**My Financial Pal**

**20/20 Vision**

**Parker Tuck**

**Allan Lieu**

**Adam Spinhirne**

**Travis Nguyen**

**SOFTWARE DESIGN DOCUMENT**

**Version 1.0**

**INDEX**

|  |  |
| --- | --- |
| Title | Page No. |

[**1.0 Introduction**](#_rrne3moej7a9) **3**

[**2.0 Overview of Architecture**](#_w4mmyueab0sx) **3**

[**3.0 Use Cases**](#_dgwm7y6sjd17) **5**

[A. Use Case Diagram](#_o3fd55m59n89) 5

[B. Use Case Specifications](#_fpe8ae8fdf0s) 6

[i. UC1 - Input Transactions](#_70mcaiwvyd6x) 6

[ii. UC2 - Manage User’s Account](#_t148itbs3dlj) 6

[iii. UC3 - Create Budget Item](#_3vgiaw9834ot) 7

[**4.0 Classes and Their Interactions**](#_1y3d07v1el2a) **8**

[A. Class Diagram](#_xfp5d9hp5pys) 8

[B. Details of Classes](#_vo0yvftq580o) 9

[i. C1 - User Class](#_xh4ny5ltd4jx) 9

[ii. C2 - Budget Class](#_ef4u5btlqpp) 10

[iii. C3 - Transaction Class](#_m22h9sa7ump9) 11

[iv. C4 Account Class](#_4r3zspdjnz6u) 12

[C. Sequence Diagram](#_gfzchh48031m) 13

[**5.0 State Transition Diagram**](#_pq92m89vzbyk) **14**

[**6.0 User Interface Design**](#_rt50jvqrphbo) **15**

[**7.0 Conclusion**](#_ehqbkukctezt) **17**

# 

# 1.0 Introduction

My Finance Pal is an interactive application that is used to analyze a user’s profits and expenses in a way that will ultimately save money for that user. This document will describe how the application should interact with the user (upfront and behind the scenes), and how the user interacts with the applications display/accessibility.

The design of this project is important to our group because it allows the developers to look back at the design as a sort of structure for how the project should be implemented. When a conflict rises, looking at the design should provide clarification for what will be implemented next, or how the product should look, or what the functions should do.

This document will begin by first describing the architecture of our design and why we chose to design it that way (Section 2). Next, in Section 3, multiple figures and tables will show the primary use cases that will be implemented in our application (use case diagram and multiple use case specifications). Following that, there will be a short description accompanied by a few figures, that will discuss the multiple classes used by the application (Section 4). The remaining two sections, Sections 5 and 6, will depict and describe the basic states of the application (state transition diagram) and the user interface.

# 2.0 Overview of Architecture

We chose to create our project using the Service-Oriented architecture model. We have many pieces of functionality that can work together or be called separately by the user to complete different tasks as needed. For instance, we have a “Transactions” module that is used directly by the user when they input their income or spending, but this module is also used by the “Budgeting” module for its processes.

This is a suitable architecture for our project because of both the functionality requirements of the app and the simplification for development. The client-server architecture parts are needed in order to save client data for later retrieval. We are using the component-based parts of our project to simplify more complex operations that our app needs to complete. We can then use these more simplified components to accomplish repetitive tasks, and tasks that are similar but called from different sources.

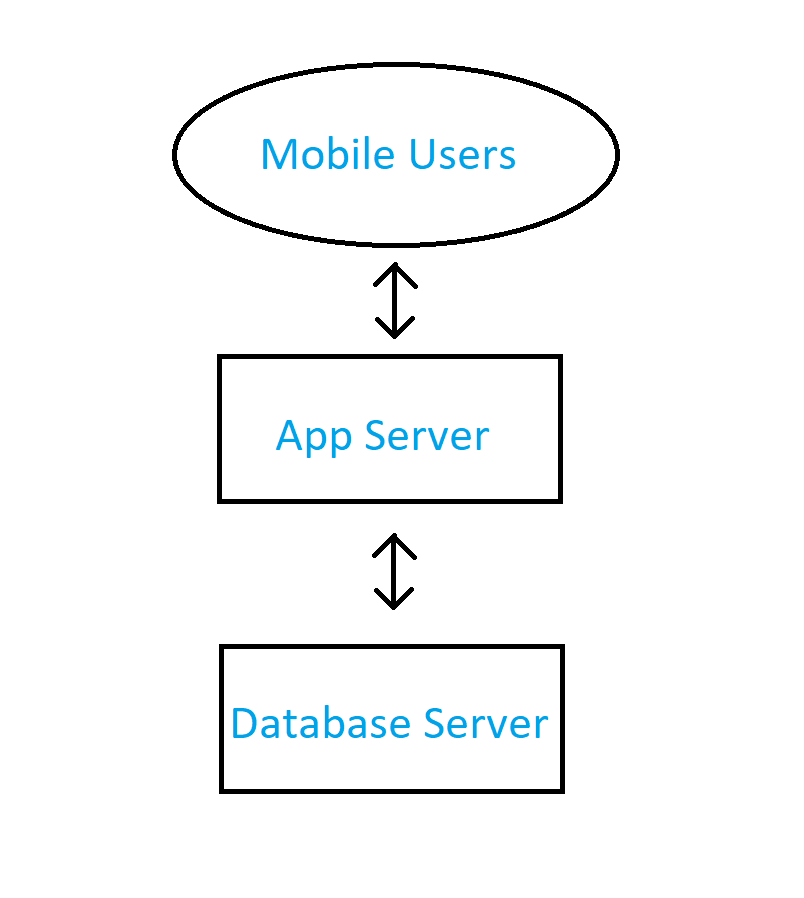


Figure 1. Architecture

# 3.0 Use Cases

## A. Use Case Diagram

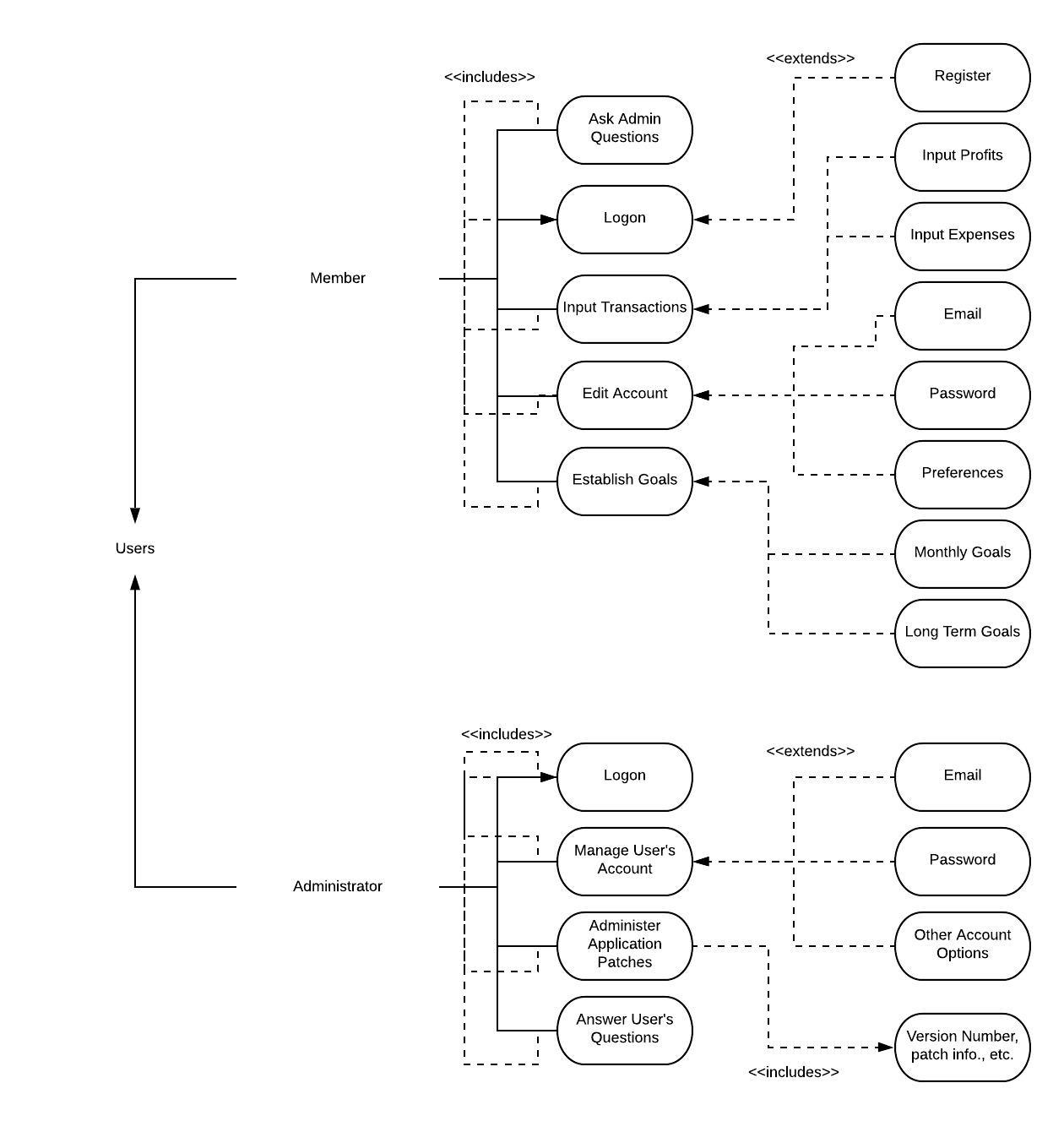


Figure 2. Use Case Diagram

## B. Use Case Specifications

### i. UC1 - Input Transactions

|  |  |
| --- | --- |
| Use Case Name: | Input Transactions |
| Overview: | Members can input/edit/remove any and all transactions made. This includes profits and expenses. |
| Type: | Primary |
| Actors: | Member |
| Pre-Condition: | Member must be logged on the app, and they must have selected the option to input a profit or expense transaction. |
| Main Flow: | 1. System requests members username/password 2. Member enters login information 3. System checks if login credentials are valid 4. Member selects to input a profit or expense transaction 5. Member enters transaction information (amount, date, location, category, if recurring, and comments) |
| Alternate Flow: | 2. Member selects option to create a new account |
| Alternate Flow: | 3. Login credentials are invalid |
| Alternate Flow: | 4. Member selects any other options from the homescreen |
| Post-Condition: | True |
| Cross-Reference: | System |

### ii. UC2 - Manage User’s Account

|  |  |
| --- | --- |
| Use Case Name: | Manage User’s Account |
| Overview: | An administrator can edit a user’s email, reset their password, or edit any other options on the user’s account upon request. |
| Type: | Primary |
| Actors: | Administrator |
| Pre-Condition: | Administrator must be logged on to the servers, and the user must specify what they done to their account in the request sent. |
| Main Flow: | 1. System requests admin’s username/password 2. Admin enters login information 3. System checks if admin credentials are valid 4. Admin changes whatever the member requests 5. Updates are sent to the member’s email and an in-application notification is sent to show that the changes have been made |
| Alternate Flow: | 3. Login credentials are invalid |
| Post-Condition: | True |
| Cross-Reference: | System |

### iii. UC3 - Create Budget Item

|  |  |
| --- | --- |
| Use Case Name: | Create Budget Item |
| Overview: | Customer will input a name and monthly amount of money they are spending on an item. (e.g. Food, Rent, etc.) |
| Type: | Primary |
| Actors: | Member |
| Pre-Condition: | Member must be logged on the app, select Create Budget |
| Main Flow: | 1. System asks user for a name/purpose for budget 2. User enters the name/purpose 3. System asks user for a monthly budget amount 4. User inputs the monthly cost 5. System registers the new budget |
| Alternate Flow: | User cancels during budget creation, Return to previous screen. |
| Alternate Flow: | User inputs an invalid amount for their budget (e.g. -3000) |
| Post-Condition: | True |
| Cross-Reference: | System |

# 4.0 Classes and Their Interactions

## A. Class Diagram

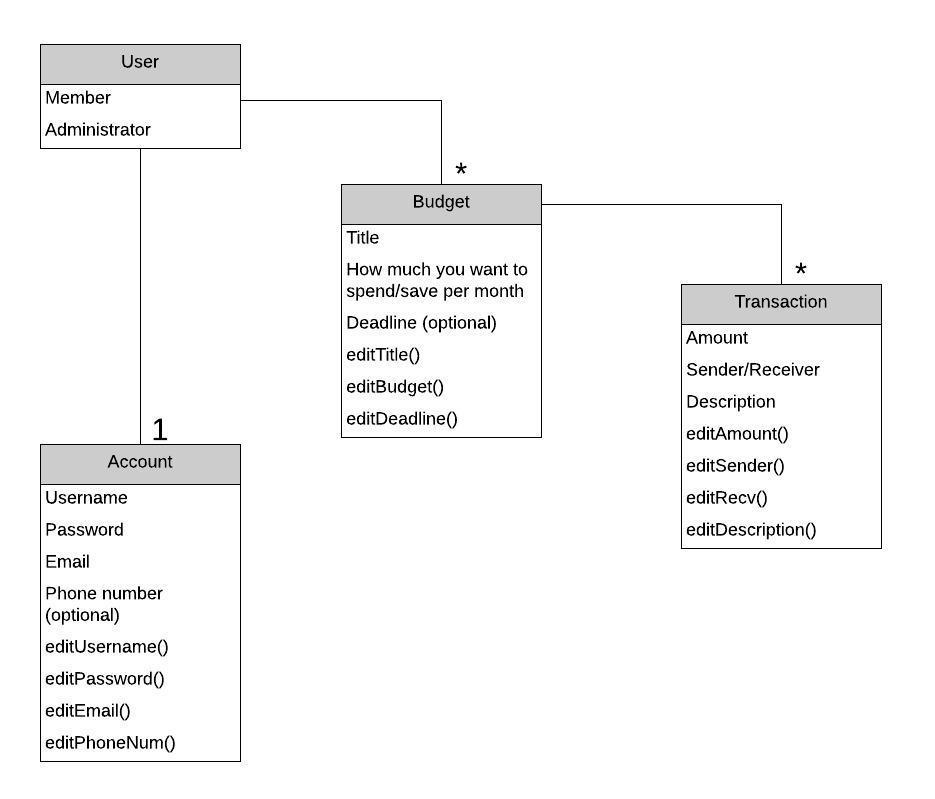


Figure 3. Class Diagram

## 

## 

## B. Details of Classes

### i. C1 - User Class

Attributes:

|  |  |  |
| --- | --- | --- |
| Attribute name | Type | Description |
| Name | String | Variable use to store name of Users |
| UserType | Bool | A boolean to check if the user is a member or an administrator |

Operations/Methods:

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Arguments passed | Expected return value | Description |
| get\_name | None | String | A function used to get the name of a user |
| assignName(name) | String | Null | A function to assign/update User’s name |
| CheckUser | None | Bool | A function to return a boolean value which indicates User or Admin |

### ii. C2 - Budget Class

Attributes:

|  |  |  |
| --- | --- | --- |
| Attribute Name | Type | Description |
| Title | String | Variable to store Budget Name |
| Budget | Float | Variable to store dollar amount |
| BudgetCategory | String | Variable to store and separate different types of budgets |
| Deadline | Date | Variable used to set a goal deadline for the user |

Operations/Methods:

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Arguments Passed | Expected return value | Description |
| editTitle(title) | String | Null | Set/Change Budget title name |
| editBudget(amount) | Float | Null | Set/Change Budget amount |
| editDeadline | Date/Int | Null | Set/Change Deadline Date |
| editCategory | String | Null | Set/Change the Budget Category(s) |

### iii. C3 - Transaction Class

Attributes:

|  |  |  |
| --- | --- | --- |
| Attribute Name | Type | Description |
| Transaction Amount | Float | Variable to store Transaction amount |
| TransactionUser | Bool | Boolean to check if the transaction is from either a Sender or Receiver |
| TransactionDescription | String | A variable used to store the description of the transaction |

Operations/Methods:

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Arguments Passed | Expected return value | Description |
| editAmount() | Float | Null | Function used to edit a transaction amount |
| editSender() | String | Null | Function used to edit Sender |
| editRecv() | String | Null | Function used to edit Receiver |
| editDescription | String | Null | Function used to edit transaction’s descriptions |

### iv. C4 Account Class

Attributes:

|  |  |  |
| --- | --- | --- |
| Attribute Name | Type | Description |
| UserName | String | Variable to store User’s Username |
| Password | String | Variable to store the User’s Password |
| Email | String | Variable to store the User’s Email |
| PhoneNumber | Int | Variable to store User’s Phone Number |

Operations/Methods:

|  |  |  |  |
| --- | --- | --- | --- |
| Method Name | Arguments Passed | Expected return value | Description |
| editUsername | String | Null | Function to edit Username |
| editPassword | String | Null | Function to edit Password |
| editEmail | String | Null | Function to edit Email |
| editPhoneNum | Int | Null | Function to edit Phone number |

## 

## C. Sequence Diagram



Figure 3. Sequence Diagram

This is the functionality of adding a transaction. This goes through the budget class because transactions attach to certain budget amounts. The budget class then creates the transaction and connects it, then adds it to the list of transactions on the Transactions page.

# 5.0 State Transition Diagram

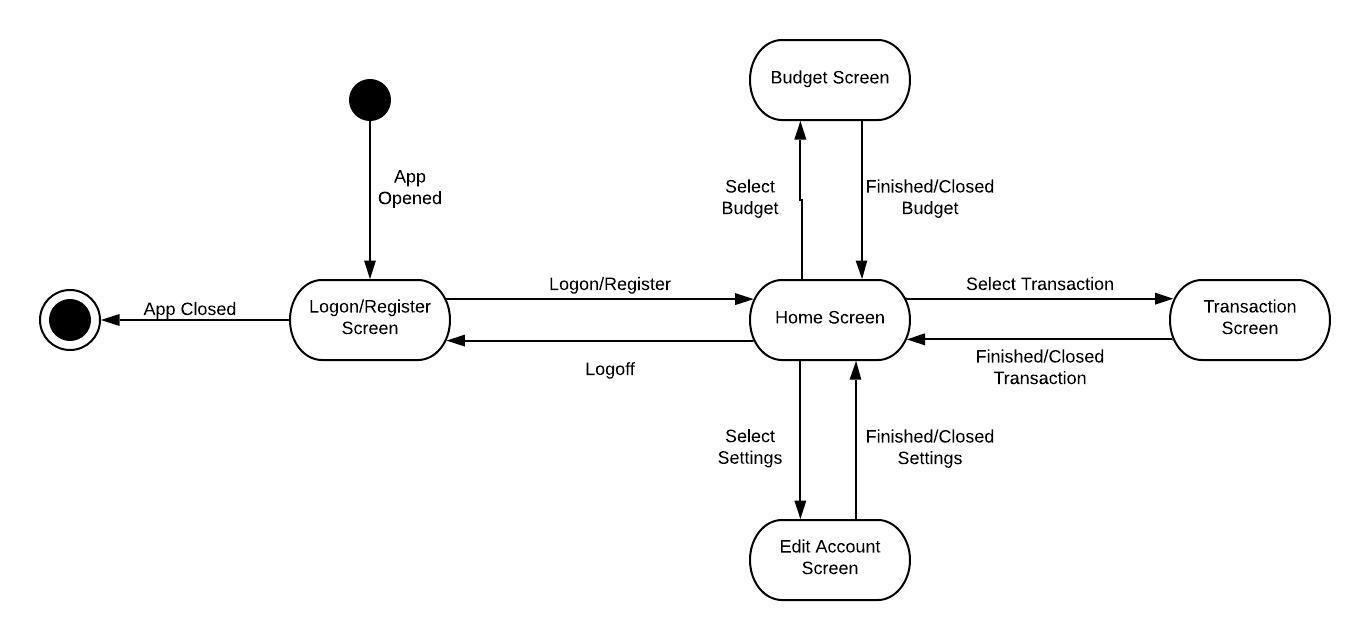


Figure 4. State Transition Diagram

The application that this group is designing is fairly simple and does not have a lot of transitions. The user will only really be able to enter transactions (profits/expenses), make personal goals (budgets), and edit their account information. Although there are only these three main interactions with the user, there is a lot of data that can be entered by the user in each of these selections. The list below will describe what each state is referring to, and how they transition from one to another:

* Login/Register Screen: This screen is as simple as it sounds, the user will login to the app, or create an account if one does not already exist. If the credentials are valid, or once they are done registering, then they will be redirected to the Home Screen. Since our application is a mobile app, the user may close out of the app at any time. They can also reach this login/register screen at any time when the app is open by simply selecting the “Logout” option.
* Home Screen: From the “Home Screen” the user will be able to access the remaining states listed below this bullet point (Budget Screen, Transaction Screen, & Edit Account Screen). At the Home Screen, the user will be able to see their budget, current balance, and past few options. The user will also be able to navigate to the Budget Sceen, Transaction Screen, and Edit Account Screen by selecting the hyperlink associated with each option. (ex. “Edit Budget”, “Enter Transaction”, & “Settings”).
* Budget Screen: At this state, the user can change, remove, or create a new budget goal. Each budget will have details such as: when the goal wants to be reached, how much money is wanting to be saved, etc. When the user is done changing, removing, or creating a budget goal, they will be redirected back to the Home Screen. The user will also be redirected back to the Home Screen if they select the cancel button.
* Transaction Screen: When the user selects the option to enter, remove, or change a transaction, they will be redirected to this screen. Here they will be able to enter information about the transaction such as: the sender/receiver, location, company, amount, details, and what goal it belongs to. Once the user is done adding, removing, or changing the transaction, they will be redirected back to the Home Screen. The user will also be redirected back to the Home Screen if they select the cancel button.
* Edit Account Settings: This state is also as simple as it sounds. The user will be able to edit their account information: update username, update password, update email, update notification settings, etc. The user will be redirected back to the Home Screen when they select the “Home” button on the screen.

# 6.0 User Interface Design

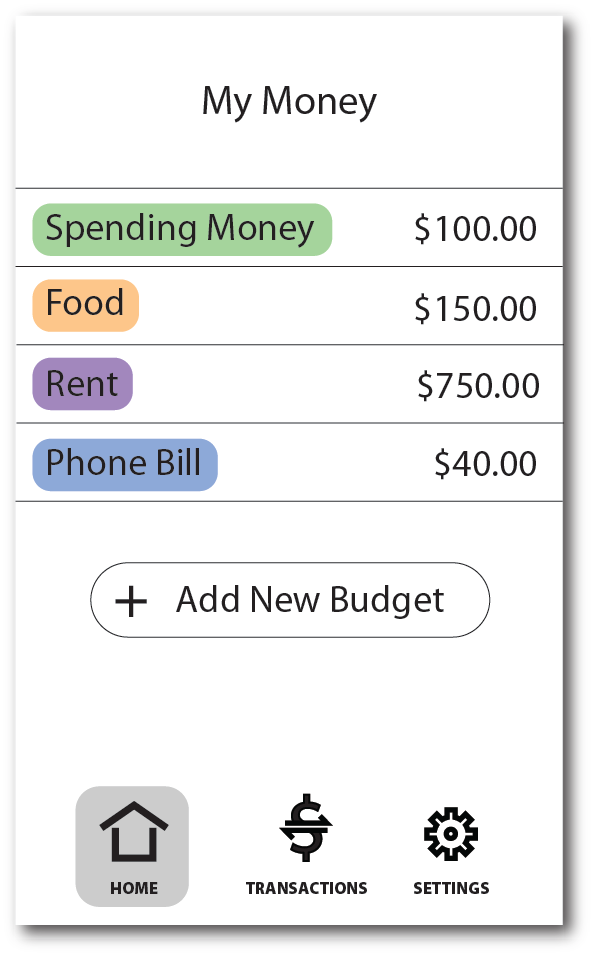
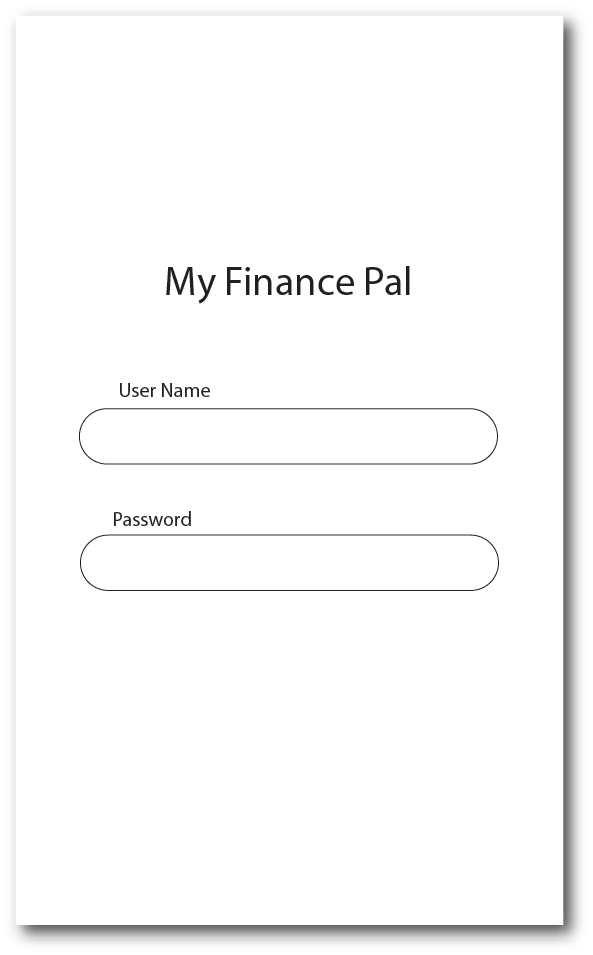


Figure UI1 Figure UI2

Figures UI1 and UI2 show the initial login process and the home screen. The login screen (Figure UI1) is a very basic interface that requires user information to get them into their account. This is needed to maintain their privacy. The home screen (Figure UI2) features the amount of money they have allotted to their budgets and the amount of money left over that was not attributed to a specific budget (Spending Money). This interface makes it simple for the user to see how much money they have available for different areas of spending. This main screen allows the user to create new budget items, and has a menu that allows the user to traverse the app and its other functionalities.

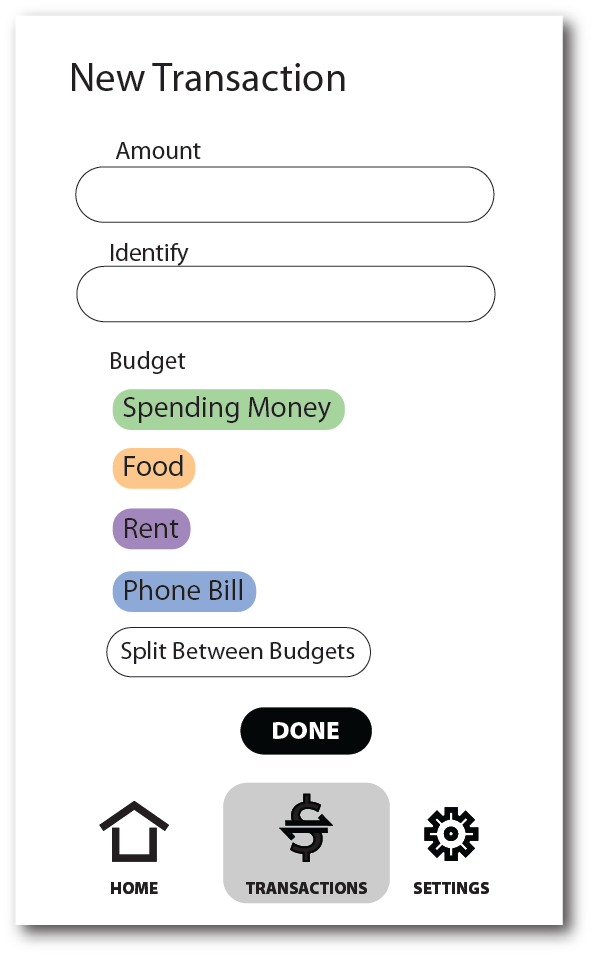
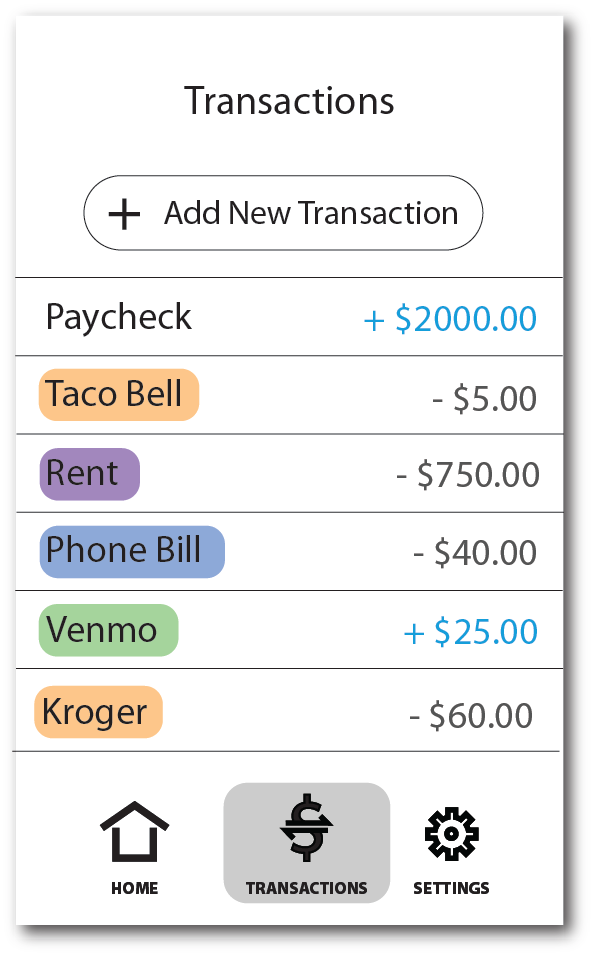


Figure UI3 Figure UI4

Figures UI3 and UI4 show the process of inputting new transactions in the Transactions menu. Pressing the Transactions button on the menu bar on the bottom of the screen will bring up the Transactions menu (Figure UI3). Here, the user can view their past transactions and the budgets that they have been applied to (e.g. the orange colored items are taken out of the Food budget). When the user presses the Add New Transaction button on the Transaction page, it will bring up an interface to input a new transaction (Figure UI4). Here, the user can input an amount of money, a name for the transaction, and a budget to apply it to. The user can specify between withdrawal or deposit here as well.

# 7.0 Conclusion

All the different designs and diagrams of our project we have created has definitely helped us in our progress of it. Not only has the creation of them taught us how to create them, but it has also taught us each unique flow for each of them. The designs and diagrams graphically represented a clear image of how we want our software application to look like and how we want it to behave, so we have a better understanding of how we want our app to be like. Our only issue when designing was our original idea of implementing different cards into the app, but we realized that was too complicated for the general theme of it so we decided to remove the idea.