



# **COMPUTER SCIENCE PROGRAM**

## **Budget Buddy – Personal Budgeting Application**

### **Capstone Project Report**

#### **Prepared By:**

Christian Solis

Bachelor of Science in Computer Science

#### **Faculty Advisor:**

Visvasuresh Govindaswamy

Associate Professor of Computer Science

**May 3, 2024**

## **Budget Buddy – Personal Budgeting Application**

### **Capstone Project Report**

**Approved by the Supervisor**

**Signature:**

---

Visvasuresh Govindaswamy,  
Professor of Computer Science

**TABLE OF CONTENTS**

<b>Introduction</b>	<b>6</b>
Abstract	6
Market Need	7
Competitors in the Market	7
Comparison Chart	9
Future Work	9
<b>USER MANUAL</b>	<b>10</b>
Installations Guide	10
Using the Application	10
Welcome Page	10
Log In	11
Create Account	11
Reset Password	12
Home Screen	12
Add Account	13
Add Expense	14
Charts	15
Manage	16
Profile	17
<b>METHODOLOGY</b>	<b>17</b>
Agile Development	17
<b>APPLICATION DESIGN</b>	<b>18</b>
Purpose of Design Diagrams	18
<b>DESIGN DIAGRAMS</b>	<b>18</b>
Graphical User Interface	18
Welcome & Log in	19
Create Account & Terms and Conditions	20
Home Screen	20
Add Account & Add Expense	21
Charts	21

CHRISTIAN SOLIS CAPSTONE	4
Manage	22
Profile	22
Use Case Diagrams	23
Log In Use Case Diagram	23
Home Screen Use Case Diagram	25
Hierarchical Task Analysis	26
Hierarchical Task Analysis Diagram	26
Sequence Diagrams	26
Log In Sequence Diagram	27
Create Account Sequence Diagram	28
Add Expense Sequence Diagram	29
Manage Expense Sequence Diagram	30
Finite State Machine Diagrams	31
Finite State Machine Diagram	32
Finite State Machine Diagram (Top Section)	33
Finite State Machine (Bottom Section)	34
Finite State Machine Definitions	35
Data Flow Diagrams	41
Data Flow Diagram	41
Data Flow Diagram(Top Section)	42
Data Flow Diagram(Bottom Section)	43
<b>CLASSES</b>	<b>44</b>
Class Explanations	44
AboutUs.java	44
AccountAdapter.java	44
AccountModel.java	44
AddAccount.java	44
BudgetLimitModel.java	45
ChartsPage.java	45
CreateAccount.java	45
ExpensePage.java	46
ForgotPassword.java	46

CHRISTIAN SOLIS CAPSTONE	5
HomePage.java	46
LogIn.java	46
MainActivity.java	47
ManagePage.java	47
ProfilePage.java	47
ReadWriteUserDetails.java	47
TransactionAdapter.java	48
TransactionModel.java	48
TransactionRecyclerViewInterface.java	48
<b>TESTING</b>	<b>48</b>
Tests Used	49
Black-Box	49
Grey-Box	49
White-Box	50
<b>CONCLUSION</b>	<b>50</b>
CODE	51
Java Code	51
<b>REFERENCES</b>	<b>110</b>

### **Introduction**

This document seeks to serve the purpose of presenting the design and implementation of the Budget Buddy mobile application. This is a personal finance application developed using Android Studio for the purpose of aiding helping users keep track of their expenses through a friendly user interface. Budget Buddy helps users keep a record of their spending by manually filling out transactions and using diagrams, such as pie and bar charts, to easily recognize spending patterns and the progress towards the budget. The application is meant to help increase your financial responsibility and help improve spending behaviors by providing an easy-to-use interface that will make it easier for the user to track expenses and maintain updated budgets.

### **Abstract**

This project falls within the field of mobile application development, specifically focusing on personal finance management. The purpose of this project is to create a user-friendly mobile application, Budget Buddy, using Java and XML in Android Studio. Budget Buddy is designed to help users track their expenses and income, categorize transactions, and visualize their financial data. The primary objective of this project is to provide users with a convenient tool for managing their personal finances. Budget Buddy allows users to input their expenses and income, categorize them, and view their financial data in graphical formats. This helps users gain insights into their spending habits and financial health. Future enhancements to Budget Buddy could include integrating with banks using the "Plaid" service to automatically import transactions, providing users with a more seamless and efficient way to track their finances.

CHRISTIAN SOLIS CAPSTONE

Additionally, improvements to the user interface and additional features for financial analysis could be considered to enhance the app's usability and effectiveness.

### **Market Need**

There are several existing budgeting applications that exist already with millions of downloads. They all provide similar features some offering unique ones like being able to share their expenses with other users that make them stand out however the biggest drawback of all these applications is that they lock away most of these features behind a paywall. This leaves room for an application to take the best features of all of them and combine them into one with no paywall restricting the user from taking full advantage of the application. Only 32% of Americans maintain a monthly budget, and among those, just 29% use a mobile application for budgeting. With Budget Buddy, a user friendly application that is completely free, would be beneficial to millions of users.

### **Competitors in the Market**

There are many budgeting applications when searching for one in the play store but the three that stood out the most based on number of downloads and ratings were Mint, Wallet, and Money Manager. I will go into detail about what each application offers.

Mint is the first application that is presented when searching for a budgeting application. It is the most popular of the three with over 10 million downloads. Mint gives users the ability to log their expenses either manually or connecting their bank account for automatic updates. Users are able to see their spending habits through a multitude of charts and graphs. Mint stands out by offering a multitude of options on how to view or log your transactions from setting up repeated

## CHRISTIAN SOLIS CAPSTONE

transactions, labeling expenses based on the category. One its biggest downsides recently is that they were bought out by Credit Karma and the ability to create an account has disappeared and now if users wish to use the app, they are forced to make a Credit Karma account with Mint. In addition to those users have noticed the decline in quality of updates resulting in various bugs.

Wallet is the second most popular app with over 5 million downloads with an average rating of 4.4 stars on the google play store. It has a sleek simple user interface that is aesthetically pleasing. Like other budgeting applications Wallet allows users to manually input their expenses and label them based on their category and offering other details such as date, location, account which transaction takes place in, and a photo. Wallet looks at your planned expenses and tells you about your projected balance at the end of the month based on the current balance of your account. The standout feature for this application is that you can share your expenses with other users, so you are able to see each other's transactions.

Lastly Money Manager was the one that stood out the least, only offering the baseline features such as being able to input expenses, view your monthly expenses and set budgets as well. Like the rest of these apps this apps falls behind by locking a majority of the functionality behind a premium subscription. The app also has various ads all over the place reducing the quality of life. The biggest complaint of this application is that it is blacklisted in some cases for reasons of their possibly being malware or viruses.



### Comparison Chart

	Mint	Wallet	Money Manager	Budget Buddy
USER FRIENDLY	✓	✓	✓	✓
AD FREE	✗	✓	✗	✓
CONNECTS TO BANKS	✓	✓	✗	✓
EDIT EXPENSES	✓	✓	✓	✓
SHARE EXPENSES	✗	✓	✗	✓
COMPLETELY FREE	✗	✗	✗	✓

The one consistent feature across all the most popular applications is that they are user friendly and allow users to edit their expenses. On the other hand, they are not all ad free or are missing features like being able to share expenses or being completely free. My app Budget Buddy takes the best features from all applications and puts them into one app that is also ad free with no premium subscription that locks features from users.

### Future Work

Future work plans for Budget Buddy would include the ability to set a goal to save up to such as if the user would want to save money for a trip. Additionally, I would want to have the user be able to connect their bank by using the Plaid API, this would make it so the user does not have to manually enter each transaction. Lastly, I would want to create a feature where users can

CHRISTIAN SOLIS CAPSTONE

set up families and multiple users in one household could share and view their expenses. These features would expand Budget Buddy's functionality and make it stand out much more in the market.

## **USER MANUAL**

### **Installations Guide**

Budget Buddy can be found on the Google Play Store it is currently the only way to obtain it. Once the application has been fully downloaded it is ready to use.

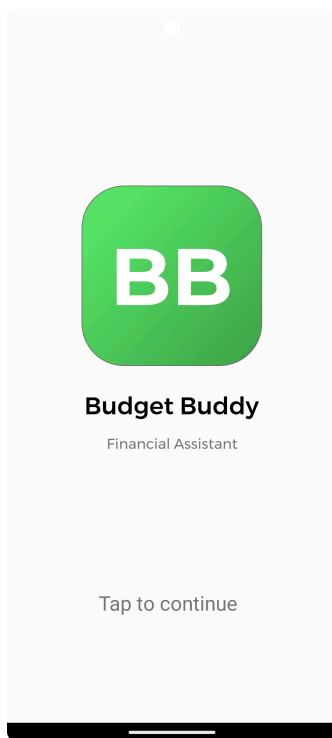
### **Using the Application**

#### **Welcome Page**

When Budget Buddy is opened the user is greeted with a welcome screen.

To continue the user must tap anywhere on the screen to proceed to the

Log in page.



### Log In

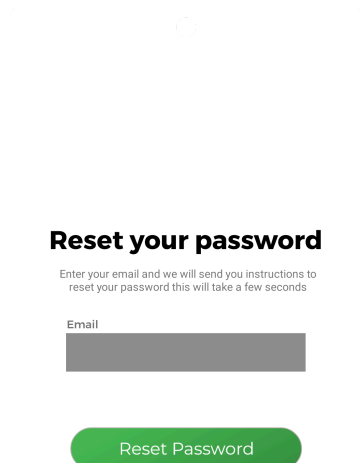
The Log in screen allows you to create an account or sign in to an existing one. If you are already registered, you can fill out your information and click “Log in” to proceed to the home screen. If you forget your password, you can click “Forgot Password?” and it will bring you to a page to reset it. If you do not have an account, you can click “Sign up” to create an account. Lastly you can click “About us” at the bottom and it will give you information about the application.

### Create Account

If you do not have an account, you can register for one by entering a valid email and a password. The user must have a password that meets the current standards of security which would be to at least add one number in your password in addition to having six or more characters. Once you click “Create Account” it will bring you back to the Log In page where you will need to fill out your information. If you accidentally clicked “Sign Up” you can return by clicking back. When you click “Sign up” you are agreeing to the terms and conditions that can be viewed by clicking the text.

### Reset Password

If you forgot your password or wish to reset it, you can do that by entering your email. To get to this page you must click “Forgot Password” on the Log in page. Enter your email and click “Reset Password”. If your email is registered with Budget Buddy, you will receive an email shortly with a link to reset your password. After you click “Reset Password” it will return you to the Log in page.



**Reset your password**

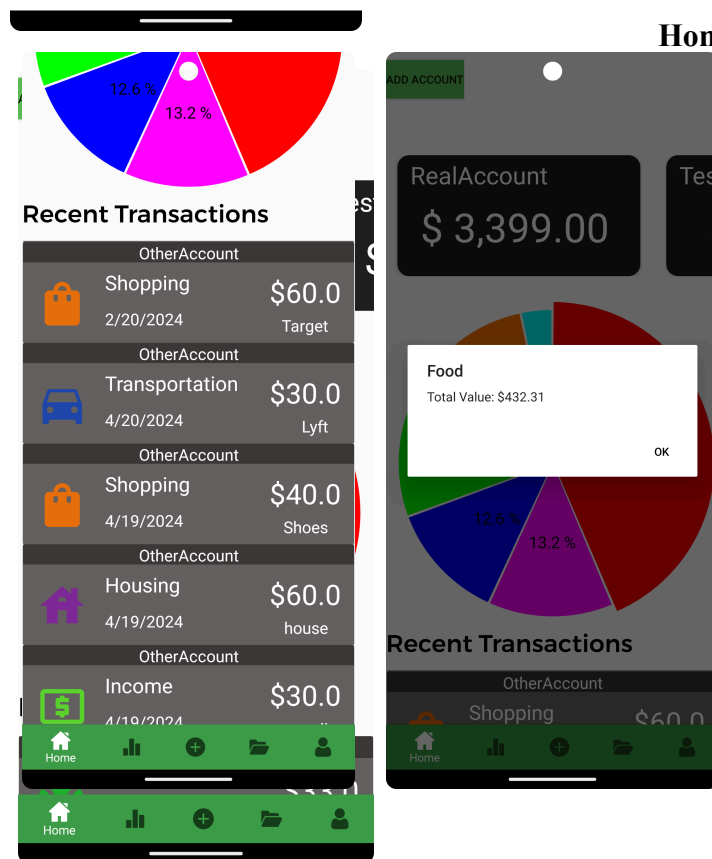
Enter your email and we will send you instructions to reset your password this will take a few seconds

Email

Reset Password

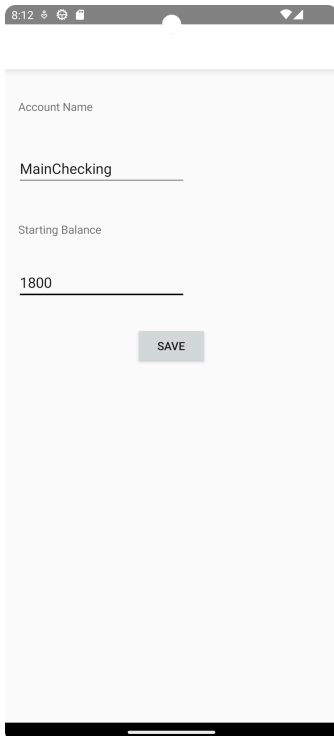
### Home Screen

The home screen is the first page you will see once you sign in. If you do not have any transactions the page will appear blank. You can start by first adding an account by clicking the “Add Account” button in the top left corner. This button will bring you to a brand-new page where you will be able to fill out your information. If you have any transactions logged into the app the pie chart will begin to appear displaying how much of your spending has gone to each category.



## CHRISTIAN SOLIS CAPSTONE

When you click a slice of the pie chart a dialog box will appear telling you how much you have spent in that category. Below the pie chart will be a list of your recent transactions. The list will take transactions from all of your accounts displaying how much you spent, location, date, the category, and which account the transaction took place in.



### **Add Account**

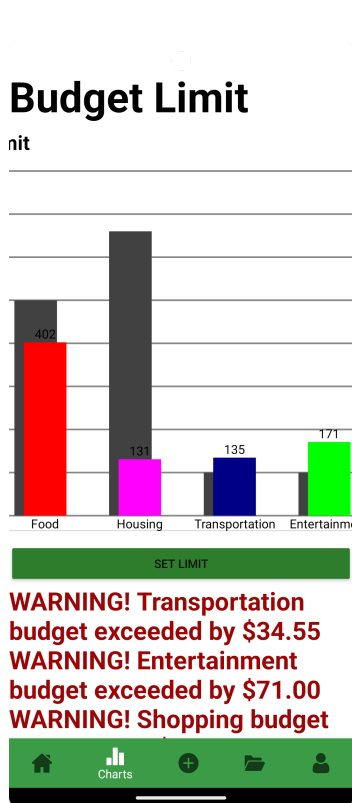
The Add Account screen is where you can enter your account. You can name your account whatever you would like in the Account name line and then you will be able to enter the starting balance of your account. Once you have entered all the necessary information you click the save button and it will bring you back to the home screen.

### Add Expense

Similar to the Add Account screen this is where you will manually log all your expenses. To log an expense, you will need to fill in the following spaces amount where you will put down how much the transaction was.

Notes, this could be used for many different purposes such as the location where the transaction took place or any additional context that will help you remember what the transaction was. Next you will select the account in which the transaction took place. If you have multiple accounts, you can click the spinner that will drop down the various account you have entered previously in the “Add Account” page. To add a date all you must do is click the “Select Date” that will prompt a dialog box in where you can select the date. Lastly you will be able to categorize your

transaction in the following types of categories, income, food, housing, transportation, entertainment, shopping, and miscellaneous. Once you have entered all your information click the save button. A message at the bottom of your screen will let you know if your transaction has been saved successfully. You will be able to enter as many expenses as you would like. Once you are done you can navigate to any of the other pages withing the app.



**Charts**

**Budget Limit**

Limit

**Set Limit**

Food	500.0
Housing	660.0
Transportation	650.0
Entertainment	100.0
Shopping	90.0
Misc	770.0

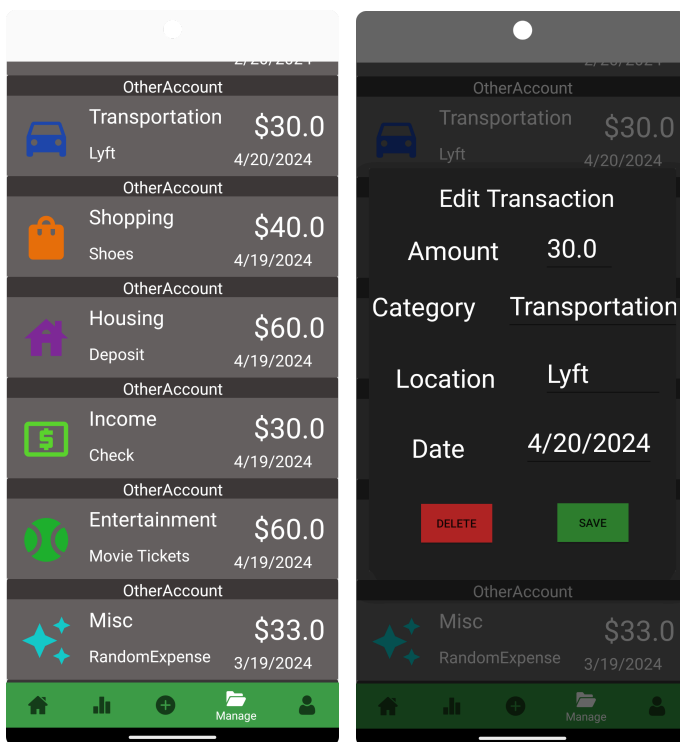
CLEAR ALL SAVE

exceeded by \$10.00

This is where you will be able to set and view your budget limits. The budget limits are shown with a bar graph. The grey will represent the limit for each category. This will be determined by the limit you set in the set limit dialog box. To set a budget limit click the “Set Limit” button below the bar chart. Here is where you can set all your budget limits for each category.

Once you have finished setting every limit click “Save”. If you have not set a limit the budget limit for that category will be 0. If you wish to edit your budget limits, you can click the same button again. If you would like to reset all the budget limits at once you can click the “Clear” button that will reset every limit back to 0. If you have gone over your budget limit the app will let you know below the bar chart and how much you have gone over.

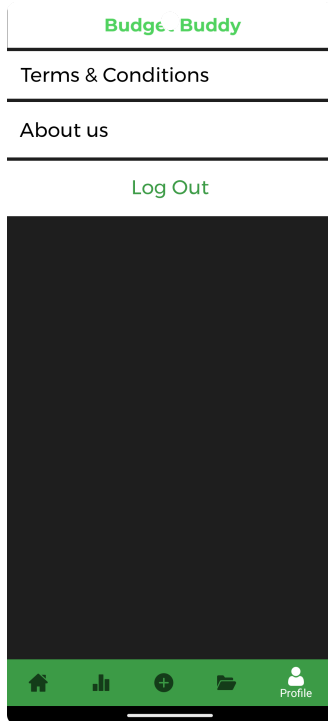
## Manage



The manage page is where you can edit and delete all your transactions. This also makes it slightly easier to view more transactions. To edit or delete all you must do is click the transaction of your choice. A dialog box will pop up allowing you to edit the following, amount, Category, Location, and Date. It is important to properly change information properly to avoid any errors. Once you are done editing the transaction you click the “Save” button that will close out the box for

you. To delete a transaction, you can click the “Delete” button and will delete the transaction from the account immediately.





### Profile

Lastly the profile page is where you can view the Terms & Conditions, About us page, and Log Out. To view the Terms & Conditions click the button and a dialog box will pop up. To view About us click the button and it will bring you to a page explaining what Budget Buddy is about. When you click the “Log out” button it will sign you out and bring you back to the Log in page where you will have to sign back in to access your account again.

## METHODOLOGY

### Agile Development

The Agile Scrum process is a specialized framework of Agile methodology, aiming at high communication, collaboration, and speedy development of complex software and product development projects. The main roles in Scrum include Scrum Master, who facilitates the process and helps to get rid of impediments; Product Owner, who represents the stakeholders and

CHRISTIAN SOLIS CAPSTONE

takes care of the product backlog; and Development Team, which is responsible for accomplishing tasks. This enables you to keep track of what needs to be done with Scrum artifacts Product Backlog, Sprint Backlog, and Increment. And so, the Scrum framework is structured around several ceremonies, including Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective. Each of them, in one way or the other, contributes to planning, review, and continuous improvement. Sprints give the team an opportunity to be able to chunk the project within a clearly predefined set of tasks, culminating in the increment review. Scrum offers a structured yet flexible approach that allows for constant feedback and iterative progress. This is ideally suited for projects whose requirements are highly likely to change fast or have to change significantly with time.

## **APPLICATION DESIGN**

### **Purpose of Design Diagrams**

Design diagrams serve multiple purposes, including demonstrating and visualizing complex systems to ensure they are easily understandable for both team members and stakeholders. This will help the team to do forward planning to anticipate any future complexities and allow you to plan accordingly. In an agile environment this helps with continuous changes and adaptation ensuring the team meets the goal.

In this document we utilize five types of diagrams. Use Case Diagrams, Hierarchical Task Analysis, Sequence Diagrams, Finite State Diagrams, and Data Flow Diagrams. These diagrams are important in demonstrating how different parts of the application interact with one another. Additionally, they show how data moves between activities and the database, proving a clear visualization of the system.

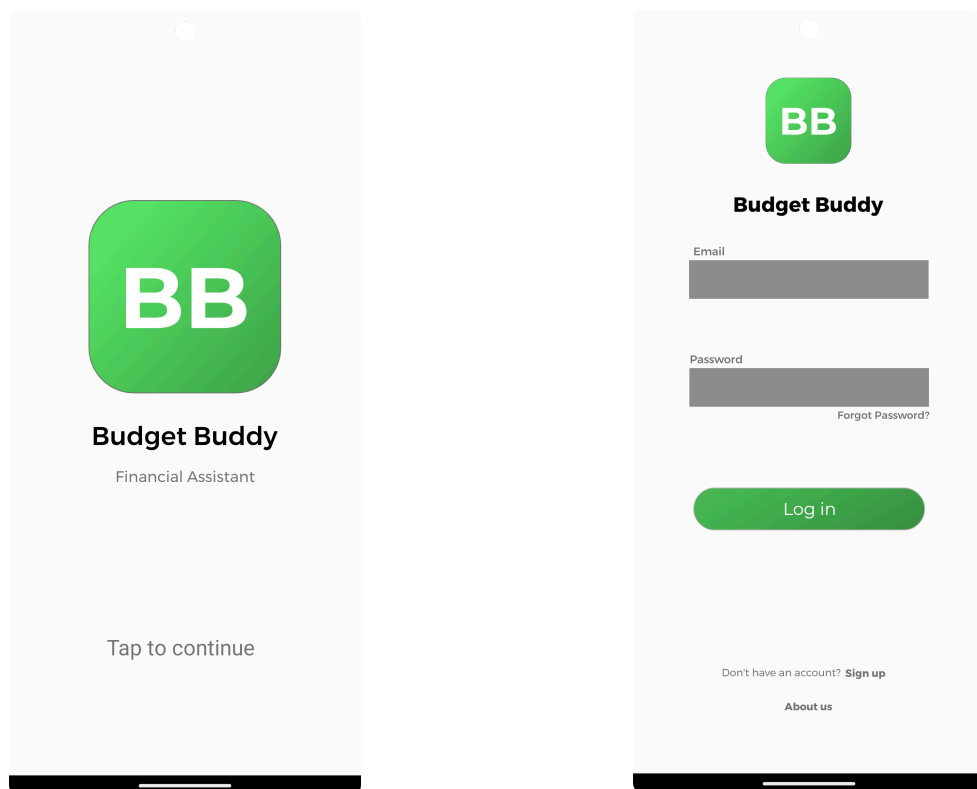
## DESIGN DIAGRAMS

### Graphical User Interface

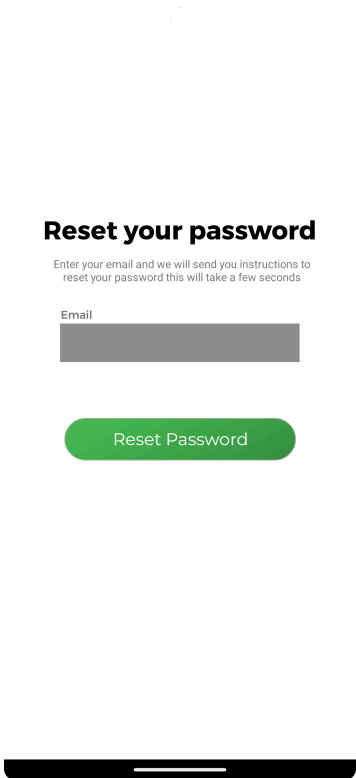
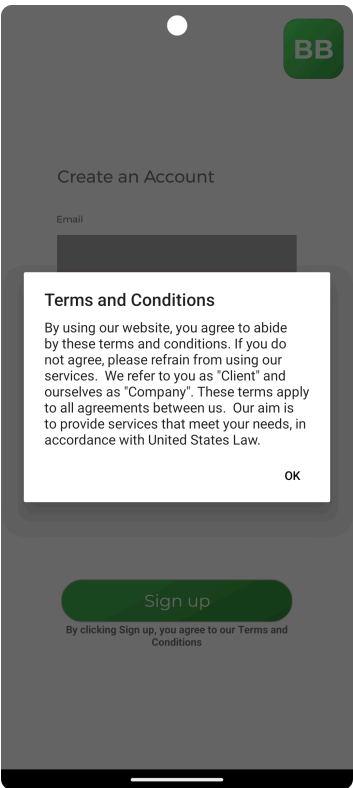
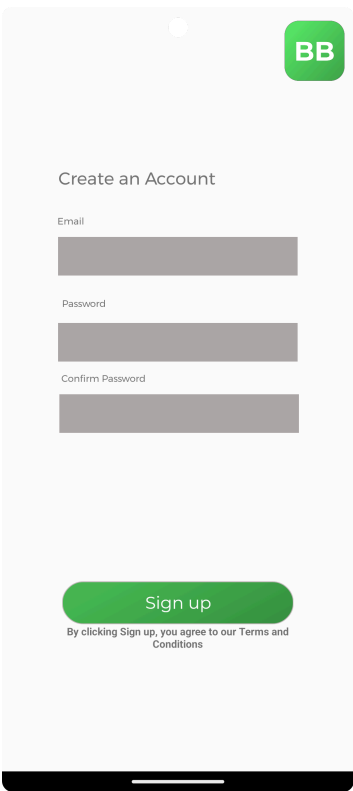
The Graphical User Interface shows what the user sees on the screen what they can interact with. Budget Buddy has multiple different screens in the app, here are the GUI's for each page in the app.

The Graphical User Interface (GUI) represents what the user sees on the screen and can interact with. Budget Buddy has a variety of different screens within the app, each designed to offer specific functionalities. Below are the GUI layouts for each page in the app, illustrating the user interface elements and their arrangement.

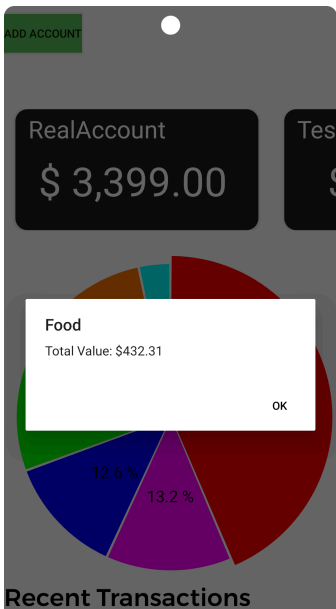
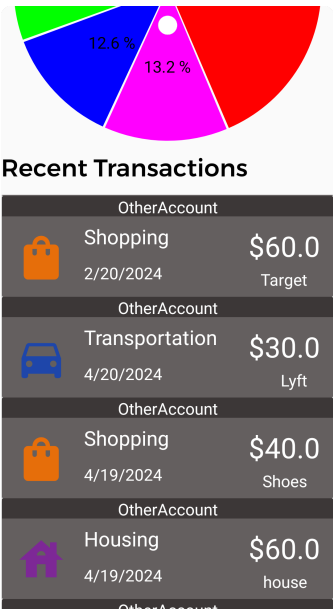
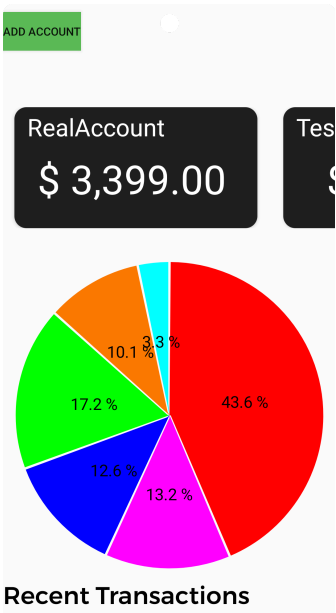
### Welcome & Log in



Create Account & Terms and Conditions



Home Screen



Add Account & Add Expense

8:12

Account Name

MainChecking

Starting Balance

1800

SAVE

Amount

19.99

Notes

Popeyes

Account

MainChecking

Date

4/27/2024

Category

Food

SAVE

Home

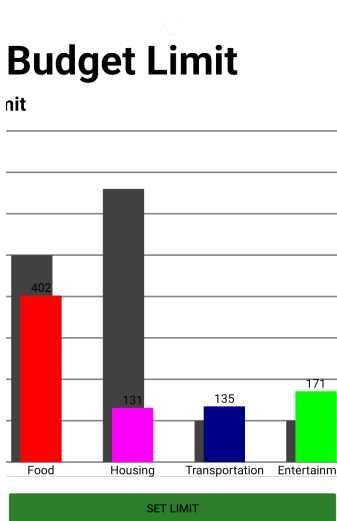
Bar Chart

Add Expense

Folder

User

Charts



WARNING! Transportation budget exceeded by \$34.55

Budget Limit

Limit

Set Limit

Food

500.0

Housing

660.0

Transportation

650.0

Entertainment

100.0

Shopping

90.0

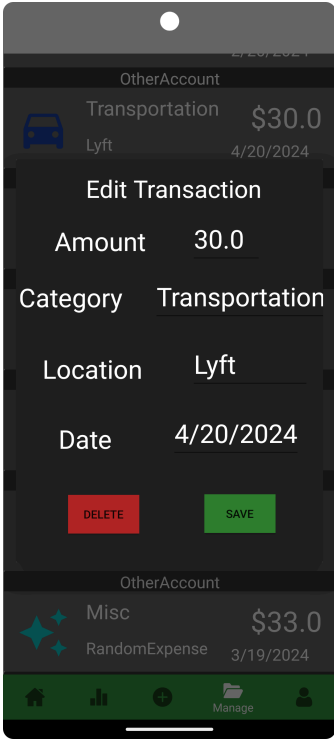
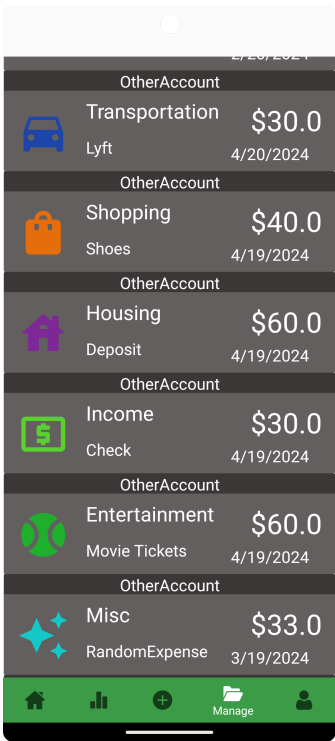
Misc

770.0

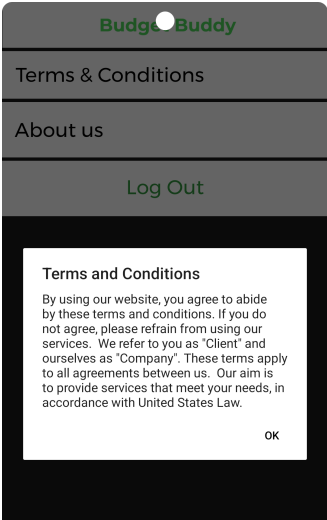
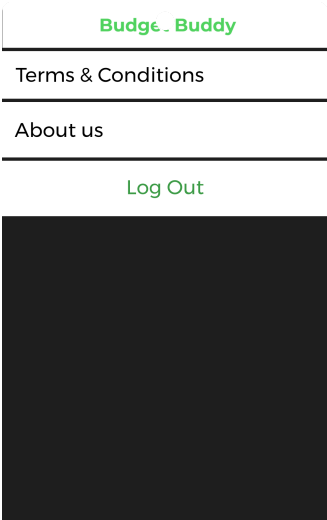
CLEAR ALL

SAVE

Manage



Profile



About us

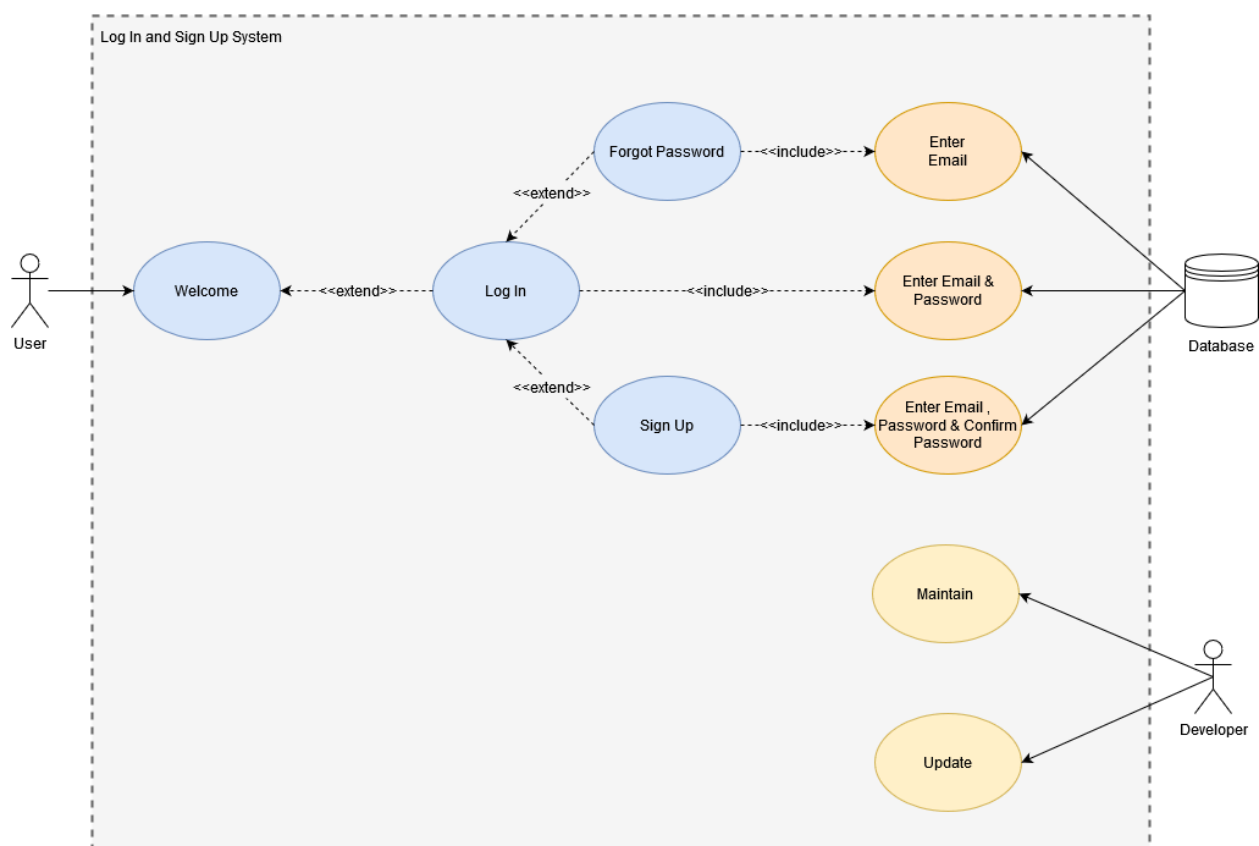
At Budget Buddy, we're dedicated to empowering users like you to take control of your financial well-being by providing intuitive tools to organize expenses and foster healthier spending habits. We are here to guide you every step of the way on your journey to financial freedom. Whether you're a budgeting novice or a seasoned pro, Budget Buddy is designed to streamline the budgeting process, making it easier than ever to track your expenses, set savings goals, and make informed financial decisions. With Budget Buddy by your side, you'll gain the confidence and clarity you need to achieve your financial goals and live your best life. Say goodbye to financial stress and hello to a brighter, more prosperous future with Budget Buddy!

### Use Case Diagrams

A use case diagram is a type of diagram that is used to represent the interaction between users or also known as “actors” and various parts of a system. Use case diagrams are useful for planning and analyzing on what the user and system will need from a high-level perspective.

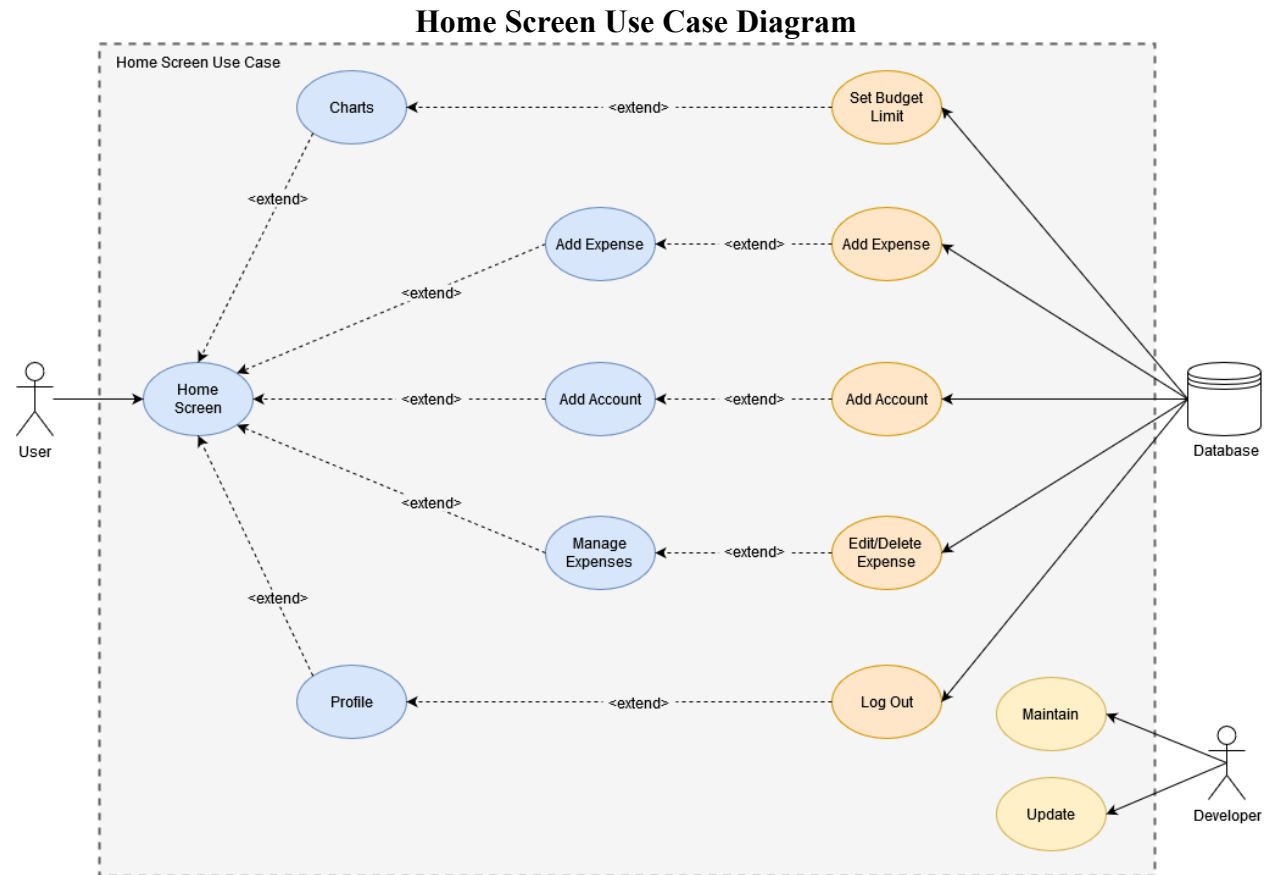
Budget Buddy has two use case diagrams, one demonstrating the log in process and the other for the main app functionality between multiple pages.

### Log In Use Case Diagram



The diagram above shows the use case diagram for the log in system. There are three actors in this diagram the user, database, and developer. The user being the one who is using the application. The database is a Firebase database. The developer is the one responsible for creating Budget Buddy and the one who will maintain the application. When the user first opens the application, they are greeted by the Welcome screen where they can then proceed to the Log In screen by tapping anywhere in the screen. From the Log In page the user can enter their email and password and the database will verify if their credentials match an account registered in the database. If the user does not have an account, they can go to the Sign-Up page where they will have to enter an email, password, and confirm their password. It will be sent to the database to make sure there is already not an existing account with the email. The user is also able to reset their password by accessing it in the Log in page where they will need to enter their email and the database will send an email to the corresponding email if it is registered in the database.





The diagram above shows user interactions with Budget Buddy's home screen after logging in. We have 3 actors present in this diagram again being the user, database, and admin. After logging in the user is brought to the home screen that provides the users with options of accessing add account, charts, add expense, manage expenses, and profile. Selecting "Add Account" allows the user to input their starting balance and name the account which will be stored into the database. The "Charts" option allows the user to view and set their budget limits, any changes made to the budget limits will be sent to the database. When selecting "Manage Expenses" the user is taken to a page where transaction details are retrieved from the database

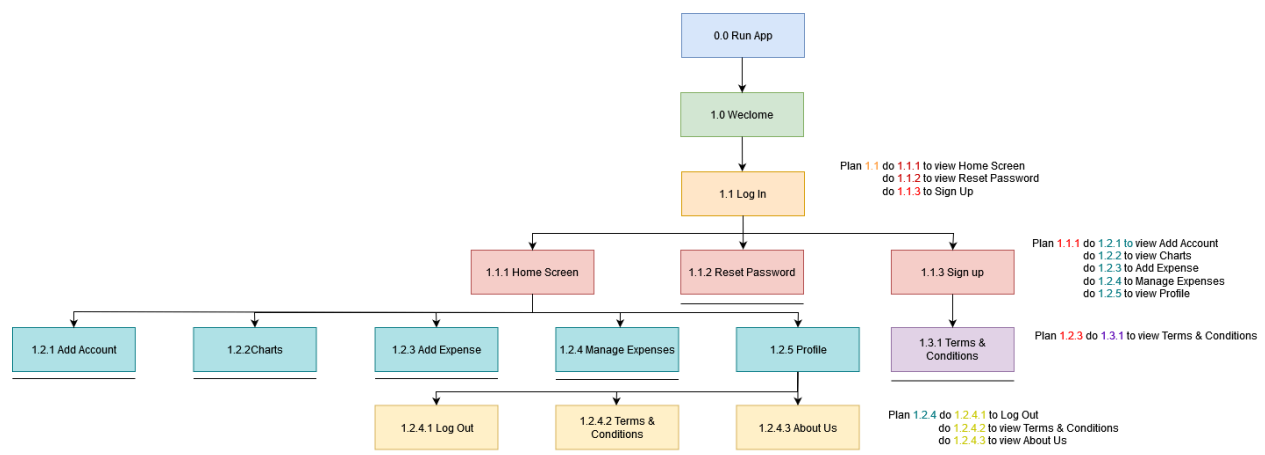
CHRISTIAN SOLIS CAPSTONE

allowing the user to edit or delete the transaction as needed. Finally, the “Profile” option offers a log out feature that returns the user back to the log in screen.

### Hierarchical Task Analysis

Hierarchical task analysis (HTA) is a diagram that is used to view tasks by breaking them down into other subtasks. It allows a clear detailed view of all tasks to understand what needs to be done to access the next subtask. By breaking down each task into subtasks it allows us to identify any redundancies that can be optimized.

#### Hierarchical Task Analysis Diagram

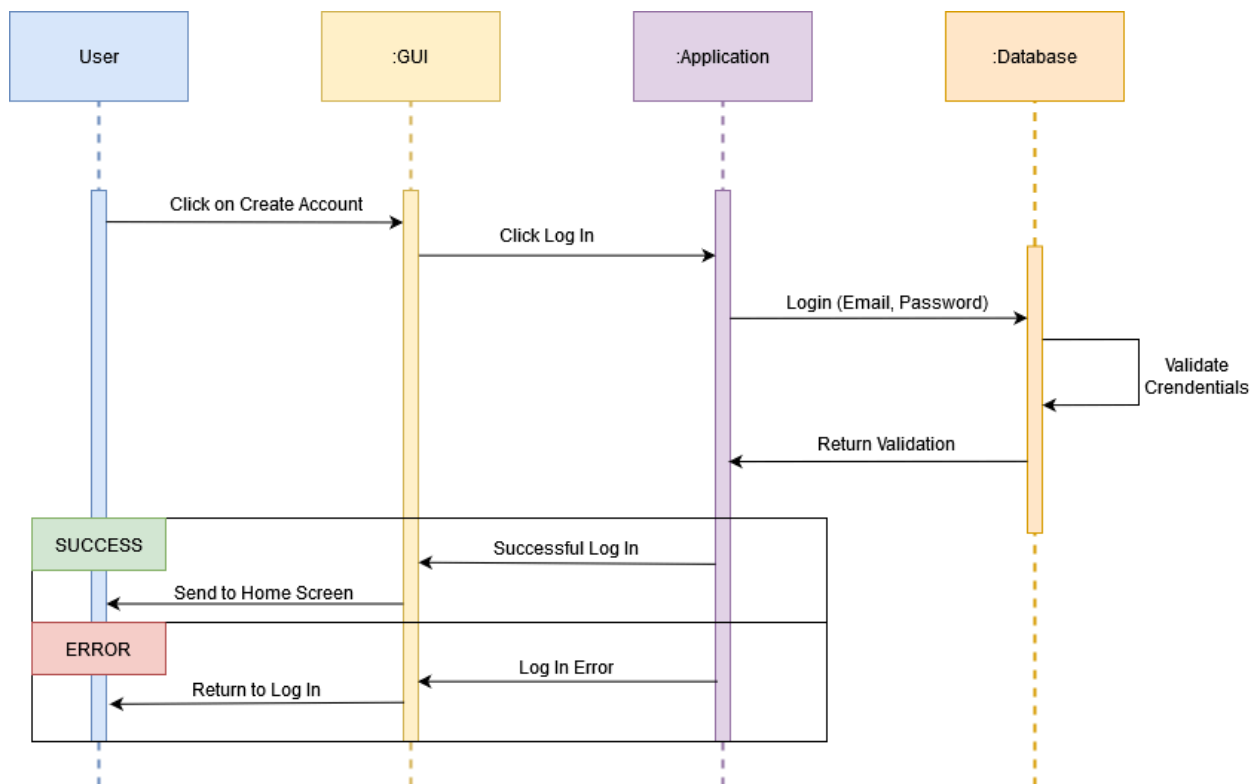


The diagram above presents the Hierarchical Task Analysis for Budget Buddy. It outlines all accessible tasks to the user and the hierarchy of accessing them. For tasks that have multiple options the diagram includes a plan specifying what each task does.

### Sequence Diagrams

Sequence diagrams are visual tools that visualize the interaction between different parts of an application in the order they occur. These diagrams demonstrate the flow of communication starting from the user's interaction with the application, GUI, application logic, and database. This makes it easier to understand the relationship between various parts of the application.

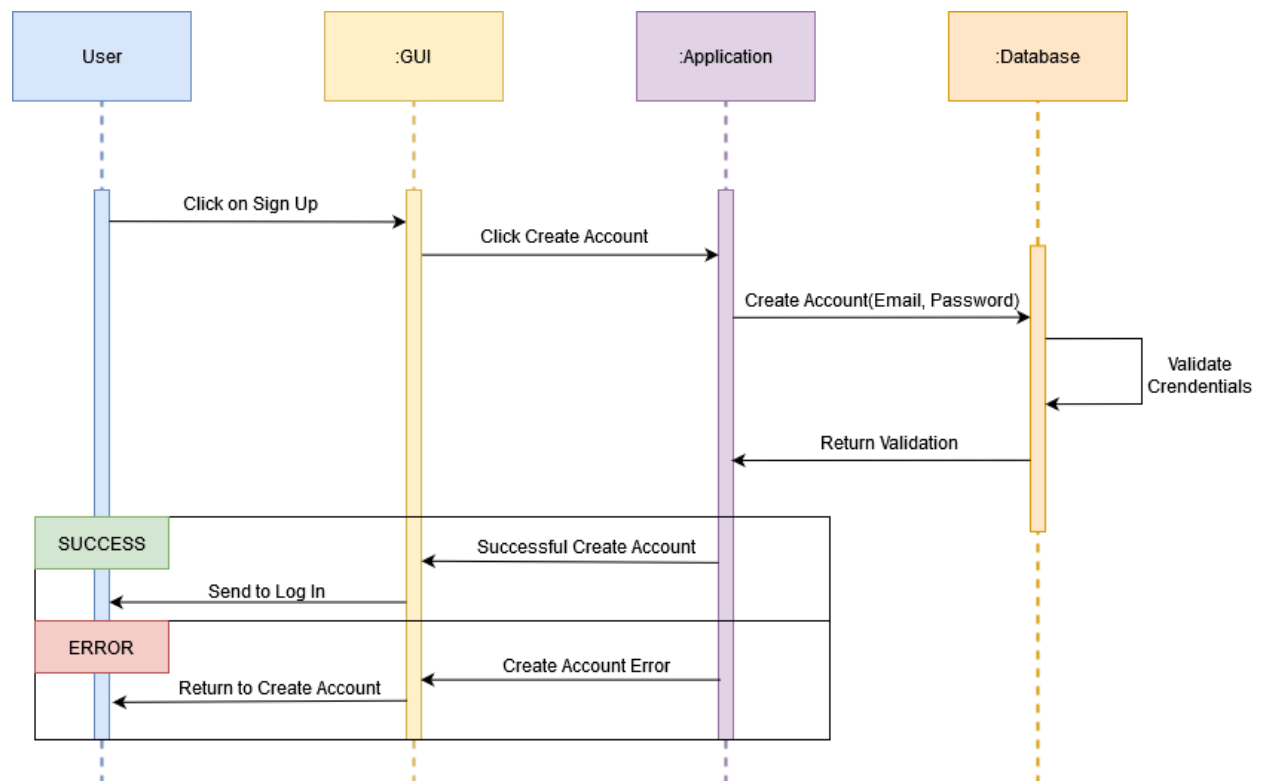
#### Log In Sequence Diagram



The sequence diagram displays the log in process within the application. Firstly, the user enters their credentials and clicks the log in button. This action triggers the application to send the credentials to the database. If the log in is determined to be successful, the user is sent to the

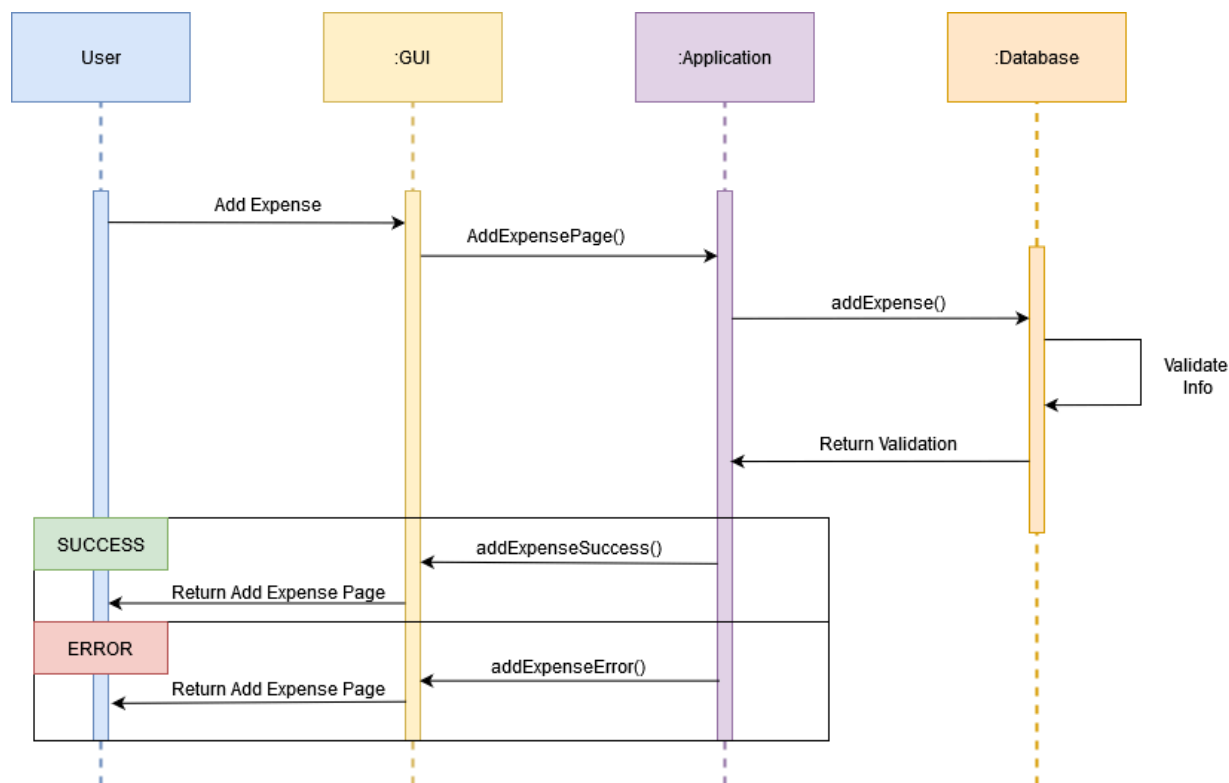
home screen. On the other hand, if the user's credentials do not match those in the database or any other occurs the user will remain on the log in screen.

### Create Account Sequence Diagram



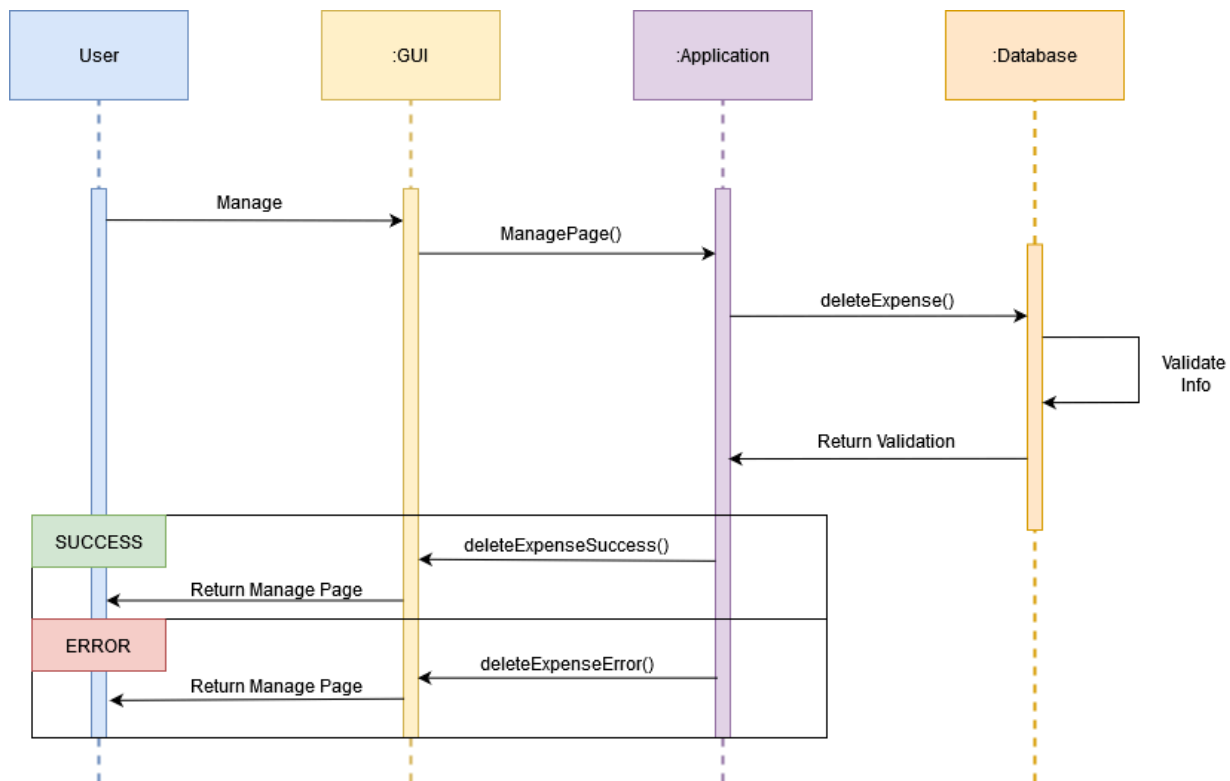
The sequence diagram above visualizes the create account process within the application. First the user clicks on sign up from the log in page, the GUI receives the request and sends the user to the create account page. Once the user has filled in the necessary credentials and clicked the create account button the information is then sent to the database to validate the credentials. A successful created account will send the user back to the log in page else if the create account is deemed to have an error the user will remain on the create account page.

### Add Expense Sequence Diagram



This diagram outlines the process of adding an expense in the application. From anywhere in the application the user selects the “Add Expense” option which will bring them to the corresponding page according to the GUI. After the user inputs the necessary credentials, the information is sent to the database. If the expense is successfully saved the user will be notified and remain on the “Add Expense” page to enter more items if wanted. On the other hand, if the expense is not saved successfully the application will notify the user and remain on the page.

### Manage Expense Sequence Diagram

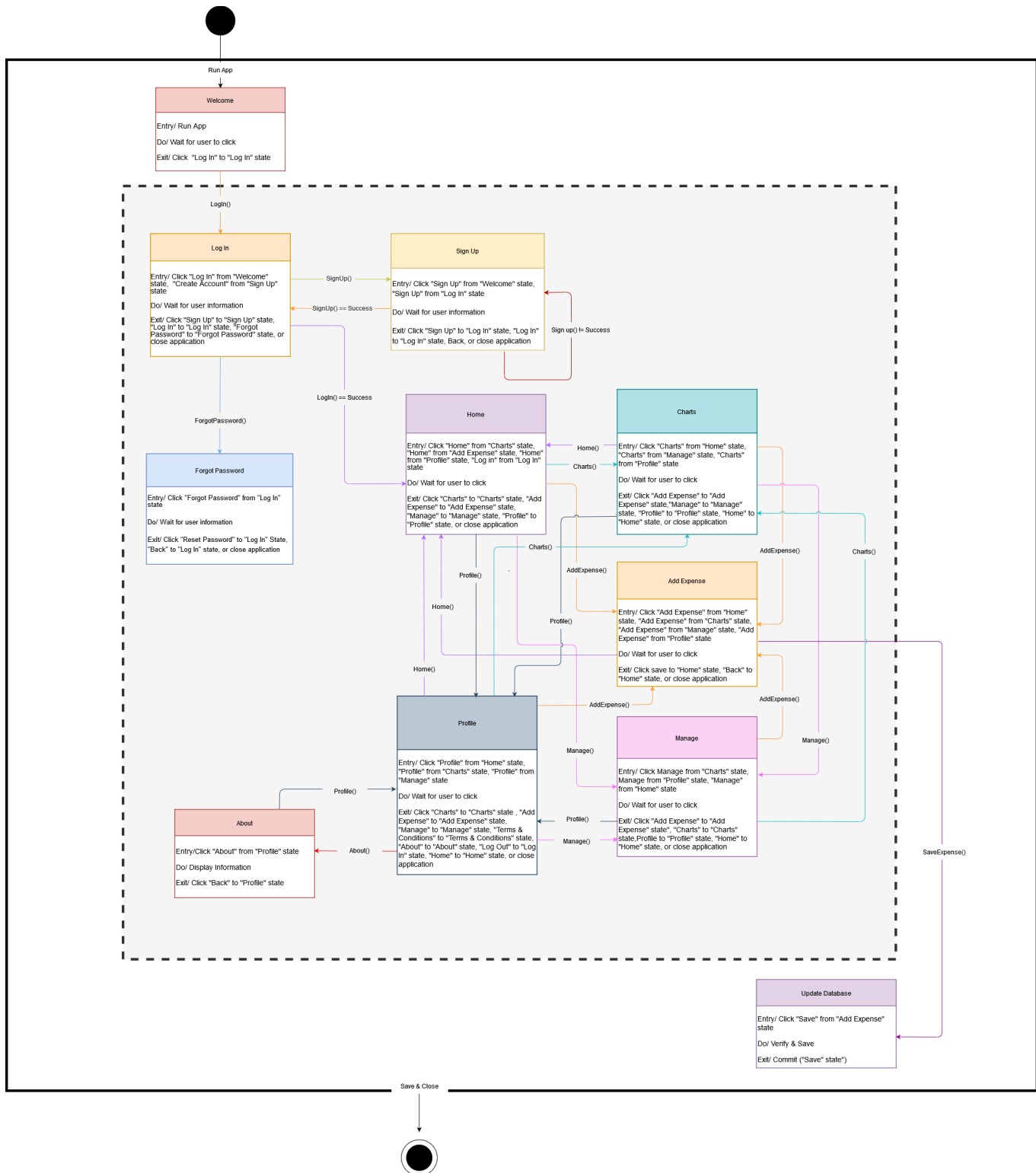


The sequence diagram displays the “Manage Expense” process within the application. When the user selects “Manage Expense” from any part of the application a request is sent and the user is navigated to the corresponding page. On this page when a user selects and item they are presented with all the related information. If the user chooses to delete the transaction, this is then sent to the database which will attempt to locate the transaction. If the transaction is successfully deleted the user will be redirected to the “Manage Expense” page. If the transaction fails to be deleted the user will be notified that there was an error.

**Finite State Machine Diagrams**

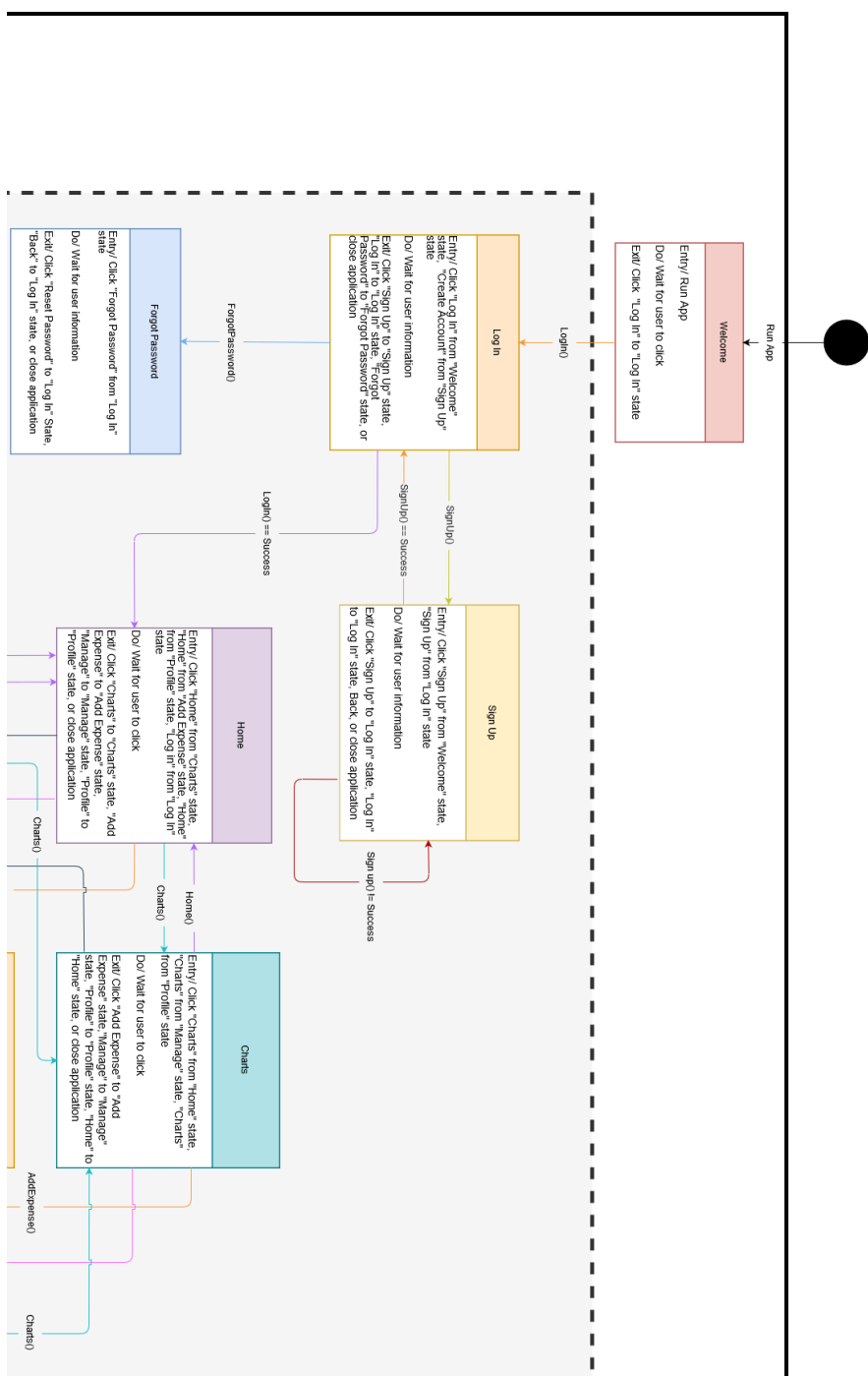
Finite State Machine Diagrams are visual representations that map out the various states within the application, demonstrating how each task or state is entered, the actions that can be performed within each state, and possible exit. Only one state in a finite state machine can be active at once. This structure is used to help the understanding of the flow of each state.

## Finite State Machine Diagram

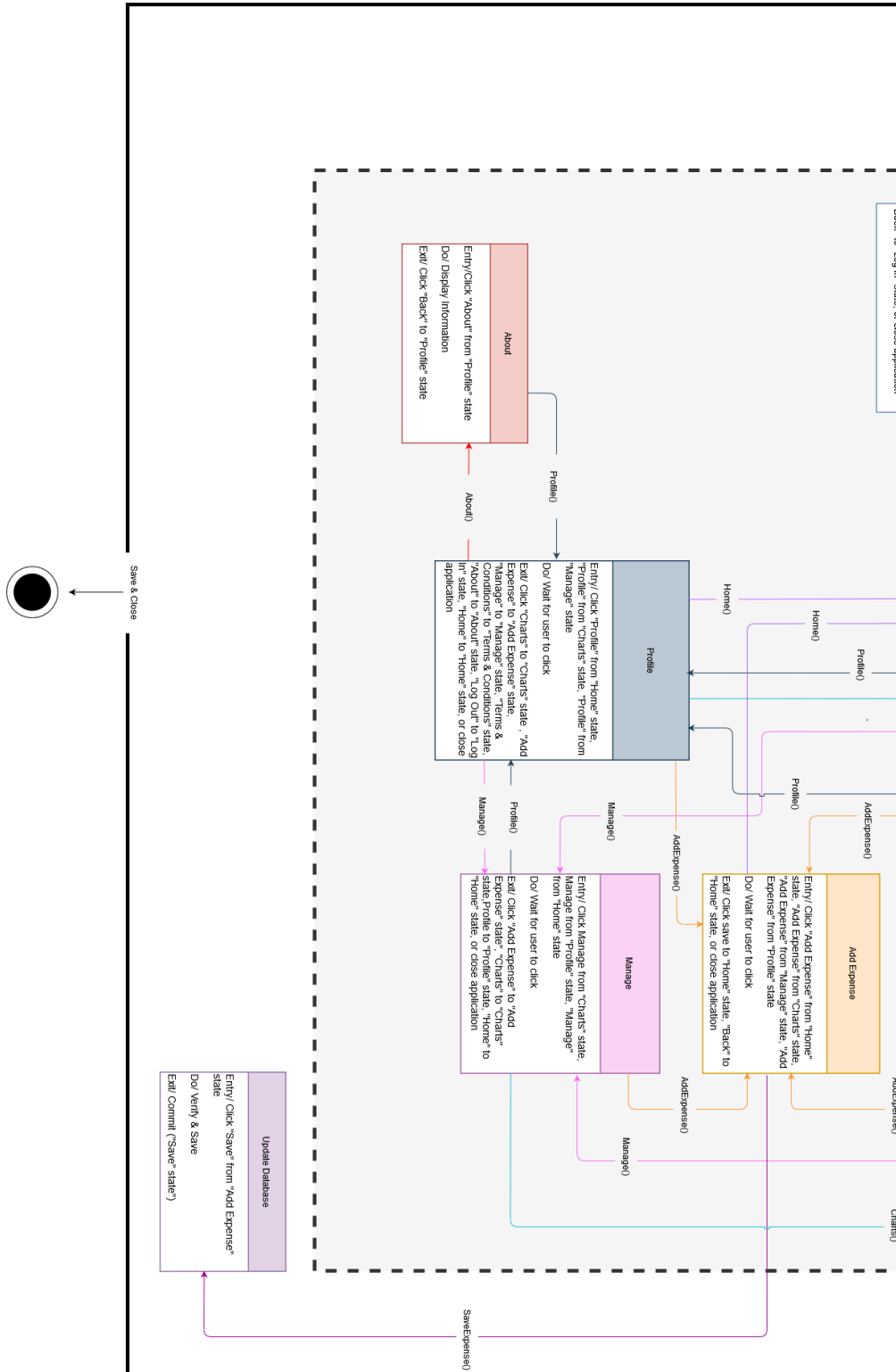




## Finite State Machine Diagram (Top Section)



## Finite State Machine (Bottom Section)



The diagram above is the finite State Machine Diagram that displays the different states in Budget Buddy showing how to enter each state and exit. This diagram is split into two sections for increased readability.

### **Finite State Machine Definitions**

The table below each state in the Finite State Machine Diagram is given a definition for each state and what can be performed. Each row color responds to the color in the Finite State Machine Diagram.

State	Description
Welcome	Welcomes users and directs them to the Log In state
Log In	Prompts user to enter account credentials or direct user to Sign Up or Forgot Password states
Sign Up	Prompts user to enter account credentials or direct user to Log In state
Forgot Password	Prompts user to enter their email or direct user to Log In state
Home	Displays account balance, pie chart, and recent transactions to the user. Access to Add Account State
Charts	Displays bar chart with budget limits to user and allows user to set budget limits
Add Expense	Prompts user to enter transaction amount, location, date, account, and category.
Manage	Displays all transactions to user and allows user to edit or delete transactions
Add Account	Prompts user to enter account name and balance
Profile	Prompts user with the option to access About Us state, display Terms & Conditions, and log out
About	Displays about us information to user

Welcome State Transition Chart		
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
Welcome	LogIn()	Log In
Welcome	Save() & Close()	Exit

Log In Transition Chart		
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
Log In	SignUp()	Sign Up
Log In	ForgotPassword()	Forgot Password
Log In	LogIn() == Success	Home
Log In	LogIn() != Success	Log In
Sign Up	Save() & Close()	Exit

Sign Up Transition Chart		
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
Sign Up	SignUp() == Success	Log In
Sign Up	SignUp() != Success	Sign Up
Sign Up	Save() & Close()	Exit

Forgot Password Transition Chart		
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
Forgot Password	ResetPassword()	Log In
Forgot Password	Back()	Log In
Forgot Password	Save() & Close()	Exit

Home Transition Chart		
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
Home	Charts()	Charts
Home	AddExpense()	Add Expense
Home	Manage()	Manage
Home	Profile()	Profile
Home	Add Account()	Add Account
Home	Save() & Close()	Exit

Charts Transition Chart		
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
Charts	AddExpense()	Add Expense
Charts	Manage()	Manage
Charts	Profile()	Profile
Charts	Home()	Home
Charts	Save() & Close()	Exit

Add Expense Transition Chart		
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
Add Expense	Home()	Home
Add Expense	Back()	Home
Add Expense	Save() & Close()	Exit

Manage Transition Chart		
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
Manage	Home()	Home
Manage	AddExpense()	Add Expense
Manage	Charts()	Charts

## CHRISTIAN SOLIS CAPSTONE

Manage	Profile()	Profile
Manage	Save() & Close	Exit

Add Account Transition Chart		
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
Add Account	SaveAccount()	Home
Add Account	Back()	Home
Add Account	Save() & Close()	Exit

Profile Transition Chart		
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
Profile	Home()	Home
Profile	Charts()	Charts
Profile	AddExpense()	Add Expense
Profile	Manage()	Manage
Profile	About()	About
Profile	Save() & Close()	Exit

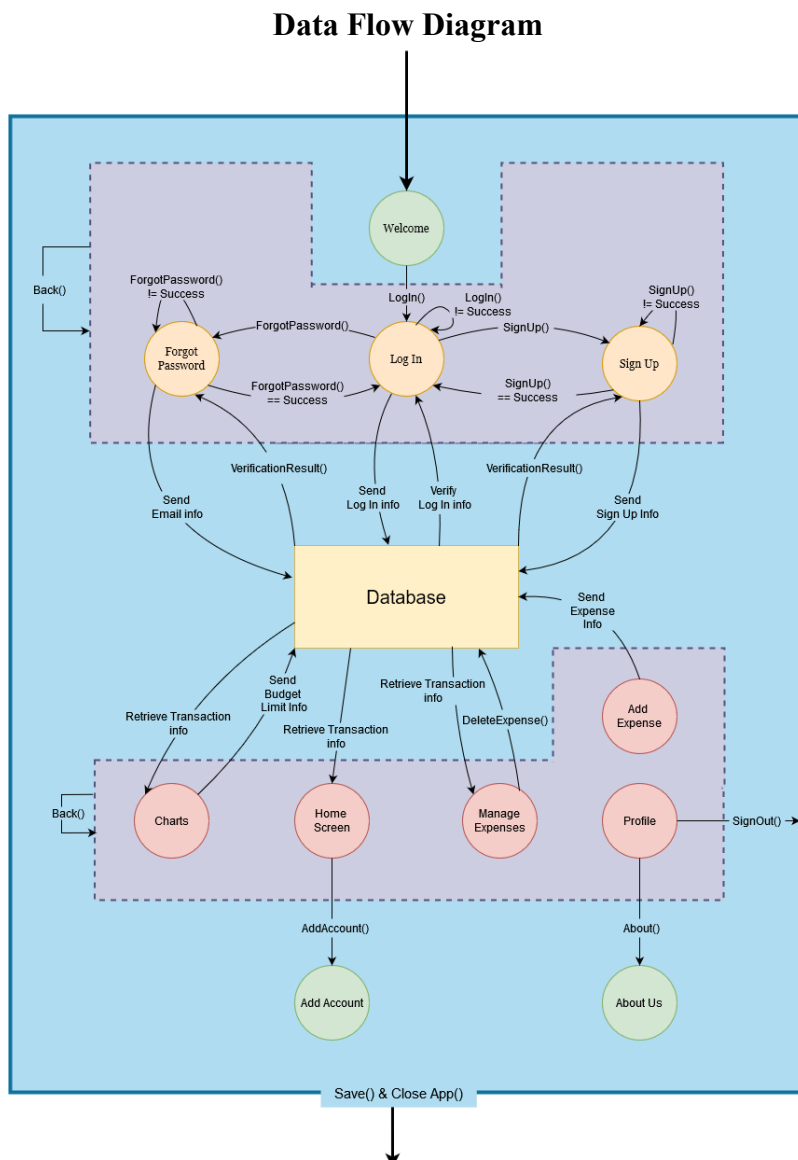
About Transition Chart
------------------------

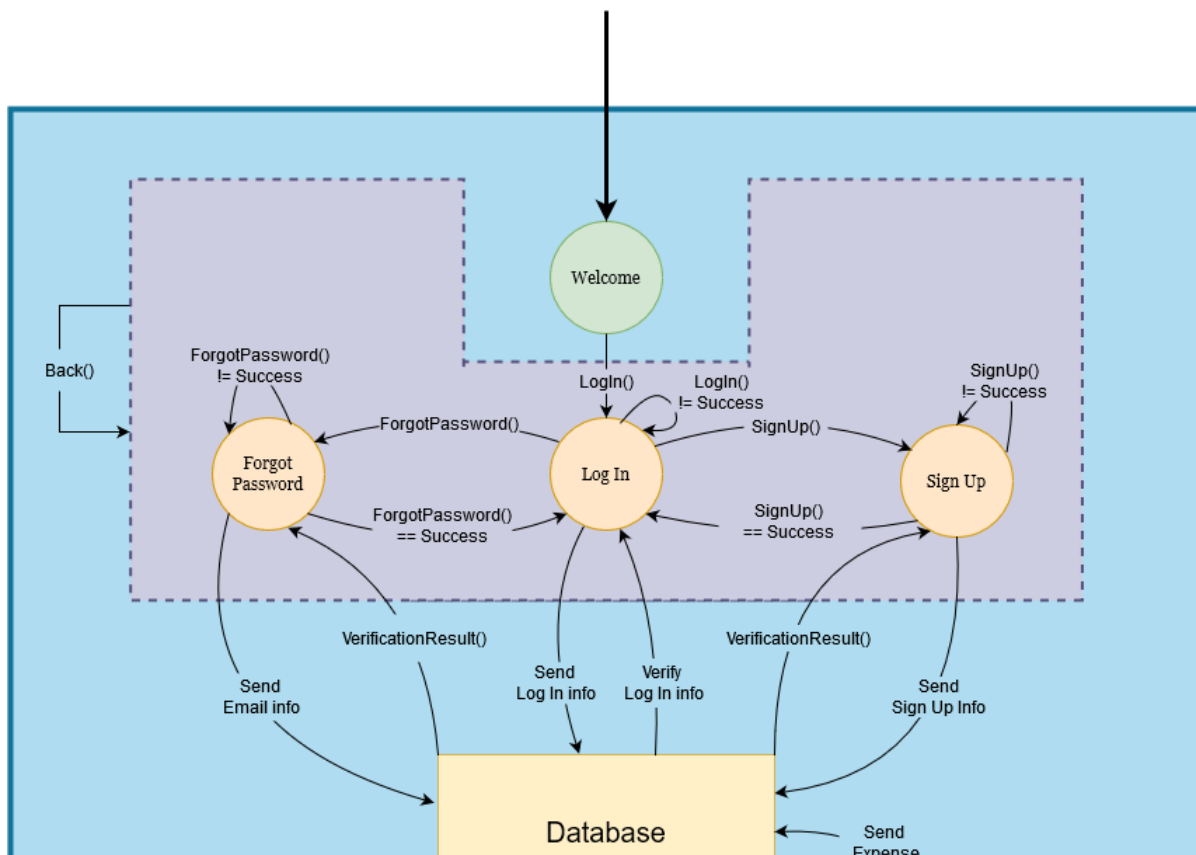


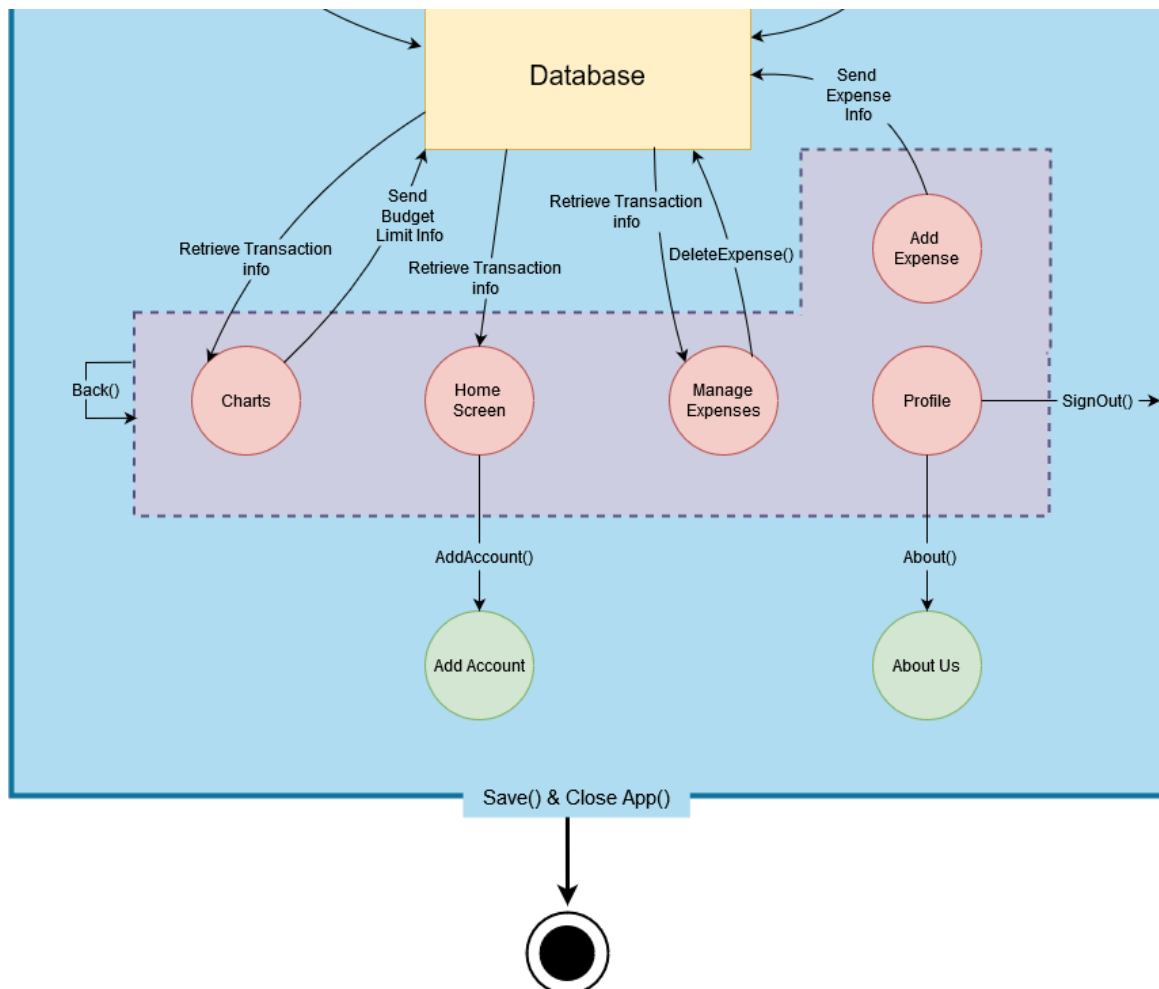
<u>Current State</u>	<u>Condition</u>	<u>State Transition</u>
About	Back()	Home
About	Save() & Close()	Exit

## Data Flow Diagrams

Data Flow Diagrams (DFD) are vital tools for visualizing how data moves throughout an application. These diagrams demonstrate the flow of data between different components of the application, specifically showing how data transfers between different states and database.



**Data Flow Diagram(Top Section)**

**Data Flow Diagram(Bottom Section)**

The diagram above is the Data Flow Diagram of Budget Buddy. It is divided into two sections for readability. This diagram shows how data moves throughout the application between different states and database. The purple box groups all the states that can be accessed after the Welcome state. The bottom section of the diagram groups all the other states that can be accessed from the current state. Add account and About state are only accessible from one state. Add account is only accessible from the Home state and About is only accessible from the Profile state.

## **CLASSES**

### **Class Explanations**

#### **AboutUs.java**

AboutUs.java is an activity that can be accessed from the Profile page when the user is logged in. This page gives the user information about the application and why it was created.

#### **AccountAdapter.java**

AccountAdapter.java is responsible for formatting a model class that will be used for interface. In this case this class is used for a recycler view. It takes in an object of an account model that consists of an account name and balance. The class takes the balance and formats it into a decimal number and rounds it the hundredth place in an event where a user might input a larger decimal.

#### **AccountModel.java**

AccountModel.java is designed to represent individual accounts in Budget Buddy. It receives the necessary values to set up the account, being the account name and balance. The class provides a constructor to initialize these values as well as getter and setter methods.

**AddAccount.java**

AddAccount.java is used to create and add brand new accounts to Budget Buddy. When the user clicks the save button the class will check if the account name or starting balance is not empty or that the starting balance is not negative. If all the credentials are valid the database will be called and allocated the brand-new account. If the task is successfully finished the user will be prompted with a toast message letting the user know, if not the user will be prompted with the error message.

**BudgetLimitModel.java**

BudgetLimitModel.java is designed to model the individual budget limits for each category. It receives six different double values that each represent a category food, housing, transportation, entertainment, shopping, and miscellaneous. The constructor initialized these variables along with getter and setter methods for each variable.

**ChartsPage.java**

ChartsPage.java is responsible for allowing the user to view their budget limits and set them as well in the same page. When the page is first accessed a setUpBarChart() method is called to set up the budget limit chart. When setting up the bar chart a call is made to the database to retrieve any budget limit values. If the user happens to go over their budget limit a warning message will appear on screen notifying the user how much they have gone over their budget.

**CreateAccount.java**

CreateAccount.java uses firebase authentication for user registration. When a user enters their credentials, the program will validate them by ensuring that the email or password is not empty as well that the password is matching. The program will then make a call to the database

CHRISTIAN SOLIS CAPSTONE

to ensure that the account is not already in it. If the account creation is successful, then the user will be sent back to the log in page.

### **ExpensePage.java**

ExpensePage.java is responsible for receiving input from the user to add a transaction.

The class receives multiple variables for amount, account, category, location, and date. The accounts are retrieved from the database that are currently initialized. If a transaction is successfully saved a message will be sent to the user letting them know, if it is unsuccessful an error message will be sent to the user. Once the transaction is saved all the currently filled in data fields will be reset and emptied.

### **ForgotPassword.java**

ForgotPassword.java allows the user to reset their password if they have forgotten it or if they choose to. For the user to reset their password they must provide an existing email that is registered with an account. If the database can locate the email a reset link will be sent to the user via email.

### **HomePage.java**

HomePage.java is the first page the user will see when they successfully log in from the log in page. On this screen the user will be able to see the total value of each of their transactions via a pie chart. This is also where the user can add an account and be sent to the add account page, once successfully added they will return to the home page and will appear . Lastly at the bottom of the page the recent transactions will appear these are received directly from the database.

### **LogIn.java**

LogIn.java is the class responsible for handling the user's credentials. To log in the user must provide an account with a registered email and password that has already been set. If the process is successful, the user will be directed to the home page. The user can go to the sign-up page by clicking the "Sign Up" text. The user is also able to reset their password if they like by clicking "Forgot Password?" text.

### **MainActivity.java**

MainActivity.java is the first page the user will see and is responsible for handling the welcome screen. For the user to continue they can tap anywhere on the screen. This is handled by an on click listener that will send the user to the log in page once initiated.

### **ManagePage.java**

ManagePage.java is where the user will be able to view all their transactions. When a user clicks on a item on the page that is set by the recycler view a dialog box will prompt the user with the transaction details. If the user would like to edit their transaction details, they can input the new details and click "Save" if the user wishes to delete the transaction they can select "Delete" and a call will be sent to the database to delete the transaction once found.

### **ProfilePage.java**

ProfilePage.java will allow the user the ability to log out. If the user wishes to log out they can click "Log Out" where they will be sent to the log in page. To view the terms and conditions the user must click "Terms & Conditions" where a dialog box will appear. Lastly if the user clicks "About Us" the user will be sent to the about us page.

**ReadWriteUserDetails.java**

ReadWriteUserDetails.java is responsible for sending user details. This is set up with a constructor that initializes it.

**TransactionAdapter.java**

TransactionAdapter.java is responsible for formatting the transaction information within a recycler view. When creating an item view the adapter is called to format the category, amount, date, account, and location. When an “onItemClick” method is called the interface is called that will format the transaction details.

**TransactionModel.java**

TransactionModel.java represents a single transaction in Budget Buddy that holds multiple variables such as icon, category, amount, date, location, and transactionID. These variables are initialized through the constructor with corresponding setter and getter methods.

**TransactionRecyclerViewInterface.java**

TransactionRecyclerViewInterface.java this is used to handle any click events that take place in the application. This interface has a “onItemClick” signature to handle any click events. This interface allows communication between a recycler view and the activity.

**TESTING**

Testing is of the highest importance in the development process of an app. It is all about finding and fixing bugs both small and big at the pre-launch stage in the market. Comprehensive testing involves individuals of different degrees of familiarity with the app. There are three major tests, and they are designed at the following levels. Black-box testing is the testing done by some group of testers who don't know at all how the application works from inside. Grey-box testing is



CHRISTIAN SOLIS CAPSTONE

executed by a person aware of the project, who does not have full acquaintance with details of the implementation mostly, the member of a company team is included in testing but without direct participation in the development of the application. On the other hand, white-box testing is performed by developers with information regarding the application up to its architecture and code. Each of such approaches to testing provides clear insights that help to keep the operation of the app smooth across the described diversity of user interaction and integration with different interacting systems.

### **Tests Used**

#### **Black-Box**

Black-box testing is done by a user that has no knowledge of how the app functions.

They have no access to the code and can only interact with the applications GUI.

An example of black-box testing done for this application was having a user create an account and sign in. This test was done by a friend with no knowledge of computer science or how the app was implemented.

The first prerequisite is that the user must have an email to sign in. We will be testing that users will be able to enter their email and password to create an account. The expected results are that the user will be able to create an account and be returned to the log in page. Lastly, they will be able to fill in their credentials and click sign in to access the rest of the application.

### **Grey-Box**

Grey-box is done by a user who has some knowledge of how the application functions, but they do not know every specific part of the application. This testing could be done by someone who has some knowledge or experience in computer science or mobile development.

Grey-box testing done for this application was to test the add expense functionality by entering the necessary credentials.

The first prerequisite is that a user would need to be signed in to the account. What is expected is that the user will fill in the necessary details and click the save button. Once the user clicks the save button it is expected that the database will save the transaction.

### **White-Box**

White-box testing is done by a user who knows exactly how the app is implemented and has access to the programs code. This testing is usually done by the developer since they know the apps implementation.

An example of white-box testing done for this application is having the user edit and delete transactions.

The prerequisite for this test is that the user already has an existing transaction. We will be testing that the user is able to click the item in in the manage page that will prompt the user with a dialog box. Once the dialog box pops up the user will try to edit the information or delete it if they choose to. What is expected is that the database will be able to retrieve the transaction information correctly and be able to update the information when edited or deleted.

### **CONCLUSION**

In conclusion, Budget Buddy would make for a great mobile app that allows users to keep track of their expenses. It outperforms other applications by taking features from other applications and improving on them. Budget Buddy focuses on delivering a user-friendly experience by simultaneously removing ads to increase user experience. By allowing users to manually import their accounts they can keep track of other expenses that might not be associated with a bank. The various charts provided allow the user to get more insight into their spending habits. Being able to set a budget limit on various categories allows the user to manage their money in a more detailed way. Users can easily input their expenses and label them to their liking by using the notes section. Lastly being able to edit or delete any transaction logged into the application allows for further customization for any accidental transactions or mistakes when logging. Using agile development allowed me to receive and implement any feedback. Each sprint was beneficial to the application improvement. Future work would include to use the “Plaid” API so users are able to connect their banks, integrating a save feature that would allow users to save aside money for a goal, and lastly set up a family or group feature that allows users to view others transactions.

## REFERENCES

- [1] Google. (n.d.). Money Manager - Apps on Google Play. Retrieved from  
<https://play.google.com/store/apps/details?id=com.dayspringtech.envelopes>
- [2] Google. (n.d.). Mint: Budget & Track Bills - Apps on Google Play. Retrieved from  
<https://play.google.com/store/apps/details?id=com.mint>
- [3] Google. (n.d.). Wallet: Budget Expense Tracker - Apps on Google Play. Retrieved  
from  
[https://play.google.com/store/apps/details?id=com.droid4you.application.wallet&hl=en\\_US&gl=US](https://play.google.com/store/apps/details?id=com.droid4you.application.wallet&hl=en_US&gl=US)
- [4] Gershon, L. (2022, July 14). Budgeting Statistics: By the Numbers. CreditDonkey.  
Retrieved from <https://www.creditdonkey.com/budgeting-statistics.html>
- [5] Consumer Financial Protection Bureau. (2022, March 29). CFPB Finds Credit Card  
Companies Charged \$12 Billion in Late Fee Penalties in 2020. Retrieved from  
<https://www.consumerfinance.gov/about-us/newsroom/cfpb-finds-credit-card-companies-charged-12-billion-in-late-fee-penalties-in-2020/>