




## SP26 – Red Budget App

---

CS 4850 – Section 03 – Fall 2024  
August 27<sup>th</sup>

 Josh Wilmath Developer	 Garrett Heffner Developer
 Avi Reale Documentation	

### ***Project Team***

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

## Table of Contents

<b>1. INTRODUCTION AND OVERVIEW .....</b>	<b>2</b>
<b>2. DESIGN CONSIDERATIONS .....</b>	<b>3</b>
2.1. ASSUMPTIONS AND DEPENDENCIES .....	3
2.2. GENERAL CONSTRAINTS.....	3
2.3. DEVELOPMENT METHODS.....	3
<b>3. ARCHITECTURAL STRATEGIES .....</b>	<b>4</b>
3.1. SOFTWARE ARCHITECTURE .....	4
<b>4. SYSTEM ARCHITECTURE .....</b>	<b>5</b>
<b>5. DETAILED SYSTEM DESIGN.....</b>	<b>6</b>
5.1. CLASSIFICATION .....	6
5.2. DEFINITION .....	6
5.3. CONSTRAINTS .....	7
5.3.1 <i>User Constraints</i> .....	7
5.3.2 <i>Data Synchronization</i> .....	7
5.3.3 <i>Internet Constraints</i> .....	7
5.4. RESOURCES .....	7
5.5. INTERFACE/EXPORTS.....	7
<b>6. GLOSSARY.....</b>	<b>8</b>
<b>7. BIBLIOGRAPHY .....</b>	<b>8</b>

## 1. Introduction and Overview

The purpose of this Software Design Document (SDD) is to outline the architecture, components, interfaces, and other important aspects of the system's design. It will be a guide for our team when developing and implementing our system.

Given our short time frame, our design approach aims to be simple but efficient. Our team has decided to use Flutter and Dart for our development. These tools and technologies have been

chosen to help us support our cross-platform compatibility goal. Our application's design will prioritize the essential features to ensure the delivery of a functional and user-friendly application.

## **2. Design Considerations**

### ***2.1. Assumptions and Dependencies***

While designing our budget app, we are assuming the application will be downloaded and used on android and iOS mobile devices. Therefore, the application will not have native compatibility with iPad devices due to the separate requirements of iPadOS. Another assumption we will be relying on is that users are running our application on devices that are at least somewhat recently released, for example within the last few years.

This application will be designed for a wide range of users and as such will be designed on the assumption of minimal technical literacy and will maintain a level of ease of use for the average user of our application. We are currently developing the application to be free to download and use, and future monetization would be small scale based on the target clientele being individuals for personal use. Our application will also leverage external APIs, such as Plaid, which adds a layer of risk for our users if Plaid encounters errors or downtime.

### ***2.2. General Constraints***

The major constraint that will impact the application's design is the intended usage on mobile devices. This means there are more limited CPU resources than a desktop application would have access to. This will also affect the coding as iOS and android devices will have different language requirements than desktop applications. A mobile application also constrains possible GUI choices as ease of use on a small screen is a primary concern.

Another constraint that our design must consider is the communication between mobile devices running the application and the that stores user login information, short-term and long-term financial information, and relevant banking information securely. Due to using Plaid to connect financial information to the application necessary security must be implemented to avoid private identity information from being stolen some method of secure server communication will need to be implemented.

### ***2.3. Development Methods***

The main development method that will be utilized will be the Agile methodology. This method was chosen because it focuses on a high degree on planning prior to developing actual code for the prototype, prioritizing a long-term plan for development rather than rushing to write physical code. Another benefit of choosing the agile method is that it synergizes with multiple release stages to fix bugs that are encountered after a previous release, which will help develop higher fidelity code.

### 3. Architectural Strategies

#### 3.1. *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://fluttergems.dev/packages/fl_chart/)

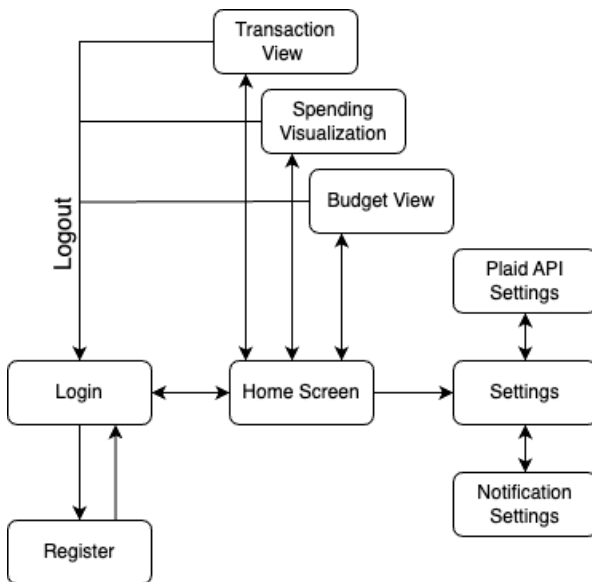
<https://flchart.dev/>

**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.

## 4. System Architecture

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.

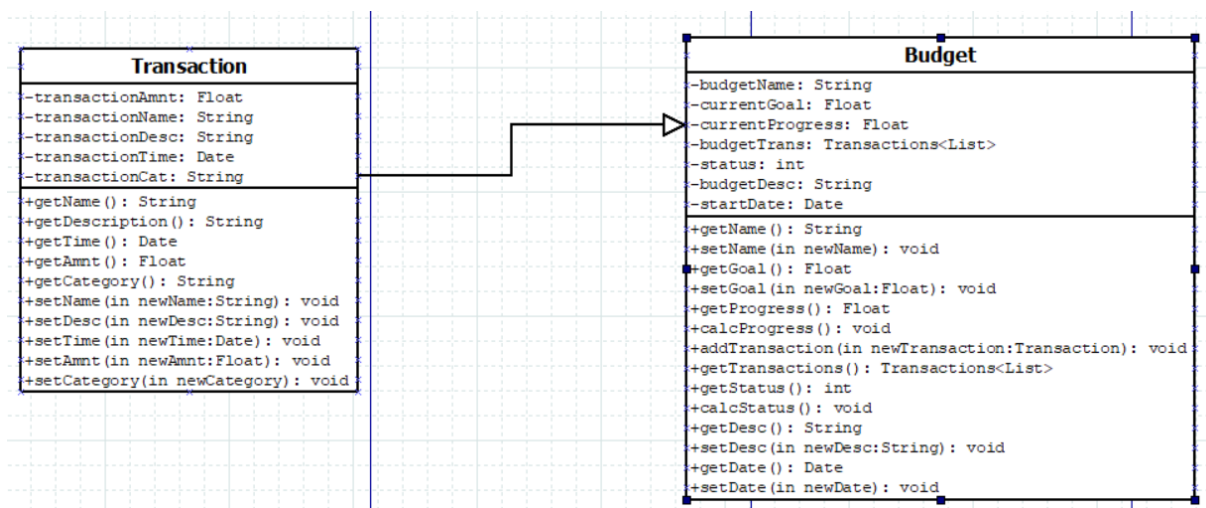
We plan to store the budgets and transactions using classes to simplify storage and storing instances of those classes in JSON files allowing for ease of exporting and importing for possible future functionality. Architecturally, this will serve as the model manipulated by the Dart code and displayed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

## 5. Detailed System Design

### 5.1. Classification

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.

### 5.2. Definition

This section will outline the key components, terminologies, and concepts that are important to understanding the system's design and its functionality:

- API (application programming interface): An API is a set of protocols and tools that allow different software applications to communicate with each other. The plaid API will be used to securely connect users' bank accounts with our app.

- JSON (JavaScript Object Notation) files: A standard file format that stores data in human-readable text and allows for easy data transmission.

### **5.3. Constraints**

#### **5.3.1 User Constraints**

The design of the application's login page assumes that there is a single user on the phone the app is downloaded on. Thus, login information is downloaded locally in a .json file assuming there is only one account on the device.

#### **5.3.2 Data Synchronization**

The application uses Plaid to collect financial data from external financial institutions, so the application is designed around the constraint of not being able to constantly call Plaid's API, so we will be implementing a sync button to call the API to collect data to synchronize with the app.

#### **5.3.3 Internet Constraints**

The majority of the application is designed to be accessed without requiring an internet connection. This allows more freedom for accessing information within the app. However, this will limit the potential for updating transaction information without internet access.

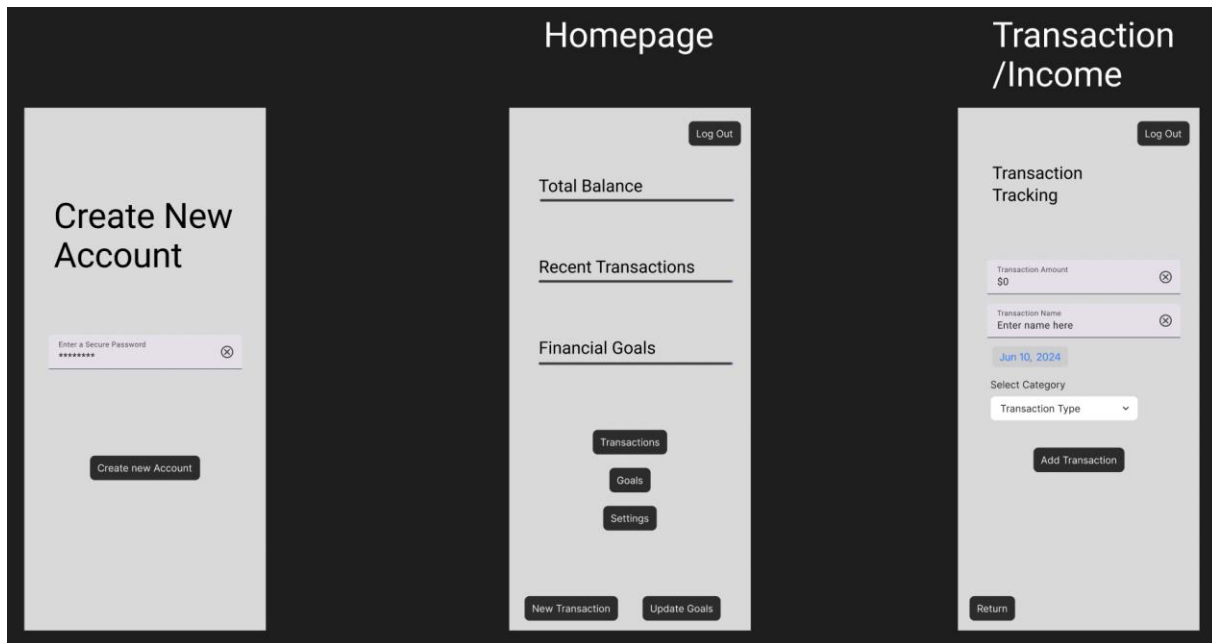
### **5.4. Resources**

The development of the mobile app will require different types of resources including technical and human resources. The primary technical resources include the Flutter and Dart SDKs for app development and access to the Plaid API for financial data integration. Secondary technical requirements are libraries for Dart that allow for easier implementation of more advanced functions for mobile interaction. This project also requires a development team (our team) which will be considered our human resources. The documentation of Flutter, Plaid, and FL Chart are also important to integrate them effectively in development.

### **5.5. Interface/Exports**

The mobile app will interact with external systems and provide an interface to facilitate data exchange and user interaction. The app will integrate with the Plaid API to securely connect users' financial accounts. The external API interface will allow the app to fetch and display user data like transaction histories and account balances. The data retrieved from the Plaid API will be processed by the synchronization module and it's displayed on the user interface. The user interface will allow users to interact with various features such as tracking expenses, setting budgets, and viewing financial charts.

We mocked up some screens to get a feel for how the user will interact with the control flow, which helped us to 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 in development and keep the UX intuitive.



**Figure 3:** *Prototype screens for the application to visualize eventual mockups.*

## 6. 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.

## 7. Bibliography

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>