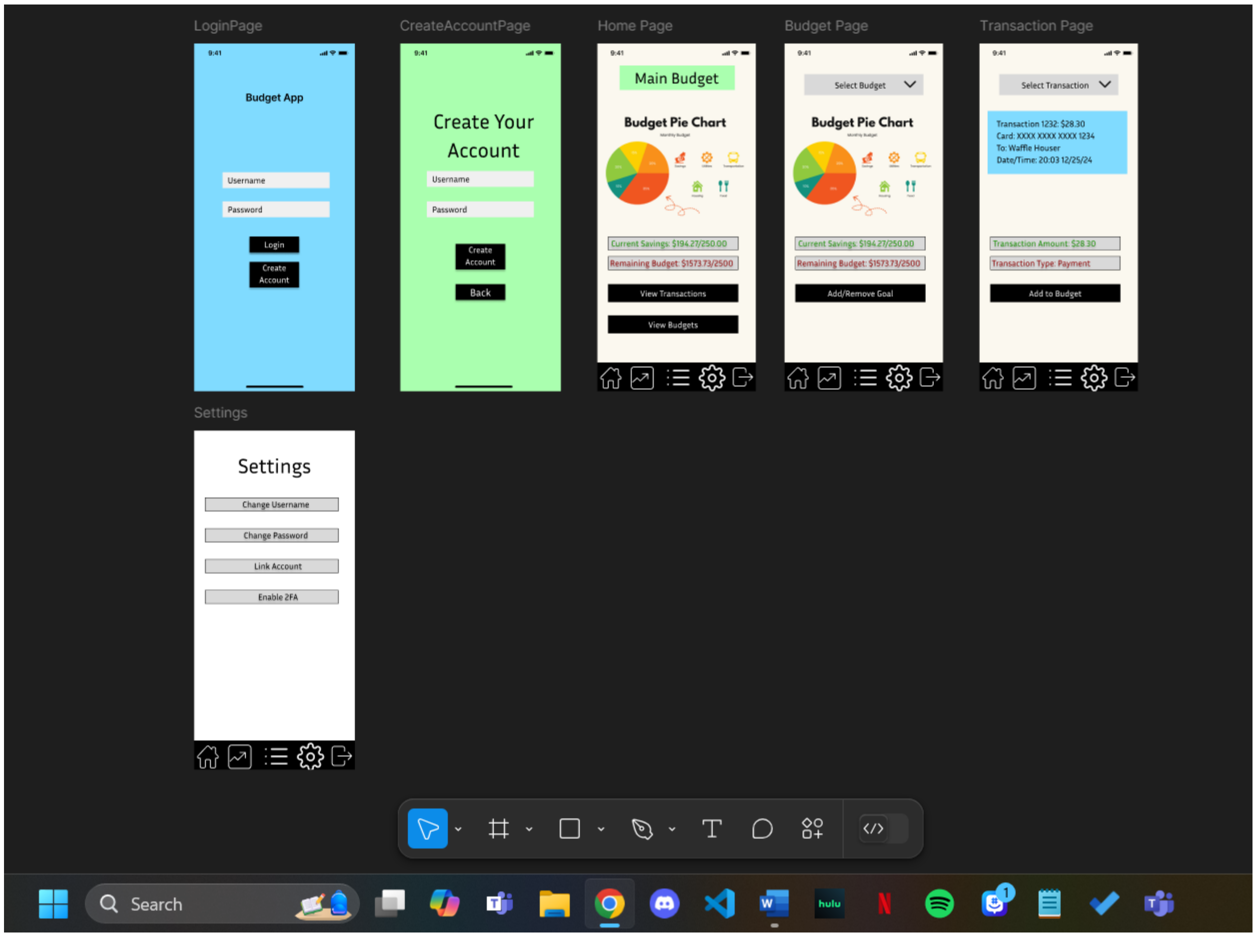




- Garrett:



- Josh:



Transaction 1232: $28.30
Card: XXXX XXXX XXXX 1234
To: Waffle Houser
Date/Time: 20:03 12/25/24

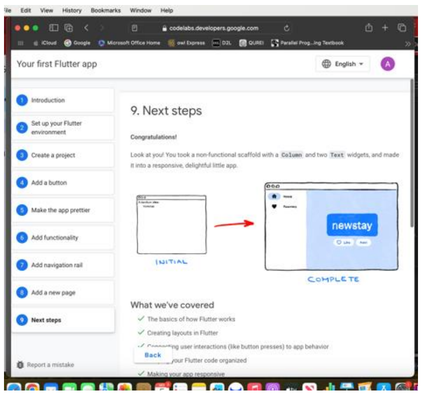
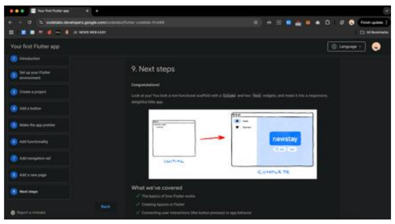
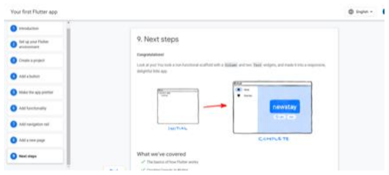
Appendix B - Gantt Chart

Sp26 - Gantt Estimate.xlsx

Appendix C – Glossary

- Flutter: An open-source UI software development toolkit created by Google for building cross-platform applications.
- Dart: A programming language optimized for building mobile, desktop, server, and web applications.
- API: Application Programming Interface, a set of tools and protocols used for building software and applications.
- Plaid: A service that enables applications to connect with users' bank accounts and retrieve financial data.
- JSON (JavaScript Object Notation) files: A standard file format that stores data in human-readable text and allows for easy data transmission.

Appendix D – Flutter Tutorial

Avi	Garrett	Josh
		

Your first Flutter app

9. Next steps

Congratulations!

Look at you! You took a non-functional scaffold with a `Column` and two `Text` widgets, and made it into a responsive, delightful little app.

What we've covered

- ✓ The basics of how Flutter works
- ✓ Creating layouts in Flutter
- ✓ Connecting user interactions (like button presses) to app behavior
- ✓ Keeping your Flutter code organized
- ✓ Making your app responsive

Appendix E – References

- Flutter Documentation: https://docs.flutter.dev
- Plaid API Documentation: https://plaid.com/docs/quickstart/
- FL Chart Developer Page https://flchart.dev/
- Flutter Architecture: https://docs.flutter.dev/resources/architectural-overview