Development Document

for

Sp-26 Green Budget App

CS 4850 - Section 01 – Fall 2024

November 10, 2024

Prepared by: *Kelly Erhunse, Kody Clark, Katie Carlson, Tristan Stocks*

Outline

Don't Go Broke

Our application is a budgeting app called 'Don't Go Broke.' It is designed to help users manage personal finances and improve financial literacy. With an intuitive interface, the app provides features like the tracking of recent transactions, budgeting tools, and personalized financial insights. It also empowers users to set savings goals, track debts, monitor spending patterns through charts, and receive advice about their finances.

Built using Flutter and Dart, the app operates cross-platform on Android and iOS, and provides users with a convenient tool for everyday financial management.

1. Goals

a. Provide users with a secure and convenient personal finance management tool
b. Improve financial literacy for those who struggle with budgeting
c. Allow users track spending and set savings goals
d. Offer financial insights
e. Remind the user of upcoming bills/debts they need to pay
f. Allow user to input their bank account transaction information, in order to help them budget
g. To give users the option to integrate their bank account via Plaid
h. To allow users to create an account, in order to keep track of any data they may want to store for the future
i. To create an application that will be cross platform

2. System Features

a. Account Creation/Deletion
b. Account Recovery
c. Login System
d. 2 Factor Authentication
e. Bank Account Linking

f. Budgeting Tools
g. Charts
h. Data Storage (in Firebase)

3. Widgets

a. Stateless Widgets:

Stateless widgets were implemented for elements like headers, labels, icons, and static buttons, where the content is not expected to change dynamically.

Examples: *The text labels, titles, and buttons were implemented using Stateless widgets, since they do not change during the user's interaction with these screens.*

b. Stateful Widgets

Stateful widgets were used for components that required dynamic changes based on user actions and inputs.

Example: *User actions like adding, editing, or deleting a transaction trigger changes in the transaction list.*

4. Front-End Design

a. Account Creation Screen
   i. *Name Field*
   ii. *Email Field*
   iii. *Phone Number Field*
   iv. *Password Field*
b. Login Screen
   i. *Email Field*
   ii. *Password Field*
   iii. *Create Account Link*
   iv. *Forgot Password Link*
c. Home Screen
   i. *Upcoming Bills Subsection/Screen*
   ii. *Spending Summary Subsection/Screen*
   iii. *Debt Payments Subsection/Screen*

    iv. *Tips and Insights Subsection/Screen*
d. Budget Screen
    i. *Monthly Expenses Subsection*
    ii. *Savings Goals Subsection*
e. Transaction Screen
    i. *Account Balance*
    ii. *Recent Transaction Section*
f. Overview Screen
    i. *Take Home Money after Taxes Subsection/Screen*
    ii. *Spendings compared to Budget Subsection/Screen*
    iii. *Credit Information Subsection/Screen*
g. Settings/Profile Screen
    i. *Password and Recovery Subsection/Screen*
    ii. *Data Privacy Subsection/Screen*
    iii. *FAQ Subsection/Screen*
    iv. *Contact Subsection/Screen*

5. Front-End Implementation

a. Account Creation and Login

Objective: Provide secure user authentication for personalized budget management.

Implementation:

- *Focused on a simplified onboarding experience for first-time users with prompts to guide through account setup.*
- *Implemented using Flutter, featuring input fields for username/email and password.*
- *Added fields for account setup, including email verification for secure access.*

b. Header and Navigation Bar

Objective: Provide a way for the user to navigate to other pages within the application.

Implementation:

- *Organized UI with buttons linking to these pages: home, budget, transaction, overview, and account settings.*
- *Logo (which is to the top left) is displayed in the header, and the settings button is to the top right. The other four buttons are in the navigation bar at the bottom.*

c. Home Screen

Objective: Provide a quick overview of the user's financial status, including total budget and savings goals.

Implementation:

- *Simple, user-centric layout showcasing essential budget insights at a glance, like the user's current account balance and monthly income.*
- *Has buttons that link to other subpages (such as 'Upcoming Bills', 'Spending Summary', 'Debt Payments', and 'Tips and Insights').*

d. Budget Screen

Objective: Allow users to track spending against their allocated budget and set monthly or weekly goals.

Implementation:

- *Display a pie chart with color-coded categories for expense tracking, making budget management visually intuitive.*
- *Include a section below the pie chart that pairs each category with its expense cost.*
- *Include nother section below the monthly expenses, that lists all of the user's savings goals.*

e. Transactions Screen

Objective: Enable users to log and review detailed records of income and expenses.

Implementation:

- *Developed a simple list layout for viewing transactions by date and type.*

f. Overview Screen

Objective: Enable users to look at their top spending categories across multiple months (via a bar graph), allow them to calculate net income, allow them to compare their spendings to their currently set budget, allow them to calculate their take home money after taxes, and give advice/information about credit.

Implementation:

- *Displayed a 'Monthly Expenditure' graph displaying the user's top spending categories amongst multiple months.*
- *Developed a section that acts as a net income calculator.*
- *Developed a section that has subpages for additional tools (like calculating take home money after taxes, to have a page for credit advice, etc).*

g. Settings Screen

Objective: Enable users to look at and edit their profile information, set the currency they are currently using, edit their password/look at their password recovery options, to look at information related to data privacy and common questions the users may have about the app, and an option for users to be able to contact the developers/owners of the application.

Implementation:

- *Display a profile picture that the user can set.*
- *Display the user's profile information (their name, email, phone number) and give the option to edit their information.*
- *Display a 'Settings' section that allows users to change settings related to their account, and look up any questions they may have about the application.*
- *Have a 'delete account' option, that would redirect back to the login page if the user deletes their account.*

6. Back-End Design

a. Client Server Model

i. Objective: Compliance for FinCEN (Financial Crimes Enforcement Network) data protection assets

1. *Implemented through Firestore Database from Firebase software*
2. *Allows authentication programs such as two-factor authentication*
3. *Protects data from corruption by utilization of a backup and synchronization methods. This system also helps protect from changing database information through injectors since the database will check incoming client information against the database first.*

b. Data

i. Objective: Create robust categorization of information data fields that will be simple to query and call follow on algorithms for searching, categorizing, and displaying fields such as debits and credits.

1. *Implemented through over 18 data fields that will cover things such as credit, debit, transaction categories, card numbers utilized and account numbers that they will be queried.*
    - *Will be able to synchronize our data fields with the Plaid banking application*
    - *Will be able to manually plug in data information for users who do not want to share their banking information*

7. Back-End Implementation

a. Firebase platform

i. Authentication

1. *Implemented through firebase API calls and imports through the fiebase_options, create_user and main_page dart classes*
2. *From the firebase console on their website, can easily switch authentication methods from single source login, to 2 factor authentications including*
    a. *Authenticators*
    b. *Email*
    c. *Mobile telephone numbers*
        i. *combinations of all of the above*

ii. Realtime Database

1. *Implemented through firebase API calls and Firebase CLI*

2. *Original design specifications utilized this as programs primary database, upon further assessment and research, it was decided to utilize a different database, reasons included:*
    a. *legacy support from Firebase*
    b. *Limited query selection for management of data*
    c. *Limited nesting for data*

iii. Firestore Database

1. *Database chosen to replace Realtime Database due to:*
    a. *Currently most supported and recommended database option from Firebase*
    b. *Queryability*
    c. *Scalability*
    d. *JSON structure is stored in collections instead of a singular JSON tree.*

b. Plaid

i. Authentication

1. *Implemented through Plaid Link Android SDK*
    a. *Token from the server giving temporary tokens to clients accessing the Plaid database*

ii. Sandbox

1. *Allows test bed implementation of most functionality without allowing a live service*
    a. *Can not use live service at this time due to FinCEN requirements for team members to be trained in all aspects of data security, hard written operating documents and a myriad of other documentation for an application of this scope.*

8. Data Requirements

a. Budget Calculation

i. Create a robust budget calculation tool

1. *Dates*
2. *Categories of expenses/transactions (i.e, 'food', 'gas', etc)*
3. *Monetary values/amount of funds*

b. Recurring payments/monthly expenses

i. Requires a category field for recurring statements

1. *Date to pay each month*
2. *Notification reminder to each month following*
3. *Query for all recurring categories and add them together to show total amount of money needed for the month*

c. Remaining Monthly Funds

i. Amount still in balance after monthly expenses

ii. Requires calculation for monthly income - monthly expenses

Set Up

1) Ensure that you have Flutter installed and running

2) Create the screens that you will be using for the project

4) Ensure that the screens have functionality with the buttons on the pages

3) Create new Firebase project

4) Add Firebase files, configuration etc with Flutter project to set up connection, connecting the screens to the backend

5) Set up login system

6) Contact Plaid API and use provided sandbox, set up connection to use that data

7) Test the application, to make sure there are no unexpected behaviors

8) Continue to maintain the tests whenever new features are added or bugs are encountered