




SP26 – Red Budget App

CS 4850 – Section 03 – Fall 2024

August 27th

 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

Table of Contents

1.0	Introduction	3
-----	--------------------	---

1.1 Overview	3
1.2 Project Goals	3
1.3 Definitions and Acronyms	3
1.4 Assumptions.....	3
2.0 Design Constraints	4
2.1 Environment.....	4
2.2 User Characteristics	4
2.3 System	4
3.0 Functional Requirements	4
4.0 Non-Functional Requirements.....	5
4.1 Security	5
4.2 Capacity.....	6
4.3 Usability	6
4.8 Other	6
5.0 External Interface Requirements	6
5.1 User Interface Requirements.....	6
5.2 Hardware Interface Requirements	6
5.3 Software Interface Requirements.....	6
5.4 Communication Interface Requirements.....	6
APPENDICES	Error! Bookmark not defined.

1.0 Introduction

1.1 Overview

This Software Requirements Specification (SRS) document is to outline all requirements and functionalities for the development of our mobile budget app. This document was created to help guide our team in the development process. This document will help clearly define all system requirements to ensure that our final product meets the stated goals.

This document does not cover topics such as timeline, budget, development methods, or testing procedures. These project issues will be addressed in different project management and quality assurance documents.

The mobile budget app is a cross-platform mobile application designed for iOS and Android platforms. This application is designed to help users manage their finances. It will include features that allow users to track expenses, income, and savings. It will also allow users to monitor finances, set financial goals, and analyze spending patterns through visual aids such as charts and graphs.

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

- **SRS: Software Requirements Specifications** - A document that specifies the functional and non-functional requirements of a software system.
- **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.

2.0 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.

3.0 Functional Requirements – EXAMPLE for MOBILE APP

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.

Commented [GH1]: How are we defining secure? How are passwords stored and verified with the user's inputs? Error message with wrong password?

- 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.

Commented [GH2]: do we have a new screen for creating expenses and viewing all expenses or are they the same screen?

Commented [GH3]: The categories are the same as the user made ones right? And should we mention setting a time frame and amount (ex. monthly grocery budget - \$500)

Commented [GH4]: I like the idea of notifications, but I think it brings up more questions (how many API calls to check spending to the budget, on what time interval?) also notification settings of when and what to notify

Commented [GH5]: similar concern with API call counts

Commented [GH6]: no email/SMS verification here? maybe having the user reenter the password to change it if there's any security concern about this

4.0 Non-Functional Requirements (use if applicable)

4.1 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

4.2 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.

4.3 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.

4.4 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.

5.0 External Interface Requirements (use if applicable)

5.1 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.

5.2 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.

Commented [GH7]: Not really sure if we need to go into detail here because Flutter should abstract any interfaces with the hardware

5.3 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.

5.4 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.