## **Final Project Documentation**

**Application Name:** *Budget Buddy*

**Author:** *Hillary Kravitz*

**Date:** *10-6-2025*

### **Project Overview**

#### **1.1 Problem Statement**

The problem my Application solves is kind of just turning your regular checkbook into a digital form, where it can show your current Balance of money you have (maybe even from multiple sources if you set it up that way), the amount of money coming in each month in the form of income, and shows your transactions in positives in negatives. It could also sort of be used a budgeting notepad if you will. Where you can record your balance, say, after you receive your paycheck. And then record all of your outgoing bills that will be coming out over the next month. It would then show you what remains to further budget out specifically.

#### **1.2 Target Audience**

This app is designed for people who may not be as good at managing their money, or need help visualizing it to avoid overspending, or save for something specific.

#### **1.3 Core Features**

* *Feature 1: User can record their monthly income.*
* *Feature 2: The user can also record incoming and outgoing transactions.*
* *Feature 3: The app has functionality to display their current balance after negative or positive transactions.*
* Feature 4: Also after each transaction is recorded, it displays what the balance was afterward that can also be viewed later on when further transactions are added.
* Feature 5: There’s also functionality to adjust their monthly income, and also set a specific amount to save away each month.

### 

### **Technical Design & Architecture**

#### **2.1 State Management Strategy**

This application uses the setState method along with a stateless widget as kind of the skeleton, and multiple stateful widgets used over the top of it for managing the local user interface state such as how it appears after transactions are added or income/balance data is manipulated. It also uses the Navigation functions to switch between screens for the Add Transaction screen as well as the Option Screen.

#### **2.2 Data Model**

Describe the main data structures or classes in your application. Include a code block for your primary model(s).

The main data structure in this application is just the \_MainPageState class that manages the state from runtime. It also uses the Navigator to switch between the AddTransaction and the Option screens, and contains standard doubles for the variables within like shown below.

// lib/Budget\_Buddy/main.dart

class \_MainPageState extends State<MainPage> {

double \_runningBalance = 4250.0; // (double) | manually entered at start

double \_monthlyIncome = 3000.0; // (double) | manually entered

List<Map<String, dynamic>> \_transactions = [

{"date": "10/3/2025", "amount": -250.0, "balanceStamp": 4250.0},

{"date": "10/3/2025", "amount": -1000.0, "balanceStamp": 4500.0},

{"date": "10/3/2025", "amount": 500.0, "balanceStamp": 5500.0},

]; // (List<String>)

double \_currentSavings = 1000.0; // (double)

double \_savingsPerMonth = 500.0; // (double)

int \_currentIndex = 0;

#### **2.3 Persistence / API Strategy**

My application doesn’t really use any shared data because it’s all hardcoded, which isn’t ideal for an application that serves this purpose. But I was having trouble understanding how to add that functionality, and it wasn’t a suggested addition when we were working on it although I do understand it was supposed to do one or the other in order to handle data I/O.

#### **2.4 Widget Tree Diagram**

A diagram of a diagram

Description automatically generated

### **Setup & Installation**

#### **3.1 Prerequisites**

* Firebase Studio and Flutter SDK

#### **3.2 Installation Steps**

1. Clone the public repository you created: **git clone <your-repo-url>**
2. Navigate into the project directory: **cd <your-project-directory>**
3. Set up your Flutter files: **flutter create**
4. After adding dependencies: **flutter pub get**
5. Run the application: **flutter run**

### 

### **Automated Testing**

#### **4.1 Testing Strategy**

My approach to testing it was running the main.dart file many times and making sure that it would work correctly when I put in different inputs and things. If I had more time and expertise in the subject to work on it, I would probably add functionality to make it so the Savings function could have been implementation where it would be able to subtract the monthly amount to be saved from the available balance, and could be set on like a calendar alert type thing where it would do it automatically every month/payday/etc (user defined). I would also like to have been able to add the file I/O functionality so the transaction lists and things wouldn’t have had to been hardcoded, and could’ve been typed in by the user and then be accessible later as well. Because of this I was not able to do any further testing than just running the file and trying it out in varying ways.