

Final Report

for

MyWallet

Version <X.X>

Prepared by

Group Members:

Reem Al-Essa	441200983	441200983@student.ksu.edu.sa
Shroog Al-Harbi	439200674	439200674@student.ksu.edu.sa
Enas Al-Zahrani	441201082	441201082@student.ksu.edu.sa

Instructor: Farah Sheikh – Sarah Alotaibi

Course: CSC343- Systems Analysis & Design

Section: 69220

Date: 22 May 2022

Contents

CONTENTS	1
REVISIONS	3
1. PROJECT PROPOSAL	
1.1 INTRODUCTION	
1.2 STATEMENT OF PROBLEM	
1.3 PROJECT'S SCOPE	
1.4 PROJECT OBJECTIVES	
1.5 RELATED WORK	
2. GLOSSARY OF TERM	
3. SOFTWARE REQUIREMENTS SPECIFICATION	
3. 1 CUSTOMER STATEMENT OF REQUIREMENTS	
3.1.1 MONEY TRACKING	
3.1.2 AUTOMATIC DATA UPDATE	
3.1.3 DAMAGED RECEIPT	
3.1.4 DIFFICULTY SAVING MONEY	
3.1.5 DIFFICULTY PAYING BILLS	
3. 2 USER REQUIREMENTS	
3.2.1 ENUMERATED FUNCTIONAL REQUIREMENTS	
3.2.2 ENUMERATED NON- FUNCTIONAL REQUIREMENTS	
3.2.3 USER INTERFACE MOCK-UP	
3. 3 FUNCTIONAL REQUIREMENTS SPECIFICATION	
3.3.1 STAKEHOLDERS	
3.3.2 ACTORS AND GOALS	
3.3.3 USE CASE CASUAL DESCRIPTION	
3.3.4 USE CASE DIAGRAM	
3.3.5 USE CASE FULLY-DRESSED DESCRIPTION	
4. SOFTWARE DESIGN DOCUMENT	
4. 1 INTRACTION DIAGRAMS	
4.1.1 THE SYSTEM SEQUENCE DIAGRAM	
4.1.2 THE SEQUENCE DIAGRAMS	
4.1.2.1 UC2 <Sign in>	
4.1.2.2 UC7 <New budget>	
4.1.2.3 UC18 <Classify payment>	
4.1.2.4 UC19 <View account details>	
4.1.2.5 UC22 <MyWallet Transfer>	
4. 2 THE SYSTEM STRUCTURAL DIAGRAM	
4.2.1 THE DETAILED CLASS DIAGRAM "SYSTEM ARTIFACT	
4.2.2 THE ATTRIBUTES AND METHODS DESCRIPTIONS FOR EACH	
CLASS	
4.2.3 THE OBJECT DIAGRAM	

4. 3 THE SYSTEM ARCHITECTURE AND DESIGN	
4.3.1 THE SYSTEM ARCHITECTURAL DESIGN	
4.3.1.1 THE SYSTEM ORGANIZATION	
4.3.1.2 THE DATA FLOW MODEL	
4.3.1.3 THE CONTROL MODEL	
4.3.2 IDENTIFYING THE SUBSYSTEMS	
4. 4 THE BEHAVIORAL DIAGRAMS	
4.4.1 ACCOUNT STATE DIAGRAM	
4.4.2 BUDGET STATE DIAGRAM	
4. 5 THE USER INTERFACE DESIGN	
4. 6 THE DESIGN OF TESTS	
4.6.1 UNIT TESTING	
4.6.2 INTEGRATION TESTING	
4.6.3 ACCEPTANCE TESTING	
4.6.4 PATH TESTING	
4.6.4.1 PATH CYCLOMATIC COMPLEXITY	
5. INDIVIDUALS CONTRIBUTIONS BREAKDOWN	
6. REFERENCES	

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00

1.1 Introduction

We live in a consumerism culture, where people want to buy things, they don't need, which can lead to them losing money easily by focusing too much on buying useless things instead of things they need. That's where the concept of organizing finances comes, which is when someone keeps track of where and when his/her money goes.

In recent years, people have realized the benefits of organizing their spending and its importance, like how it can help you in saving money for your plans and helps you to focus on what's important.

The lack of efficient spending management can help to drive people into debt and can harm their finances as much as not having enough money, and that's why people need to consider organizing their spending more seriously.

1.2 Statement of Problem

Many people want to start organizing their finances, but some reasons may prevent them from doing so, a person may want to start organizing his/her spending but not have the time to do so, or don't even know where to start.

Nowadays it becomes even harder for someone to organize his/her finances due to the many paying options and the often loose of bills. Therefore, we thought about a solution for the mentioned problems, we call it 'MyWallet'. MyWallet is a phone application that is designed to help users in organizing their finances and income. It provides the user with the ability to categorize his/her spending to help him/her to know how where his/her money goes.

We work on providing help to those who don't have an idea on where to start saving by 'MyWallet' application that will contain the features they need.

1.3 Project's Scope

MyWallet focuses on people who want to manage their economy on a personal level or a larger economy. MyWallet will be available on Android and IOS.

1.4 Project's Objectives

MyWallet helps anyone who wants to manage his/her money or/and wants to save money or/and wants to keep his/her spending in check.

- Adding money:
The customer can add money to his/her MyWallet account with different options.

Adding money options:

- Apple Pay
- STC pay
- credit\debit card
- bank transfer
- SADAD

▪ Transferring money:

○ to MyWallet accounts:

The customer can transfer money from his/her MyWallet account to another MyWallet account using the account number or by using a QR code.

○ Local Transfer

The customer can Transfer locally to any bank in the KSA from his/her MyWallet account with simple steps.

▪ Budget:

MyWallet makes spending money more efficient, by simply categorizing the customer's money into Budget sections (grocery, electricity, gas, ext.).

▪ Bills:

MyWallet enables the customers to control their finances, including paying their bills by using the billing account or bill code with the amount.

▪ Pay:

Using MyWallet The customer will pay easily using the QR code. The customer can add the Receipt by taking a photo of the Receipt or scan the Receipt's QR code.

▪ Receipts:

MyWallet saves the customer's Receipts by easily taking a Pictor of the Receipt or scanning the QR code and the customer also can link the Receipt with a Transaction.

▪ Saving Plans:

MyWallet makes achieving goals more convenient by easily enable the Customers to make saving plans and keeping track of their Saving plans.

1.5 Related Works

There are a lot of mobile applications that are about managing, tracking, and saving money, but there are two of them which are like our app:

1-STC pay:

STC pay gives you complete control over your finances, such as issuing cards, transferring money, paying bills, gaming & entertainment.

This application allows you to add money to your STC account and a card with different options, also enables you to control your finances, including paying your STC bills and recharging Sawa prepaid lines with total smoothness [1].

2-SpendNotes:

SpendNotes help you manage your spending with the least amount of effort, and it has lots of features like tracking your expenses & income, organizing your transactions into separate accounts according to their source or nature, defining budgets to control expenses over a day, a week or a month, year or over a specific date range, and define plans to track recurring expenses & incomes or to be paid at a specific time in future[2].

What distinguishes MyWallet from these applications is that we have more features in our application like being able to add money with more options for example, adding the money by using Apple pay or STC pay or SADAD and more options, also the user of our application can add the Receipt by taking a photo of the Receipt or just by scan the Receipt's QR code, our application help the users to save their money by making saving plans.

2. Glossary of Terms

TERM	DESCRIPTION
Savings	an amount of money that you do not need to spend, and you keep it in an account in a bank or similar financial organization.
Budget	a plan to show how much money a person or organization will earn and how much they will need or be able to spend.
Debt	something, especially money, that is owed to someone else, or the state of owing something.
QR code	a pattern of black-and-white squares that is printed on something and that can be read by some types of mobile phone to give information to the user of the phone.
Data	information, especially facts or numbers, collected to be examined and considered and used to help decision-making, or information in an electronic form that can be stored and used by a computer.
Finance	the management of money, or the money belonging to a person, group, or organization.
Bill	a request for payment of money owed, or the piece of paper on which it is written.

3.1 Customer Statement of Requirements

As life progresses, living requirements and expenses increase, putting a lot of physical and psychological strain on us. This makes us want an appropriate plan to organize our spending and save our money so that we can buy what we want and do what we want without a worry, and this does not happen without closely monitoring our money, where we spent it, how we spent it, and what we plan to spend in the future.

So, we decided to build an application capable of assisting in the arrangement of client finances to find a solution to this problem and in order to assist the user in achieving financial goals and making him able to track his money with ease through our application, and to ensure that our application will be able to meet the needs of the client, we decided to hold meetings via Zoom with a random volunteer group of people living in Saudi Arabia, representing various age groups, in order to learn about the difficulties they face in organizing and tracking money, and we will seek to solve these problems through our application .

3.1.1. Money Tracking

One of the difficulties that clients face is not remembering where their money went at the end of the month because they spent money in a random and unregulated way, which makes them more vulnerable to problems and financial pressures, which also causes them to have to postpone their plans or even borrow money to pay off their debts.

The solution to these problems is to use our application, which has the benefit of tracking money and allowing the customer to see how much money was spent daily, allowing the customer to be more informed about the finances.

3.1.2. Automatic Data Update

Many people find that updating data on their financial consumption is difficult, and they do not have the time to do it so, as a result, many people stop paying attention to tracking their money and debts, and they return to their previous habits of not tracking their money consumption.

As a result, we have given our application with an automatic update on their financial consumption and data, saving the customer a lot of time and work.

3.1.3. Damaged Receipt

We all know how easy it is to misplace or damage Receipt, which causes many problems for customers who want to keep them to take advantage of them.

But, because of our application, it's no longer so difficult. Our application gives customers a variety of options for adding Receipt and keeping them in the app, such as taking pictures of them or scanning the Receipt's QR code and linking each Receipt with its Transaction.

3.1.4. Difficulty Saving Money

Saving money is necessary to make life easier. it allows people to buy and do whatever they want without having to worry about their budget or going into debt, but saving money is not easy because it requires a clear goal and consistent plan for prioritizing their spending.

To make the savings process simple and clear, we created an application that can organize appropriate saving plans that save the customer time and effort while also renewing each period to meet the user's goals and conditions.

3.1.5. Difficulty Paying Bills

Dealing with bills can be stressful, especially if you're behind on payments, but sticking to good billing habits will help you not only pay on time, but also continue to pay without delay.

To help our customers in paying their bills on time and in the easiest ways possible, our app allows them to manage their finances, including paying bills by using the billing account or bill code with the amount

3.2 User Requirements

3.2.1 Enumerated Functional Requirements

REQ- ID	DESCRIPTION	PW
REQ - 1	The user shall be able to sign in	5
REQ - 2	The user shall be able to Log in	5
REQ - 3	The user shall be able to Log out	5
REQ - 4	The user shall be able to add money to MyWallet account using VISA	5
REQ - 5	The system shall provide the user with a unique 10 number as there MyWallet account number	5
REQ - 6	The system shall provide the user with a QR code for their MyWallet account information to share (name – account number).	1
REQ - 7	The user shall be able to transfer money to anther MyWallet accounts.	3
REQ - 8	The user shall be able to transfer money to a KSA Bank account	1
REQ - 9	<p>The user shall be able to create a budget section</p> <p>9.1 The user shall set a monthly amount for the budget</p> <p>9.2The user shall be able to add a title for every Budget.</p> <p>9.3 The user shall be able to add notes to the sections</p>	5
REQ - 10	<p>The user shall be able to create a Saving plan</p> <p>10.1 The user shall enter the wanted amount of money to be saved</p> <p>10.2 The user shall set number of months</p> <p>10.3 The system shall be able to calculate the monthly payment for the saving plan.</p> <p>10.4 The system shall alert the user every month.</p> <p>10.5 system shall calculate the unpaid money and red label it.</p>	5
REQ - 11	<p>The user shall be able to Pay using MyWallet application.</p> <p>11.1 The system shall view the payment at the Transactions List</p> <p>11.2 The system shall be able to save the date and the time of the payment.</p>	3
REQ - 12	<p>The user shall be able to Pay bills using</p> <p>12.1 The user shall be able to add a title for the bill</p> <p>12.2 The user shall be able to add notes for the bill</p> <p>12.3 The system shall save the bill</p> <p>12.4 The system shall keep the bill updated</p> <p>12.5 the user shall be able to view the previse add bills</p>	2
REQ - 13	<p>The user shall be able to add his/her Receipts</p> <p>13.1 link the Receipts with one of the account Transactions</p> <p>13.2 View previse add Receipts</p>	4

REQ - 14	The user shall be able to view home page	3
REQ - 15	The user shall be able to view budget details 15.1 the user shall be able to view the Transactions of the budget from the budget details.	5
REQ - 16	The user shall be able to view the welcome page.	1
REQ - 17	The user shall be able to view the account profile.	4
REQ - 18	The user shall be able to update profile information.	4
REQ - 19	The user shall be able to view saving plans list.	5
REQ - 20	The user shall be able to pay by scanning QR	2
REQ - 21	The user shall be able to view the Transaction details. 21.1 the user shall classify the budget Transaction. 21.2 the user shall be able to add notes. 21.3 the user shall be able to add the Receipt.	3
REQ - 22	The user shall be able to view account details 22.1 the user shall be able to view the Transactions from the account details page.	4
REQ - 23	The user shall be able to view financial analysis	1

3.2.2 Enumerated Non-Functional Requirements

REQ- ID	DESCRIPTION	PW
REQ – 25	The system shall secure, the user`s shall be the only one who is allowed to view and use the account.	5
REQ – 26	The System defect rate shall be less than 1 failure per 1000 hours of operation.	3
REQ - 27	New user shall be able to perform any function on the system within less than 5 minutes.	3
REQ - 28	The system shall protect the financial information of the user	5
REQ - 29	The system shall be able to perform all the operations in less than 1 minute.	4
REQ - 30	The system shall be able to handle 50.000 accounts.	3
REQ – 31	The system shall be able to recognize the KSA bank accounts.	2

3.2.3 User Interface Mock-up







SAVE

New Bill

billing account number

00.00 SAR

notes

Pay



Transfers

MyWallet Account

Local Bank



My Wallet Transform


00.00 SAR

account number

full name

note

conform



Local Transform

00.00 SAR

IBAN

SA-

full name

note

conform



123456789012345

3,844.50 SAR

Last Transactions

P-701-11-21165.00 SAR

P-630-10-2115.00 SAR

P-530-10-21187.00 SAR

P-429-10-2198.50 SAR

P-328-10-21165.00 SAR



SAVE



WIFI


billing account number

165.00 SAR

notes

Pay





Scan QR





shroog alharbi
123456789012345

3.3 Functional Requirements Specification

3.3.1 Stakeholders

- **Users:**

Ordinary people who are struggling with organizing their finances. They are interested in this application because they can categorize their spending, pay, and make saving plans. Furthermore, they can pay their bills and save them.

- **Back-End Developers:**

Professional programmers who are responsible for creating, coding, improving the application and databases before publishing the application.

- **Front-End Developers:**

The people who are responsible for ensuring that website visitors can easily interact with the page.

- **Administrator:**

The people who are responsible for the system management, such as, security, database, network and responding to any problem that accrue.

- **Banks:**

Banks are the companies who are responsible for providing the customer with the ability to add money.

- **Advertisers:**

Business owners who are interested in promoting their services through the application.

- **Customer service:**

The employees who are responsible for receiving and answering the emails sent by the users and solving the problems they face.

3.3.2 Actors and Goals

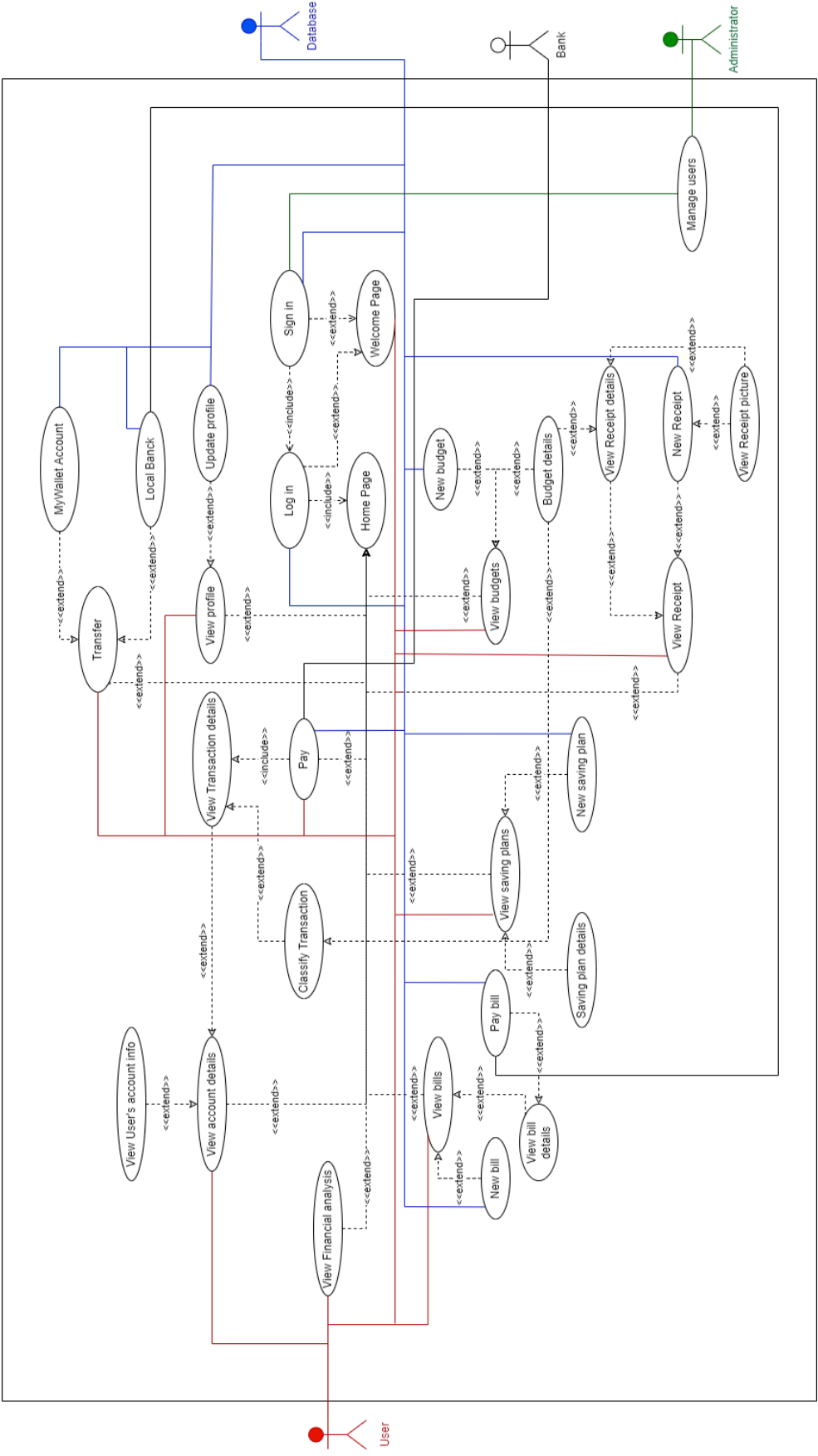
Actor	Type	Goal
User	Initiating	People who use this application to organize his/her finances
Administrator	Participating	People who manage the application's security, database, network and responding to any problem
Database	Participating	It stores application's information
Banks	Participating	It provides the user with the ability to add money

3.3.3 Use Case Casual Description

Use Case ID	Name	Short Description	Corresponding REQ-id
<i>UC1</i>	<i>Welcome page</i>	The user chooses whether he/she wants to sign in or log in.	<i>REQ - 16</i>
<i>UC2</i>	<i>Sign in</i>	The user registers an account by entering his/her information.	<i>REQ - 1</i>
<i>UC3</i>	<i>Log in</i>	The user logs in by entering his/her email and password.	<i>REQ - 2</i>
<i>UC4</i>	<i>Home page</i>	The user views his/her home page.	<i>REQ - 14</i>
<i>UC5</i>	<i>View profile</i>	The user views his/her profile information which allows him/her to update his/her profile.	<i>REQ - 17</i>
<i>UC6</i>	<i>Update profile</i>	The user updates his/her profile.	<i>REQ - 18</i>
<i>UC7</i>	<i>New budget</i>	The user adds new budget which allows him/her to categories his/her spending.	<i>REQ - 9</i>
<i>UC8</i>	<i>New Receipt</i>	The user adds new Receipt and link it with a Transaction if wanted.	<i>REQ - 13</i>
<i>UC9</i>	<i>New saving plan</i>	The user adds new saving plans for future projects.	<i>REQ - 10</i>
<i>UC10</i>	<i>Budget details</i>	The user views the budget details and information.	<i>REQ - 15</i>

<i>UC11</i>	<i>Saving plan details</i>	The user views the saving plans details list.	<i>REQ - 19</i>
<i>UC12</i>	<i>View Receipt picture</i>	The user views the Receipt picture.	<i>REQ - 13</i>
<i>UC13</i>	<i>View Receipt</i>	The user views the Receipt that been already added.	<i>REQ - 12</i>
<i>UC14</i>	<i>View budgets</i>	The user views the budgets that been already added.	<i>REQ - 15</i>
<i>UC15</i>	<i>View saving plans</i>	The user views the saving plans that been already added.	<i>REQ - 19</i>
<i>UC16</i>	<i>Pay</i>	The user makes a payment (purchase) using a QR.	<i>REQ - 20</i>
<i>UC17</i>	<i>View Transaction details</i>	The user views the Transaction details and information.	<i>REQ - 21</i>
<i>UC18</i>	<i>Classify Transaction</i>	The user is asked to classify the Transaction belong to which budget.	<i>REQ - 21</i>
<i>UC19</i>	<i>View account details</i>	The user views the details, resent activities and the account balance.	<i>REQ - 2</i>
<i>UC20</i>	<i>View Transfer Options</i>	The user views transaction options (Local/MyWallet).	<i>REQ - 26</i>
<i>UC21</i>	<i>Local Transfer</i>	The user transfers money to a local bank account.	<i>REQ - 8</i>
<i>UC22</i>	<i>MyWallet Transfer</i>	The user transfers money to another MyWallet account.	<i>REQ - 7</i>
<i>UC23</i>	<i>View Bills</i>	The user views previous added bills.	<i>REQ - 12</i>
<i>UC24</i>	<i>new Bill</i>	The user adds a new billing account to view or pay.	<i>REQ - 23</i>
<i>UC25</i>	<i>Pay Bill</i>	The user pays the bill using the bill's account number.	<i>REQ - 12</i>
<i>UC26</i>	<i>View Financial Analysis</i>	The user View the <i>Financial Analysis</i>	<i>RQ-23</i>
<i>UC27</i>	<i>View Account Info</i>	The user views the account number – user's name – QR.	<i>RQ - 6</i>

3.3.4 Use Case Diagram



3.3.5 Use Case Fully Dressed Description

UC2 <Sign in>

Initiating Actor: User

Actor's Goal: Sign into the system.

Participating Actor: Database

Pre-Conditions: User clicked on Sign in button from the welcome page.

Post-Condition: User sign in successfully and the user's home page will appear.

Flow of Events for Success Scenario:

1. User enters all the information.
2. User clicks on the sign in button.
3. system validates the entered email, adds the User to the Database then signs the User to the system.
4. View Login page.

Flow of Events for Extension (Alternate Scenario):

1. User clicks on Sign in button from the welcome page
2. User enters a registered email, an error message "This email is already registered" will appear.

UC7 <New budget>

Initiating Actor: User

Actor's Goal: Adding a new Budget Section.

Participating Actor: Database

Pre-Conditions: User have logged in; User clicked on Budget in home page and then clicked the plus '+' button in the View budgets page.

Post-Condition: The User added a new budget section successfully.

Flow of Events for Success Scenario:

1. User enters needed information.
2. User clicks on Save button.
3. system validates the entered Title, then adds the budget to the Database.

Flow of Events for Extension (Alternate Scenario):

1. User enters needed information.
2. User clicks on Save button.
3. system Invalidates the entered Title, an error message "Budget Exist " will appear.

UC18 <Classify payment>

Initiating Actor: User

Actor's Goal: Classify the Transaction to the wanted budget section.

Participating Actor: Database

Pre-Conditions: User has logged and clicked on view Account details in home page and made at least one Transaction and clicked on a Transaction to view the Transaction details and have at least one Budget section.

Post-Condition: The User has classified the Transaction and the Transaction has been added to the selected Budget section.

Flow of Events for Success Scenario:

1. User clicks on Classify Transaction at the Transaction detail page.
2. User select a Budget section.
3. system validates the Transaction classification, then adds it to the chosen Budget section in the Database.

Flow of Events for Extension (Alternate Scenario):

1. User clicks on Classify Transaction at the Transaction detail page.
2. No Budget section to select an error message "No Budget Sections" will appear.

UC19 <View account details>

Initiating Actor: User

Actor's Goal: View the Account details.

Participating Actor: Database

Pre-Conditions: User have logged in; User clicked on the 'Account Details' button in the home page.

Post-Condition: The User have viewed the account details successfully.

Flow of Events for Success Scenario:

1. User press the 'Account Details' button.
2. system opens the User 's account details page.

Flow of Events for Extension (Alternate Scenario):

None.

UC22 <MyWallet Transfer>

Initiating Actor: User

Actor's Goal: Transfer money to another existing MyWallet account.

Participating Actor: Database

Pre-Conditions: User have logged in, the user clicked on Transfer button in the home page and click on and then selected "MyWallet Account" button.

Post-Condition: Money has been Transferred to another MyWallet account successfully.

Flow of Events for Success Scenario:

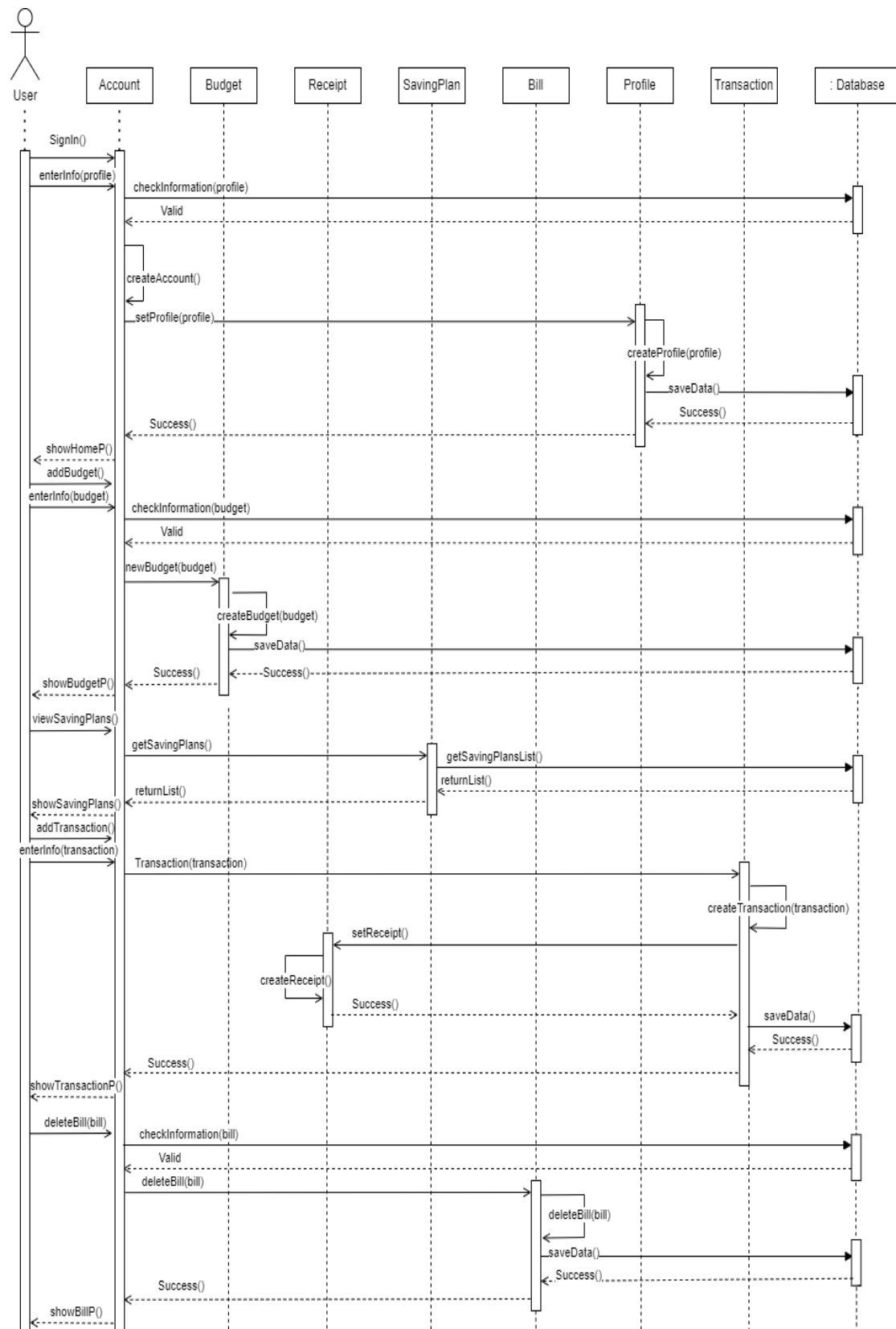
1. User enters needed information.
2. User clicks on "Conform" button.
3. system validates the transaction, then transfer the money to the chosen account.

Flow of Events for Extension (Alternate Scenario):

1. User enters needed information.
2. User clicks on "Conform" button.
3. User 's Balance is less than the entered amount, an error message " Low Balance" will appear.

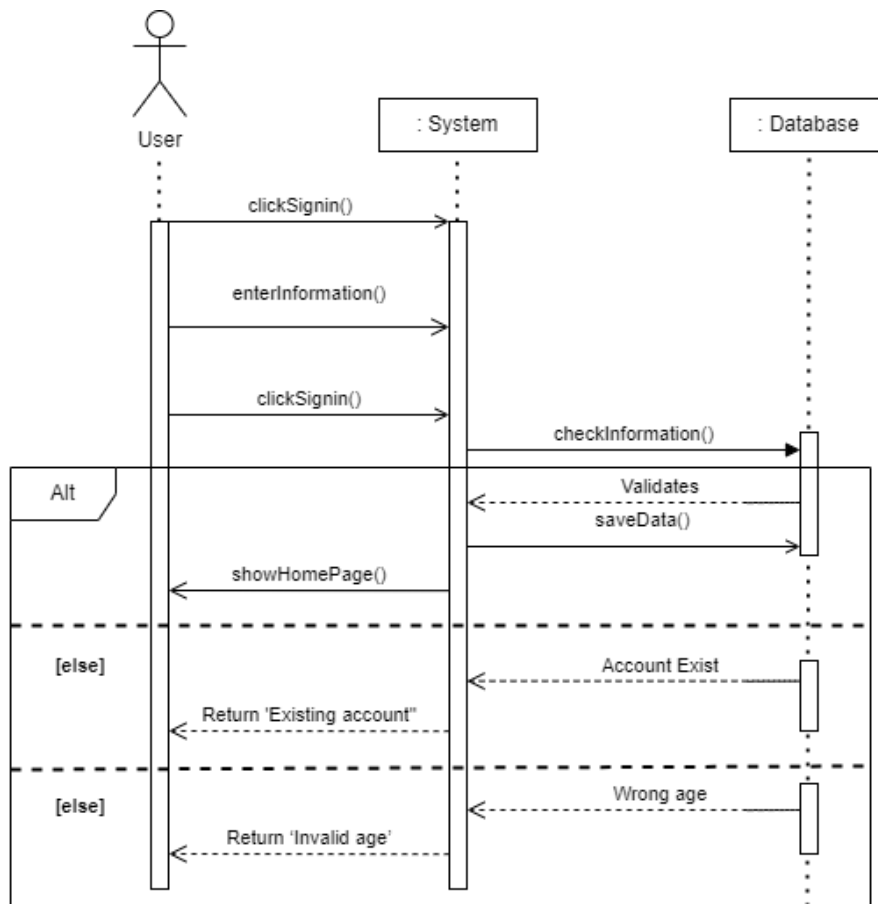
4.1 Interaction Diagrams

4.1.1 System Sequence Diagrams

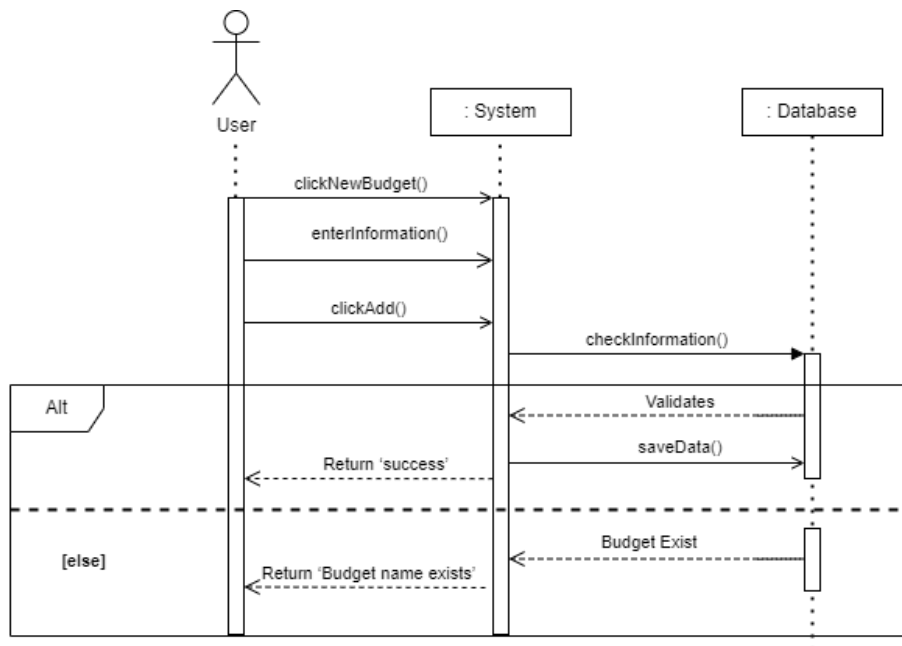


4.1.2 Sequence Diagrams

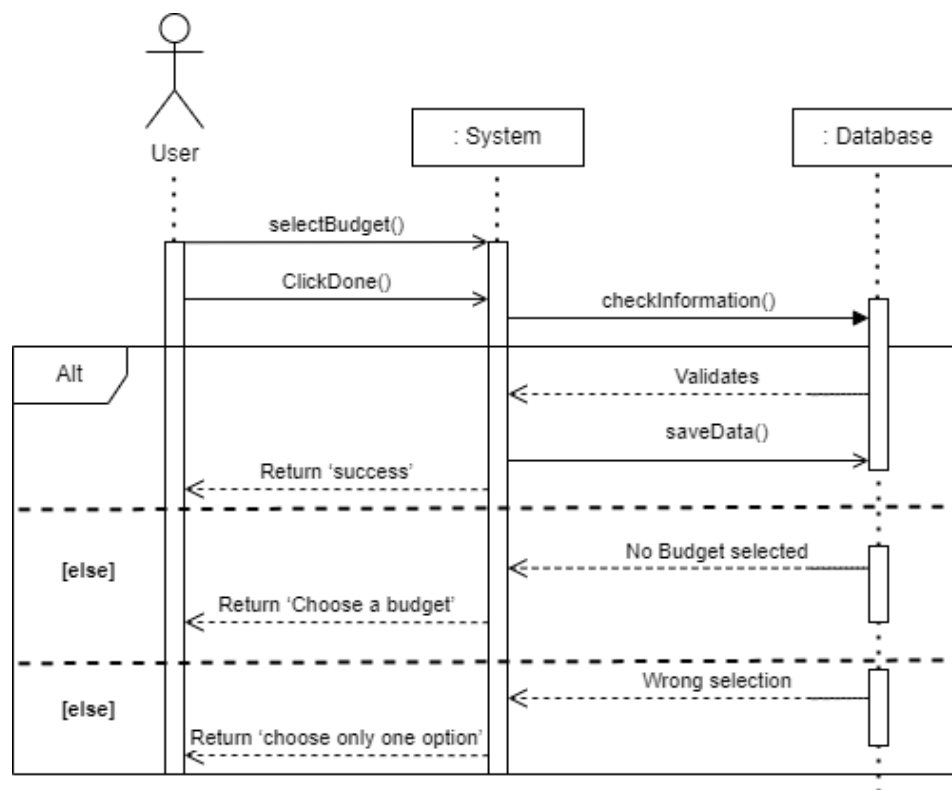
4.1.2.1 UC2 <Sign in>



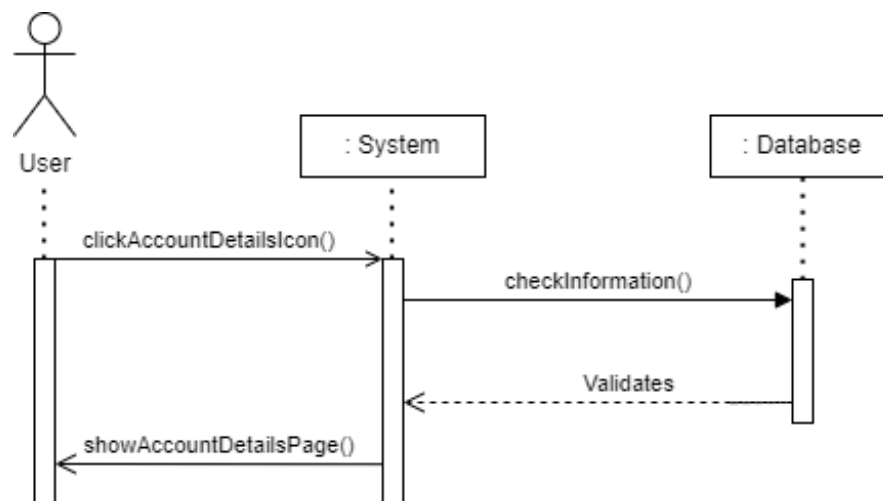
4.1.2.2 UC7 <New budget>



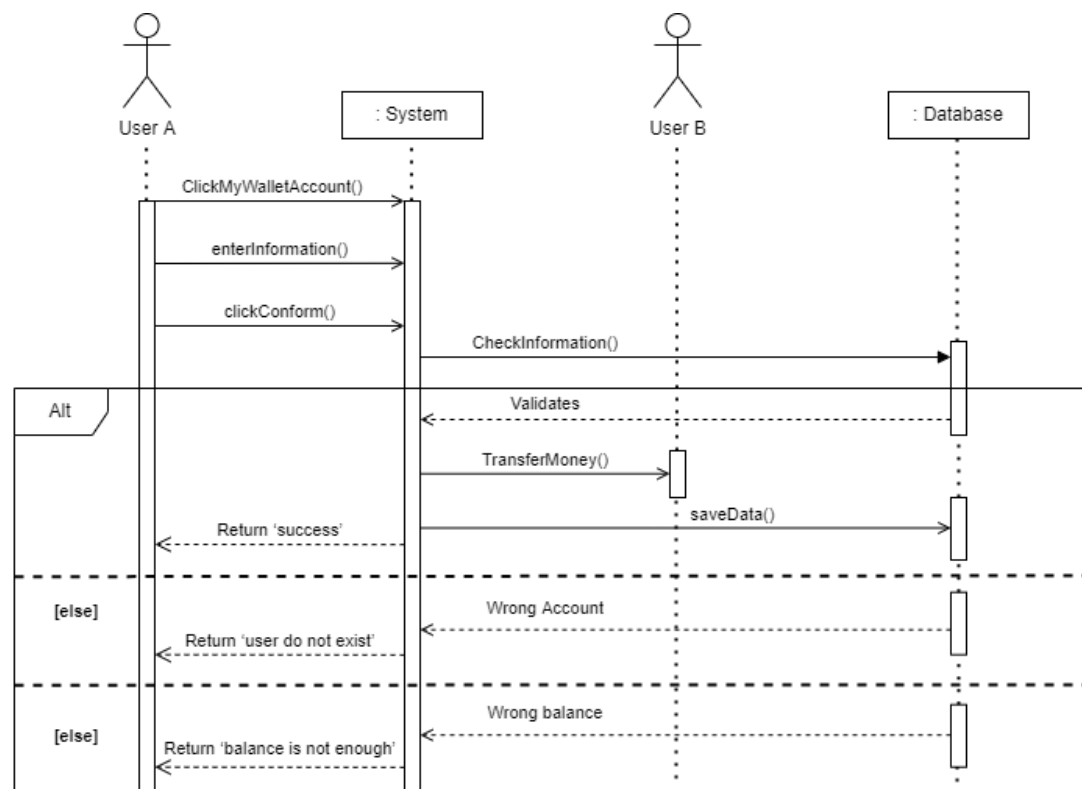
4.1.2.3 UC18 <Classify payment>



4.1.2.4 UC19 <View account details>



4.1.2.5 UC22 <MyWallet Transfer>



4.2 The System Structural Diagram

4.2.1 The Detailed Class Diagram “Software Artifact”



4.2.2 The Attributes and Methods description for each Class

Class Account

Attributes:

accountNum: a String represents the users account number
balance: a double represents the user total balance
QR: a String represents an URL for the user QR which contains the users account number
profile: a Profile object which contains the user profile information
myWalletAccounts: Linked List of type String contains MyWallet account numbers user adds them to transfer money.
localAccounts: Linked List of type String contains Local Banks account numbers user adds them to transfer money.
Budgets: Linked List contains Budget objects represents the user budget sections.
transaction: Linked list contains the transaction done be the account.
savingPlans: Linked List contains SavingPlan objects represents the user saving plans.
bills: Linked List contains Bill objects represents the user bills.
receipts: Linked List contains Receipts objects represents the user receipts.

Methods:

addMonye(): enable the user to add money to the account.
myWalletTransfer: enable the user to transfer money to another MyWallet account.
localTransfer: enable the user to transfer money to a local bank account.
newSavingPlan: enable the user to make a new saving plan.
deleteSavingPlan: enable the user to delete a saving plan.
viewSavingPlanList: view the user's saving plans.
viewSavingPlanInfo: enable the user to view a saving plan's info.
newBudget: enable the user to make a new budget.
deleteBudget: enable the user to delete a budget.
viewBudgetList: view the user's budget sections.
viewBudgetInfo: enable the user to view a budget info.
viewProfileInfo: view the user's profile.
updateProfileInfo: enable the user to edit the profile info.
newBill: enable the user to make a new bill.
deleteBill: enable the user to delete a bill.
viewBillList: view the user's bills.
viewBillInfo: enable the user to view a bill info.
newReceipt: enable the user to make a new receipt.
deleteReceipt: enable the user to delete a receipt.
viewReceiptList: view the user's receipts.
viewReceiptInfo: enable the user to view a receipt info

Class Profile

Attributes:

firstN: a String represents the user's first name
lastN: a String represents the user's last name
age: a int represents the user's age.
phoneNum: a String represents the user's phone number.
emailAddress: a String represents the user's email address.
Methods:

updatePhoneNum: enable the user to edit the phone number.
updateEmailAdress: enable the user to edit the email adress.

Class Bill

Attributes:

amount: a double represents the amount of the bill.
billingAccount: a String represents the bill account number.
title: a String represents the bill title.
note: a String represents the note that the user adds.

Methods:

updatePhoneTitle: enable the user to edit the title.
updateNote: enable the user to edit the note.
pay: enable the user to pay the bill.

Class Receipt

Attributes:

title: a String represents the receipt title.
note: a String represents the note that the user adds.
transaction: a Transaction object represents the receipt payment if the payment done using MyWallet and null otherwise.

Methods:

updateTitle: enable the user to edit the title.
updateNote: enable the user to edit the note.
updateTransaction: enable the user to link the receipt with a transaction.

Class Transaction

Attributes:

amount: int represents the transaction amount.
accountNum: a String represents the receiving or the sending account number.
date: represents the date of the transaction.
receipt: represents the receipt of the transaction or null.
sections: represents the budget sections for the transaction.
note: a String represents the note that the user adds.

Methods:

updateNote: enable the user to edit the note.
setBudget: enable the user to set a budget to the transaction.
setReceipt: enable the user to set a budget to the receipt

Class SavingPlan

Attributes:

total: double represents the total amount the user wants to save.
duration: int represents the number of months that the user wants to save the total in.
monthlyAmount: double represents the monthly amount taken from the account and it's the total divided by the duration.
remaining: double represents the remaining money to save.
title: a String represents the receipt title.
note: a String represents the note that the user adds.

Methods:

updateTitle: enable the user to edit the title.
updateNote: enable the user to edit the note.
updateDuration: enable the user to edit the saving plan duration.

Class Budget

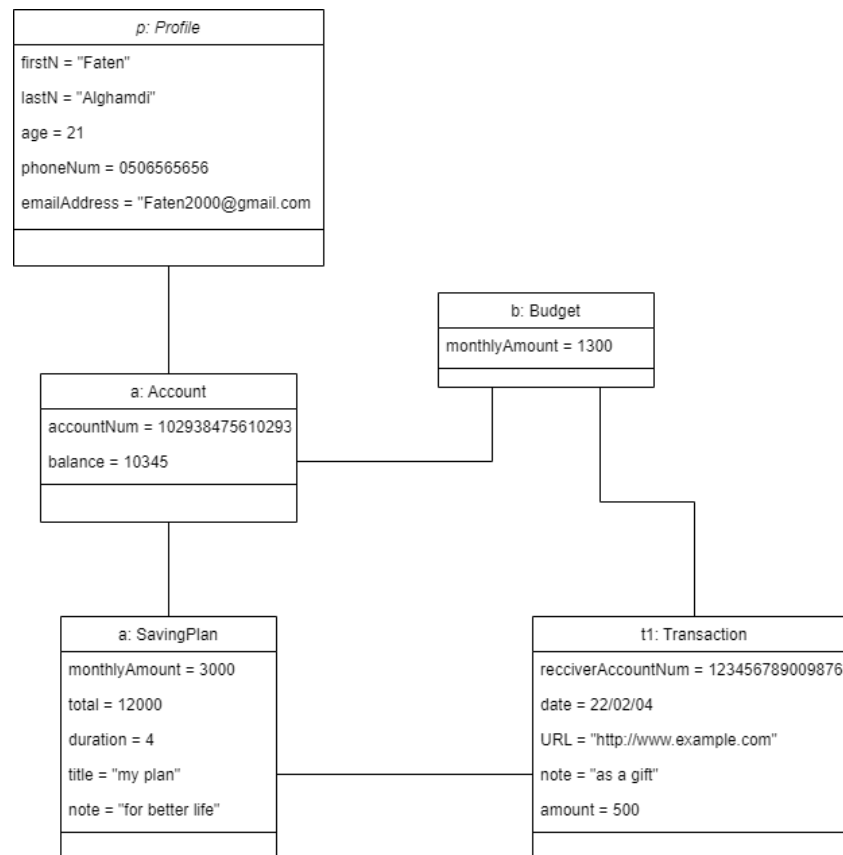
Attributes:

monthlyAmount: double represents the monthly amount for the budget.
total: represents the total money the budget has.
title: a String represents the receipt title.
note: a String represents the note that the user adds.

Methods:

updateTitle: enable the user to edit the title.
updateNote: enable the user to edit the note.

4.2.3 The Object Diagram



4.3 The System Architecture and design

4.3.1 The System Architectural Style

4.3.1.1 The System Organization

Client-server architecture

We chose the client/server architecture because the system requires constant access to stored data and persistent data storage to maintain the documents, which the client/server architecture can provide. On the server side, there is a database subsystem that holds the data for our application system. This server can handle client requests and send or receive information as needed. The server can also support multiple clients at the same time while controlling access to shared services and data. In addition, because the server uses software, it does not require a dedicated server computer or complex communication protocol, which makes our product cheaper and easier to implement.

4.3.1.2 The Data Flow Model

Object-oriented model

We chose the object-oriented model because our system is strongly object-oriented and has a well-defined interface. Object-oriented models enable object identification and communication while supporting data abstraction, inheritance, and encapsulation. It also creates application and database development union and transforms them into a unified data model and language environment.

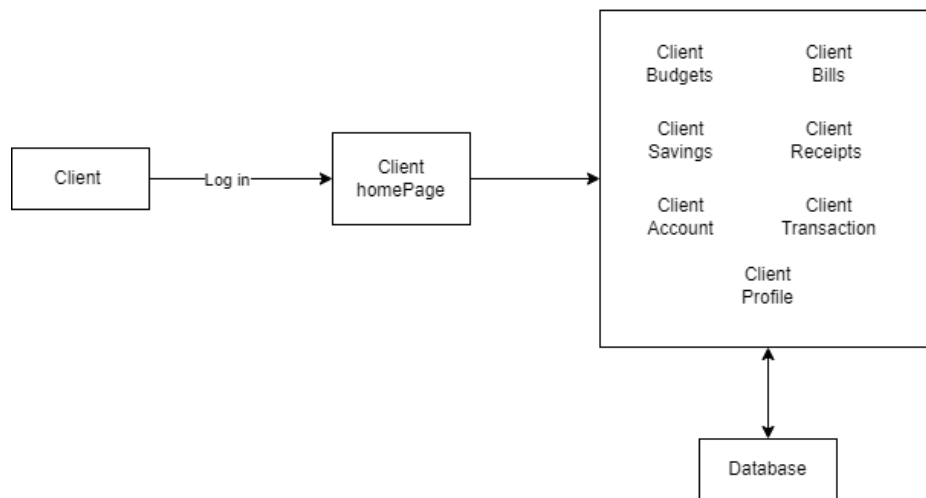
4.3.1.3 The Control Model

Centralized control model

We selected centralized control model since in this model it is relatively simple to analyze control flows and work out how the system will respond to inputs.

Centralized control model implies that one component is designated as the controller and is responsible for managing the execution of other components.

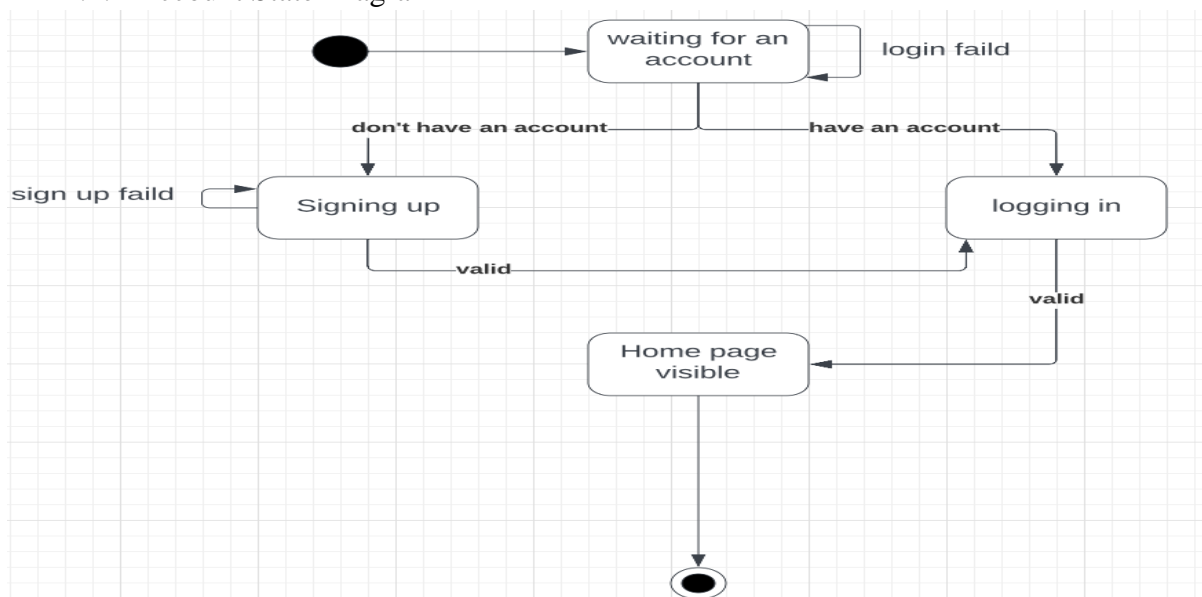
4.3.2 Identifying the Subsystems



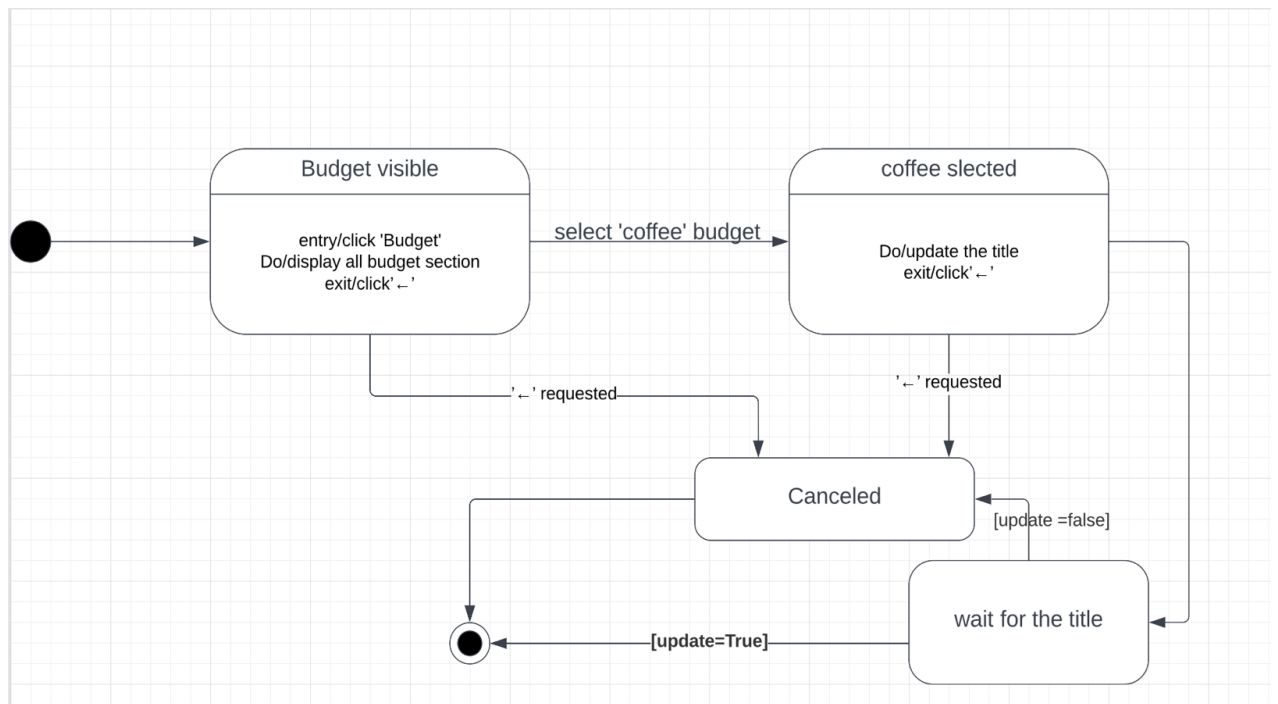
4.4 The System Behavioral Diagrams

The State Diagram

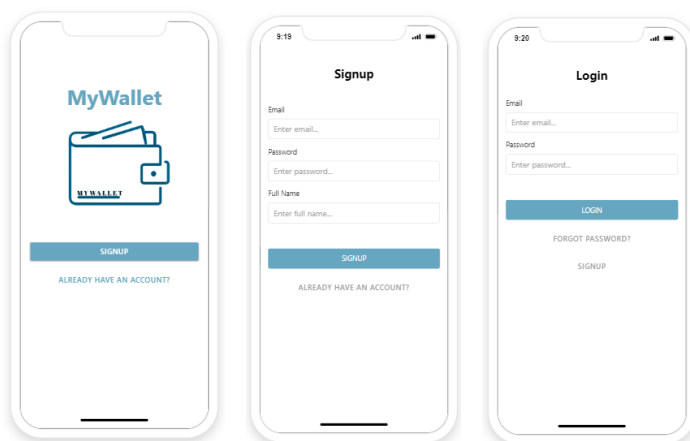
4.4.1 Account State Diagram

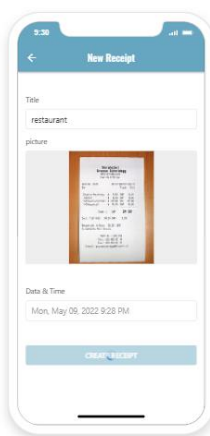
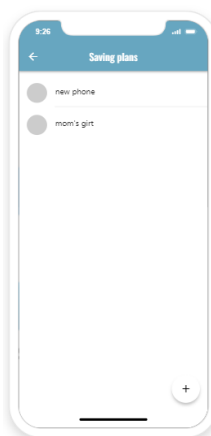
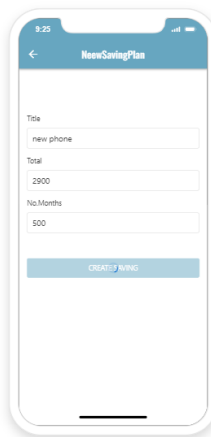
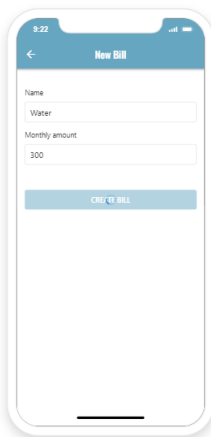
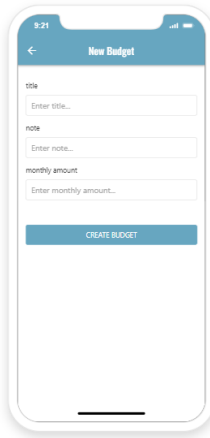


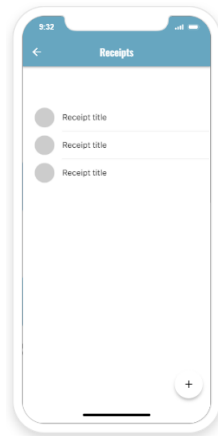
4.4.2 Budget State Diagram



4.5 The User Interface Design







4.6 The Design of Tests

4.6.1 Unit Testing

Test Case ID	Use case tested	Description	Test data/input	Expected Output	When it considered pass/fail	Actual Output
TC1	UC1	Tests if the system allows valid user to log in	Valid email address and password	Display a message "log in successfully"	Pass if the user can log in and view the home page screen	As expected
TC2	UC1	Tests if the system allows invalid user to log in	Invalid username/password	Display a message "Invalid username/Password"	Pass if the user is not able to log in and view the home page screen	As expected
TC3	UC3/4	Tests if the system allows valid user to sign in	Valid input	Takes the user to the homepage	Pass if the user can sign in and view the home page screen	As expected
TC4	UC3/4	Test if the system allows invalid user to sign in	Invalid input	Display a message "Invalid input"	Pass if the user is not able to sign in and view the home page screen	As expected
TC5	UC5	Test if the system allows the valid user to view their profile	User clicks on the "profile" button	User can view their profile	Pass if the user can view their profile	As expected
TC6	UC6	Test if the system allows the valid user to update their profile	User clicks on the "profile" button and change their valid information then click "Save"	Email has been changed	Pass if the user can update their profile information	As expected
TC7	UC6	Test if the system allows the invalid	User clicks on the "profile" button and change their	Display a message "Invalid input"	Pass if the user is not able to update their profile information	As expected

		user to update their profile	invalid information then click "Save"			
TC8	UC7	Test if the system allows the valid user to add new budget section	user presses the plus '+' icon in the budgets and entered valid input	New budget plan has been added successfully	Pass if the user can add new budget section	As expected
TC9	UC7	Test if the system allows the invalid user to add new budget section	user presses the plus '+' icon in the budgets and entered invalid input	Display a message "This Title is already registered, please enter a different title."	Pass if the user is not able to add new budget section	As expected
TC10	UC8	Test if the system allows the valid user to add new Receipt	user presses the plus '+' icon in the bill and entered valid input	New Receipt has been added successfully	Pass if the user can add new Receipt and link it with a purchase if the user wants to	As expected
TC11	UC8	Test if the system allows the invalid user to add new Receipt	user presses the plus '+' icon in the bill and entered invalid input	Display a message "Invalid input"	Pass if the user is not able to add new Receipt	As expected
TC12	UC9	Test if the system allows the valid user to add new saving plan	user presses the plus '+' icon in the savings and entered valid input	New saving plan has been added successfully	Pass if the user can add new saving plan	As expected
TC13	UC10	Test if the system allows the valid user to view Budget details	User presses on the selected budget	Display the selected Budget details	Pass if the system displays the details of the selected Budget	As expected
TC14	UC11	Test if the system allows the valid user to view Saving plan details	User presses on the selected Saving plan	Display the selected Saving plan details	Pass if the system displays the details of the selected Saving plan	As expected
TC15	UC12	Test if the system allows the valid user to view Receipt's Picture	User presses on the selected Receipt	Display the selected Receipt's picture	Pass if the system displays the picture of the selected receipt	As expected
TC16	UC13	Test if the system allows the valid user to view Receipts list	User clicks on the " Receipt" button	Display all the user's Receipts as a list	Pass if the system displays all the receipt	As expected
TC17	UC14	Test if the system allows the valid user to view Budgets list	User clicks on the " Budget" button	Display all the user's Budgets in a list	Pass if the system displays all the budgets	As expected

TC18	UC15	Test if the system allows the valid user to view Saving plans list	User clicks on the " Savings" button	Display all the user's Saving plans in a list	Pass if the system Display all the user's Saving plans in a list	As expected
TC19	UC16	Test if the system allows valid QR Transaction code	Valid QR	Display a message “success”	Pass if the user can pay and view the payment details screen and the user is able to add the Transaction's Receipt picture	As expected
TC20	UC16	Test if the system allows invalid QR Transaction code	invalid QR	Display a message “Invalid QR”	Pass if the user is not able to pay and view the Transaction details screen	As expected
TC21	UC17	Test if the system allows the valid user to View Transaction details	User presses on the selected Transaction	Display the selected Transaction details	Pass if the system Display the selected Transaction details and the user can update the Transaction's Receipt picture	As expected
TC22	UC26	Test if the system allows the valid user to view the financial analysis	User clicks on the financial analysis icon	Display the financial analysis for the user account	Pass if the system Display the financial analysis for the user account	As expected
TC23	UC18	Test if the system allows the valid user to Classify Transaction	User clicks on the "Save" button	Add the payment to the classified budget	Pass if the system added the Transaction to the classified budget	As expected
TC24	UC18	Test if the system allows the invalid user to Classify Transaction	User clicks on the "Save" button	Display a message “Invalid input”	Pass if the system did not add the Transaction to the classified budget	As expected
TC25	UC19	Test if the system allows the valid user to view account details	User clicks on the account details icon	User can view their account details	Pass if the user can view their account details	As expected
TC26	UC20	Test if the system allows the valid user to view transfer options	User clicks on the "transfer" button	User can view the transfer options	Pass if the user can view the transfer options	As expected
TC27	UC21	Test if the system allows the valid user to transfer	User clicks on the "Local Bank" button and enters valid input	User can transfer money	Pass if the user can transfer money	As expected

		money via local bank				
TC28	UC21	Test if the system allows the invalid user to transfer money via local bank	User clicks on the "Local Bank" button and enters invalid input	Display a message "invalid input"	Pass if the user cannot transfer money	As expected
TC29	UC22	Test if the system allows the valid user to transfer money via MyWallet account transfer	User clicks on the "MyWallet Account" button and enters valid input	User can transfer money	Pass if the user can transfer money	As expected
TC30	UC22	Test if the system allows the invalid user to transfer money via MyWallet account transfer	User clicks on the "MyWallet Account" button and enters invalid input	Display a message "invalid input"	Pass if the user cannot transfer money	As expected
TC31	UC23	Test if the system allows the valid user to view previous added bills list	User clicks on the "Bills" button	User can view the previous added bills list	Pass if the user can view the previous added bills list	As expected
TC32	UC24	Test if the system allows the valid user to add new Bill	User presses the plus '+' icon in the bill and enters valid input	User can add new accounting bill	Pass if the user can add new accounting bill	As expected
TC33	UC24	Test if the system allows the invalid user to add new Bill	User presses the plus '+' icon in the bill and enters invalid input	Display a message "invalid input"	Pass if the user cannot add new accounting bill	As expected
TC34	UC25	Test if the system allows the valid user to pay bill	User presses the plus '+' icon in the bill and enters valid input	User can pay bill	Pass if the user can pay bill	As expected
TC35	UC25	Test if the system allows the invalid user to pay bill	User presses the plus '+' icon in the bill and enters invalid input	Display a message "invalid input"	Pass if the user cannot pay bill	As expected

4.6.2 Integration Testing

Test Case ID	Description	Test data/input	Expected Output	When it considered pass/fail	Actual Output
TC36	<i>Test the interface link between the DB and the user profile page</i>	<i>email address phone number password</i>	<i>DB updated & profile information stored in DB</i>	<i>pass if the DB been updated and the information stored in DB</i>	TC36
TC37	<i>Test the link between DB and New Receipt</i>	<i>title image note</i>	<i>The receipt shall be added to the DB</i>	<i>pass if the DB been updated and the information stored in DB</i>	TC37

4.6.3 Acceptance Testing

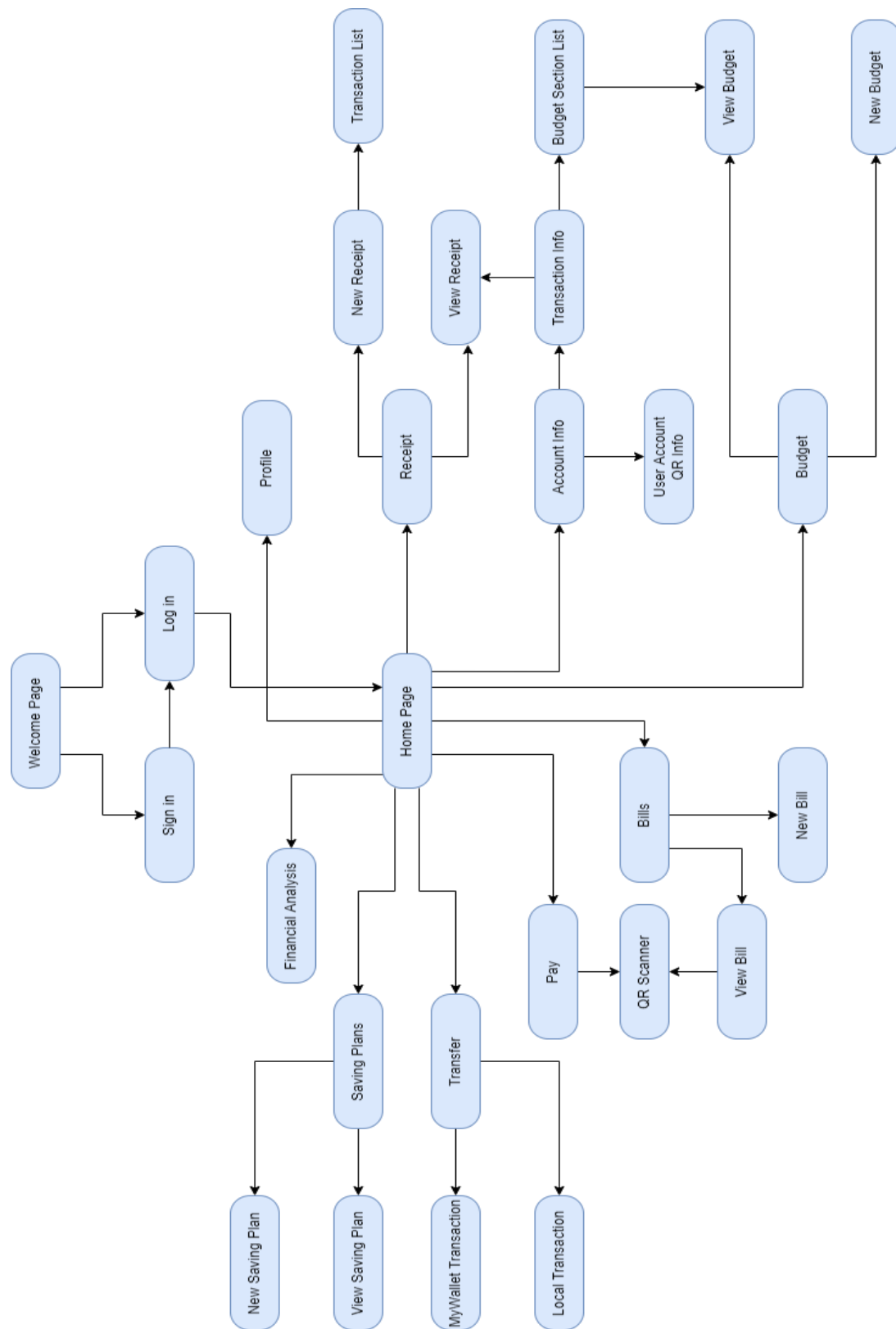
A volunteer customer signed in, the customer wanted to add a new Receipt and link it with a Transaction.

The customer wanted to view a graphical analysis of the amount of increase in savings money but it was not available, the user also mentioned that she would have liked if there was button to change the app theme color and if she can add her credit cards in the application, then she wanted to transfer money to another MyWallet account and it was fast and easy , after that she opened her profile to update her phone number and clicked on the 'SAVE' button and it was successfully updated, the customer mentioned that it will be more convenient if the system automatically without having to click on the "SAVE" button. finally, she added a new Budget section for a new car, and it went smoothly.

Actions to fix defects:

- Color Button:
The system will make the user the ability to change the app theme color to any color she/he want to.
- Auto Saving
The system will save the updated info automatically so it will help the user to update her/his information without worrying about the 'SAVE' button anymore.

4.6.4 Path Testing



4.6.4.1 Path Cyclomatic complexity

$$\text{Cyclomatic complexity} = 31 - 29 + 2 = 4$$

5. Individuals Contributions Breakdown

Member name	Deliverable	Your Contribution (in details)
Reem Alessa	Proposal	Introduction and Statement of Problem
	SRS	Functional Requirements Specification
	SDD-part1	The System Sequence Diagram and MyWallet Transfer Sequence Diagram
	SDD-part2	The System Architecture and Design
	SDD-part3	Unit Testing (TC23-TC35) and Integration Testing
Shroog Alharbi	Proposal	Project's Scope and Project's Objectives
	SRS	User Requirements
	SDD-part1	Sign in and New budget Sequence Diagrams
	SDD-part2	Class Diagram
	SDD-part3	Unit Testing (TC14-TC22) and Path Testing
Enas Alzahrani	Proposal	Related Works and References
	SRS	Customer Statement of Requirements and Glossary of Terms
	SDD-part1	Classify payment and View account details Sequence Diagrams and the State Diagrams
	SDD-part2	Object Diagram
	SDD-part3	Unit Testing (TC1-TC13) and Acceptance Testing

6. References

[1] "Our services," stc pay, 06-Dec-2021. [Online]. Available: <https://stcpay.com.sa/our-services/>. [Accessed: 04-Feb-2022].

[2] S. Al-Absi, "Spendnotes - budget tracker, App Store, 07-Apr-2020. [Online]. Available: <https://apps.apple.com/us/app/spendnotes-budget-tracker/id1505752740>. [Accessed: 04-Feb-2022].

[3] Dictionary.cambridge.org. 2022. *Cambridge English Dictionary: Meanings & Definitions*. [online] Available at: <https://dictionary.cambridge.org/dictionary/english/> [Accessed 5 March 2022].