**Table of Contents**

[**1. Introduction 2**](#_Toc166089848)

[**2. Agile Methodology Selection 2**](#_Toc166089849)

[**2.1 Using Agile Techniques 2**](#_Toc166089850)

[**2.3 Choosing the Tool 2**](#_Toc166089851)

[**3. Documentation and Working Together 3**](#_Toc166089852)

[**3.1 User Stories and Test Cases 3**](#_Toc166089853)

[**3.2 Project Planning 8**](#_Toc166089854)

[**4. Prototype Design 9**](#_Toc166089855)

[**4.1 System Design 9**](#_Toc166089856)

[**5. Using Git Version Control 16**](#_Toc166089857)

[**5.1 Our Git repository had these main parts 16**](#_Toc166089858)

[**6. Pair Programming and OOP Principles 16**](#_Toc166089859)

[**7. Implementation and Testing 17**](#_Toc166089860)

[**8. Conclusion 21**](#_Toc166089861)

[**9. References 21**](#_Toc166089862)

[**10. Appendix 21**](#_Toc166089863)

# Introduction

In today’s fast-paced and competitive world, the development of software systems requires us to follow strong methods and best practices to make robust and good software products. This report helps in understanding how our team used the agile process to make a robust banking software system. The systems use agile methods, Object-Oriented Programming, Git version control system, and automated testing to deliver a high-quality, easy-to-use software system with lots of features that can meet the changing needs of today’s banking world.

# Agile Methodology Selection

We chose Scrum which is a popular Agile framework because it lets us to work in small increments - Sprints (Drumond, 2022) and these practices help the team work together, learn from their experiences, and continuously improve

## **2.1 Using Agile Techniques**

* **User Stories:** We made user stories to understand what our banking software needs to do. Each user story was like “As a… I want to… So that…” which helped us see things from the user’s point of view.
* **Planning Sessions:** A meeting regularly is held to decide which user stories are most important and for planning our work. In these meetings, we talked about user stories, discussed how much work they would take, and decided which one to work on next.
* **Simple Design Principles:** The design of the system is kept simple so that it’s easy to understand and change. We focused on making a minimal viable product – MVP which is a simple version of the software that does the most important things without making things too complicated.
* **Pair Programming:** We encouraged developers to work in pairs which means two people writing code together. This helped us share knowledge, make fewer mistakes, and improve the overall quality of the code.
* **Testing:** We tested our software at every step of the development including unit, integration, and acceptance testing. We carefully made test cases for each user story to make sure that the software worked as expected.

## **2.3 Choosing the Tool**

Trello was used as a tool to help us in project management as Trello allows us to organize our work and tasks into visual boards (Markup Hero – Blog, 2024). Trello is also great for teamwork as well because it lets everyone on the team see the same information and updates in real-time simultaneously.

We made a Trello board for our project and divided it into several lists:

* **Backlog:** We put all the user stories here which are descriptions of what our software needs to do. Each user story also had acceptance criteria which are conditions that need to be met for the story to be considered done.
* **Sprint Planning:** In our planning meetings we picked user stories from the backlog and moved them to this list. Each user story is broken down into smaller tasks and assigned to each team member respectively.
* **In Progress:** Once we started working on a task we moved that into this list. We kept updating the status of the task as we worked on it.
* **Testing:** After we finished developing a task we moved it to this list for testing. Separate cards were made for writing test cases and then recording their results after running them.
* **Done:** Once a task passes all the tests and is finished, it is moved to this list.

In this way, we could see the overall progress of our project to make sure that everything was on track.

# Documentation and Working Together

We used Trello’s feature that lets us comment on cards to save our group discussions, including planning talks, task division, and test case generation. We attached important documents or links to each discussion to give more context. This includes user stories, planning sessions, and other important features.

## **3.1 User Stories and Test Cases**

1. **Account Management**

**User Stories**

**As a bank customer,**

* I want to make a new bank account (either Personal or Business) so I can start using bank services.
* I want to change my account details i.e., user information, account type, etc., to keep my information up to date.
* I want to be able to close my account when I need to.

**Test Case 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input Steps** | **Expected Output** | **Result** |
| Creating a New Account | Navigate to create account section of the banking system.  Select the type of account (Personal/Business).  Enter required personal/business information.  Submit the form to create the account. | A new account is successfully created, and the customer receives confirmation | Pass |

**Test Case 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input** | **Expected Output** | **Result** |
| Edit Account Details | Log in to banking system.  Go to section for account details.  Change user information or account type and then save the changes. | The account details are updated successfully and the changes show up in the system. | Pass |

**Test Case 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input** | **Expected Output** | **Result** |
| Closing an account | Log in to banking system.  Go to the account settings or close account section.  Confirm that you want to close the account. | The account is closed and data related to it is saved securely and the customer gets a confirmation that the account is closed. | Pass |

1. **Money Transactions**

**User Stories**

**As a bank customer,**

* I want to put money into my account and get a receipt for keeping records.
* I want to take out money and see my balance update right away so I can my money easily.
* I want to move money between different accounts in the same bank to manage my money easily.
* I want to see my past transactions with filters (data, type, and amount) to keep track of my money.

**Test Case 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input Steps** | **Expected Output** | **Result** |
| Put money into account | Log in to the banking system.  Navigate to the section for depositing the money.  Enter the amount to deposit and choose the account. | The money is deposited into the chosen account successfully and a receipt is made. | Pass |

**Test Case 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input Steps** | **Expected Output** | **Result** |
| Money withdrawal | Log in to the banking system.  Navigate to the section for withdrawing money  Enter the amount to withdraw and confirm the withdrawal. | The money is withdrawn from the account successfully and the balance updates right away. | Pass |

**Test Case 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input Steps** | **Expected Output** | **Result** |
| Seeing Transaction History | Log in to the banking system.  Navigate to the section for transferring history.  Use filters (data, type, and amount) as needed. | The transaction history shows up according to the filters used. | Pass |

1. **Account Services**

**As a bank customer,**

* I want to see detailed information about my account, including current balance, recent transactions, and account type to manage my money efficiently.
* I want to ask for and manage services like checkbooks and debit/credit cards to meet my banking needs.
* I want to set up and manage regular payments and direct debits to automate my money transactions.

**Test Case 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input Steps** | **Expected Output** | **Result** |
| Seeing detailed account information | Log in to the banking system.  Navigate to the section for account details | Detailed information about the account including current balance, recent transactions, and account type is shown. | Pass |

**Test Case 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input Steps** | **Expected Output** | **Result** |
| Asking for and managing services | Log in to the banking system.  Navigate to the services section.  Choose the service you want like checkbook, debit/credit card.  Follow the steps to ask for or manage the service. | The service the customer asked for is processed successfully and the customer gets a confirmation. | Pass |

**Test Case 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input Steps** | **Expected Output** | **Result** |
| Setting up and managing regular payments/direct debits | Log in to the banking system.  Navigate to the section for regular payments/direct debits.  Add/edit/delete regular payments/direct debits as needed. | Regular payments/direct debits are set up, modified, or deleted successfully as per the customer's instructions. | Pass |

## **3.1 User Stories and Test Cases**

1. **Support and Help Desk**

**User Stories**

**As a bank customer,**

* I want to take assistance about account.
* I want to report fraudulent activity and issues.

**Test Case 1**

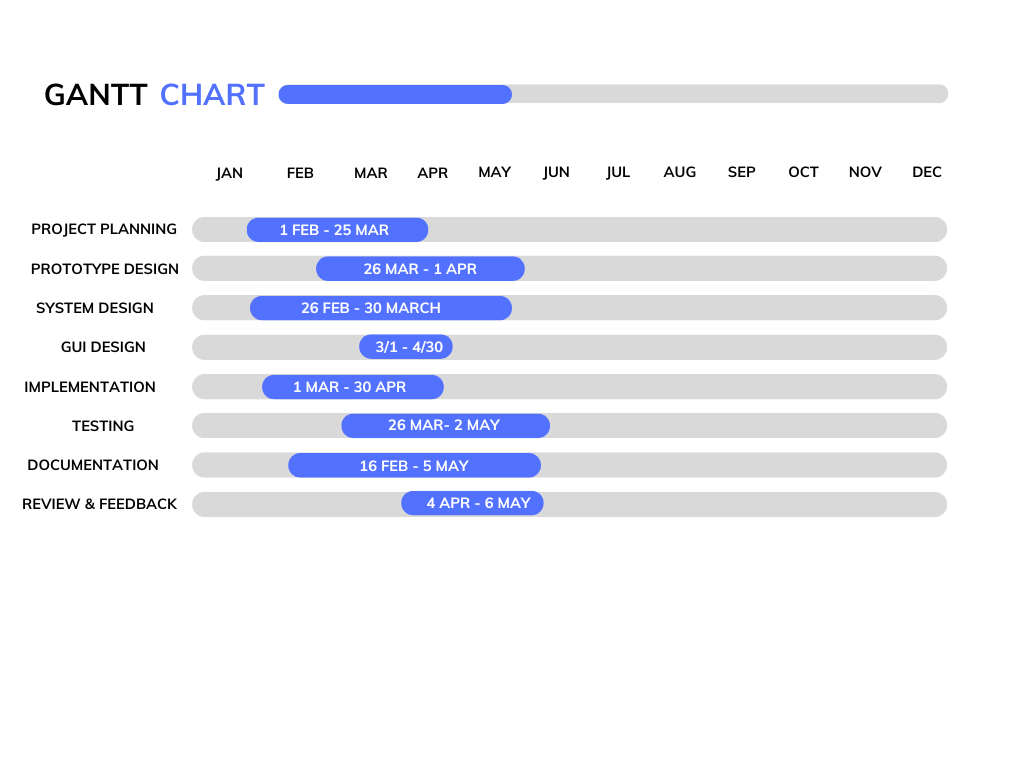
|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input Steps** | **Expected Output** | **Result** |
| Need General Information about the account | In the Help Desk section click on Help.  Select a question to explore the given options. | Question answers must be displayed and contain complete information about the given issue. | Pass |

**Test Case 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Input** | **Expected Output** | **Result** |
| Report Issue | After Log in to the system navigate to the Help Desk option.  Click on Report Issue.  Provide details of the Issue and submit | The issue must be submitted and stored in a database. | Pass |

## **3.2 Project Planning**

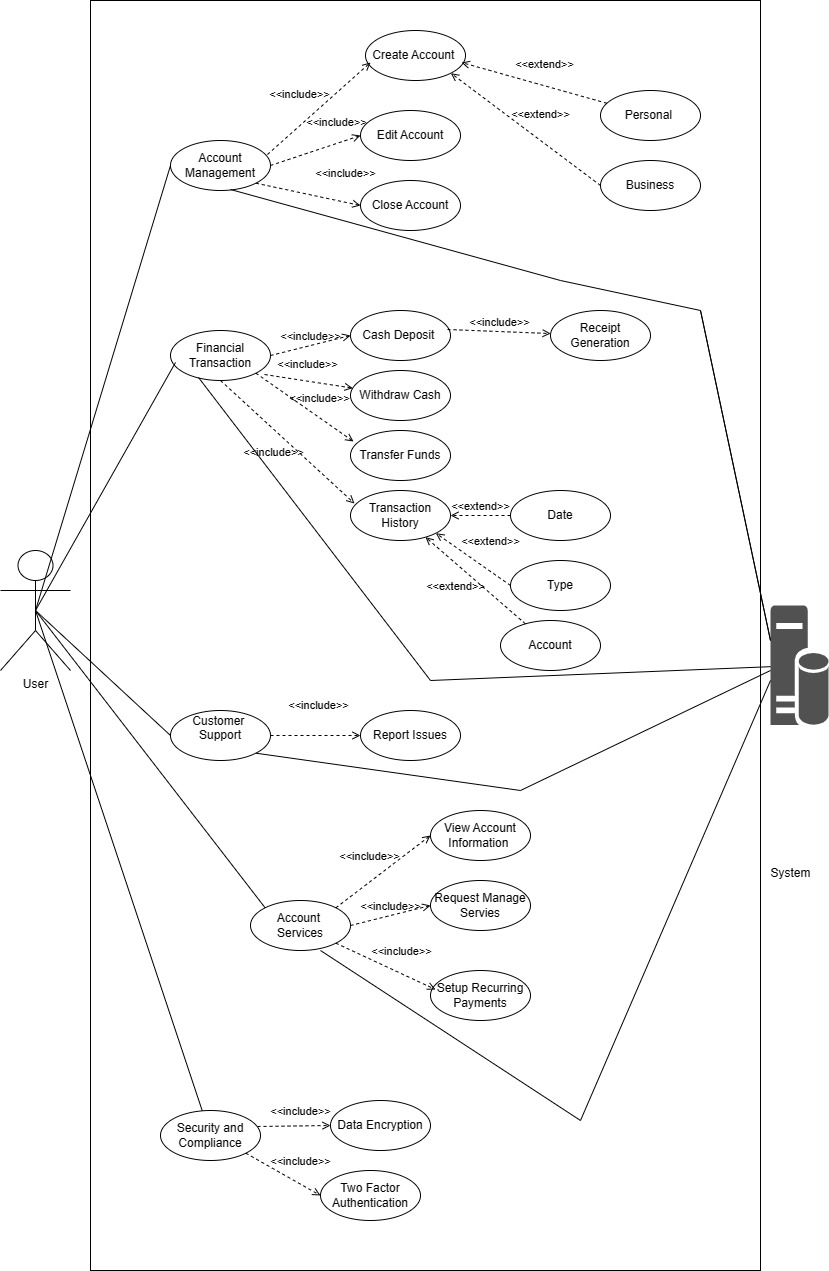
We utilize the Gantt chart to plan different phases of our project.



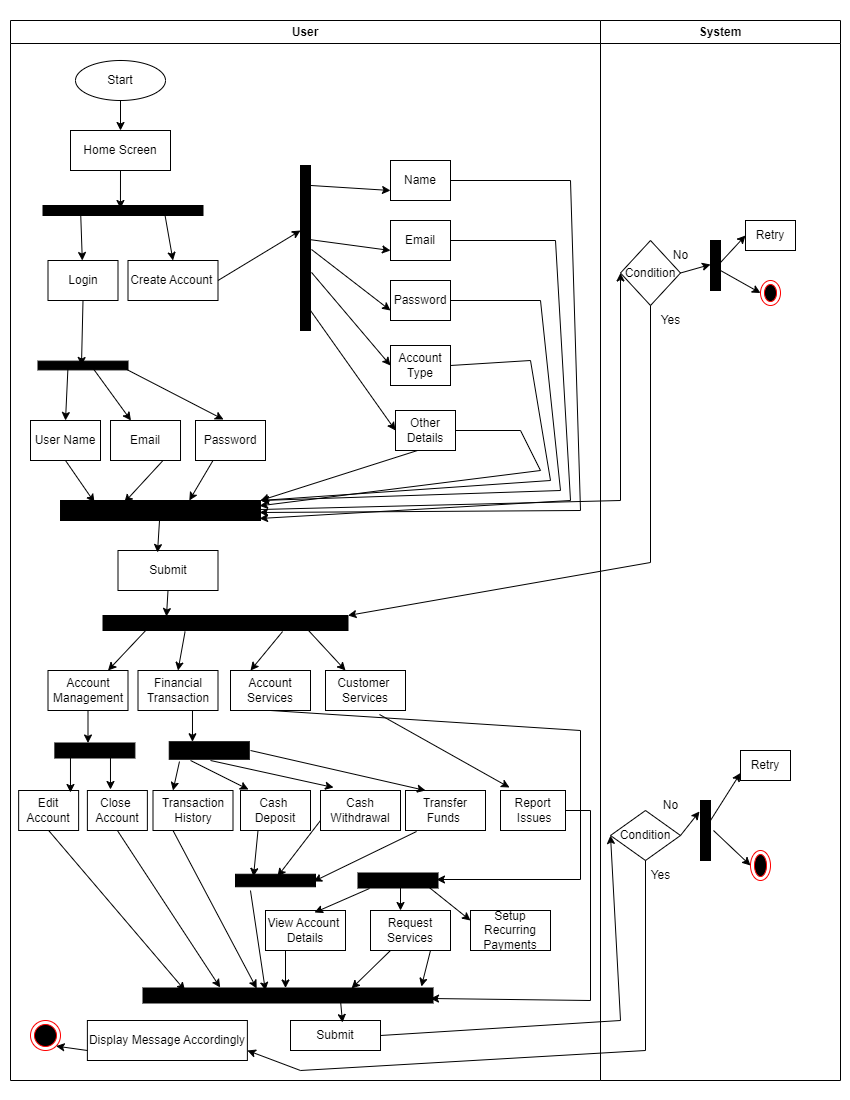
**Figure 1: Gantt Chart**

# Prototype Design

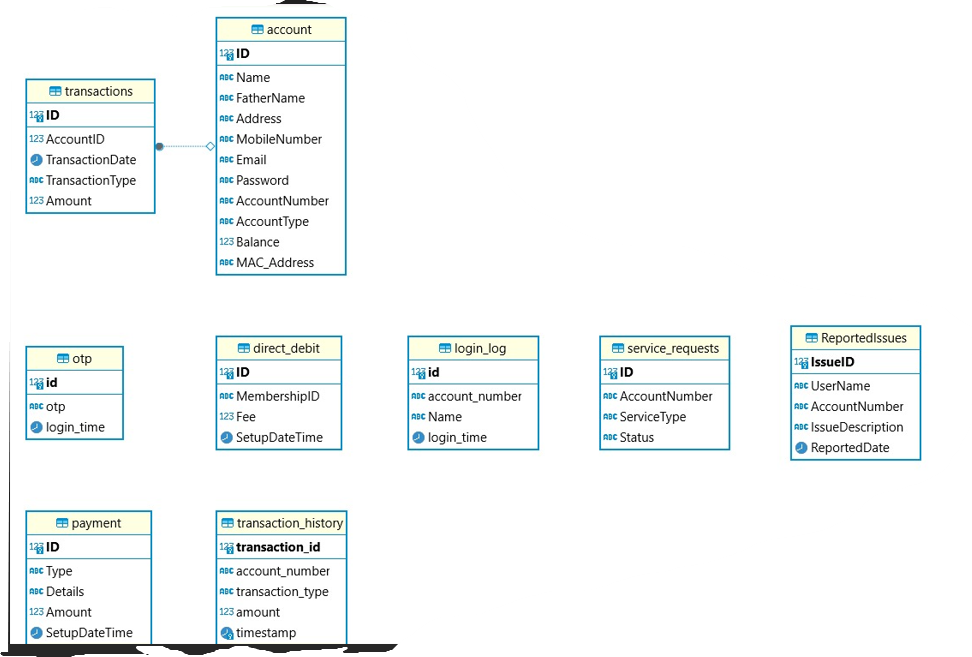
## **4.1 System Design**

****

**Figure 2: Use Case Diagram of System**

****

**Figure 3: Activity Diagram**

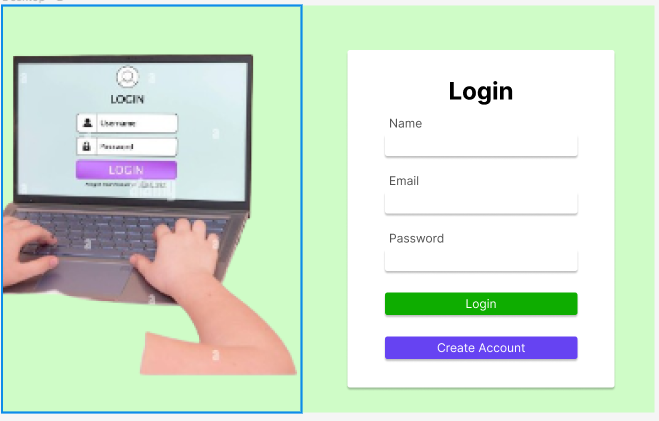


**Figure 4: ERD Diagram**

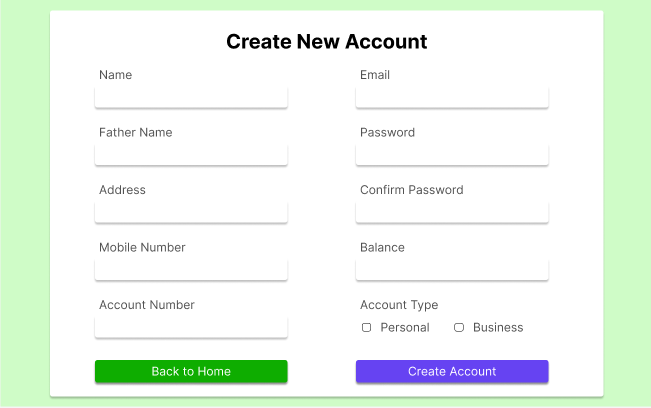
**4.2 GUI Design**

****

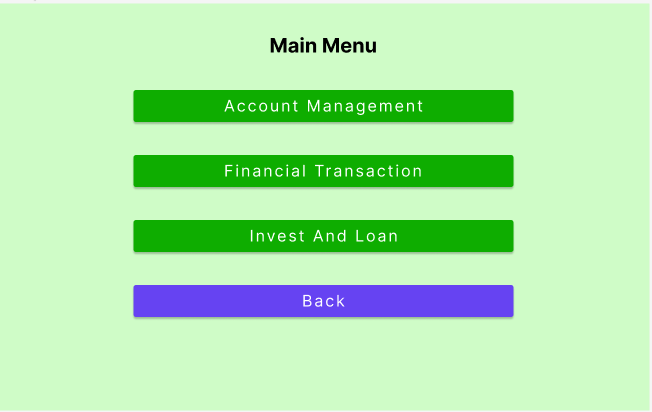
**Figure 5: Home Page**

****

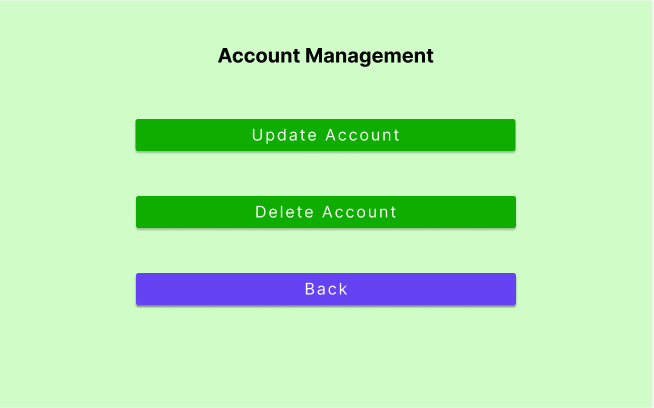
**Figure 6: Login Form**

****

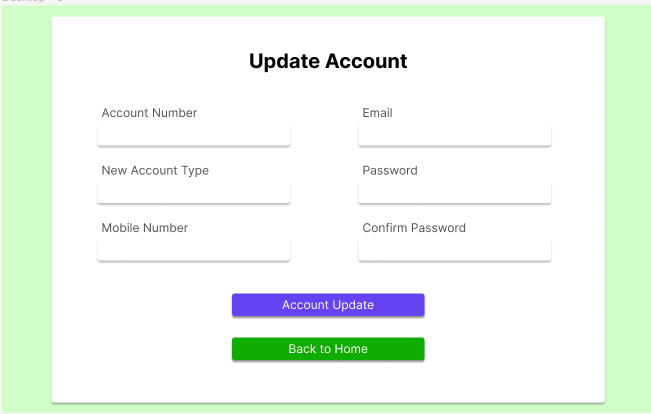
**Figure 7: Create Account**

****

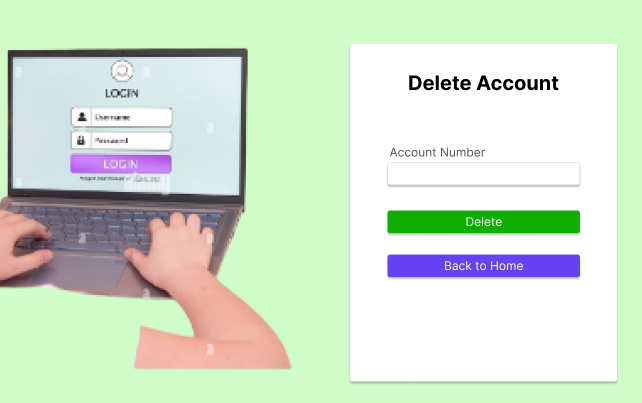
**Figure 8: Main Menu**

****

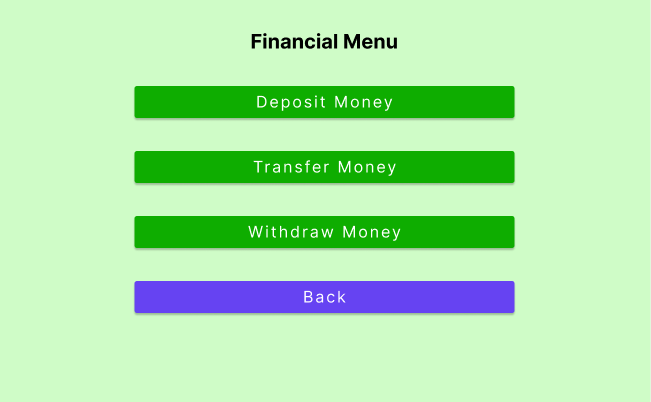
**Figure 9: Account Management**



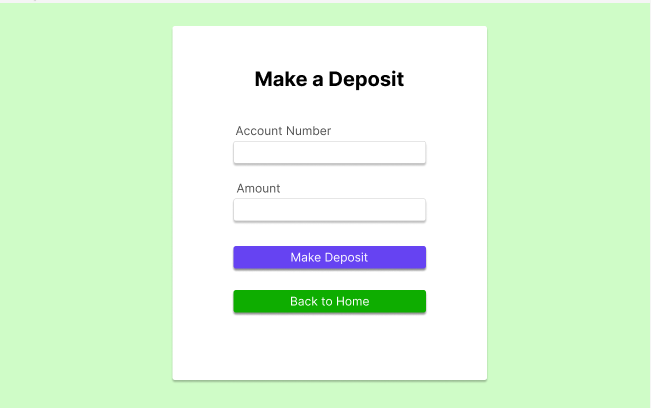
**Figure 10: Update Account**



**Figure 11: Delete Account**



**Figure 12: Financial Menu**



**Figure 13: Make Deposit**

# Using Git Version Control

We used GitHub during the complete development process, which helped us manage all source files related to the banking software system. GitHub provides a centralized space where users can store, track changes, and work together on the same project efficiently without any conflict. By using Git as a version control system we keep a full history of our project, track changes made by each team member and can also go back to the previous version if needed.

## **5.1 Our Git repository had these main parts**

* **Code:** We stored our all coding files related to the development of the banking software system including, Python scripts, configuration files, and any other related files in the public repository that can be accessed through the link given in the appendix. So that, all team members can access the latest version of the code and work together on development tasks easily and in a parallel way.
* **Documentation:** This document is also stored in the public repository that contains different information related to the project including design documents, user stories, test cases, and other relevant information. These documents provide valuable information about the system architecture, design decisions, and details about how we implemented the different functionalities which helped us work together and share knowledge.
* **Prototype and Testing Artifacts:** Prototype design consists of an Entity-relationship diagram, use case diagrams, activity diagram, and GUI Mockups designed in Figma are also included in the GitHub repository to help us see the system architecture and interface designs, giving us a reference for implementing features.
* **Evidence of Developed System:** The GitHub repository also includes a short video in which features of systems are explained and helps users to understand how the system is working. This video serves as proof of the development of artifacts, and skills we gained while developing the system, and it also stakeholders to see the system in action.

# Pair Programming and OOP Principles

Besides Agile methods, our team also used pair programming to make our code efficient and shareable. Pair programming is an Agile development technique in which two developers work together on one computer at a project with one person writing the code and the other watching and giving feedback (Codecademy, n.d.). By following this collaborative approach we were able to identify the bugs and errors at earlier stages and also allowed us to share data and make sure we followed the best practices for our code.

We used Python programming language for the development of the banking software which is an Object-Oriented Programming language. We applied different OOP principles during the development phase including, encapsulation, polymorphism, and inheritance (Erinc, 2020).

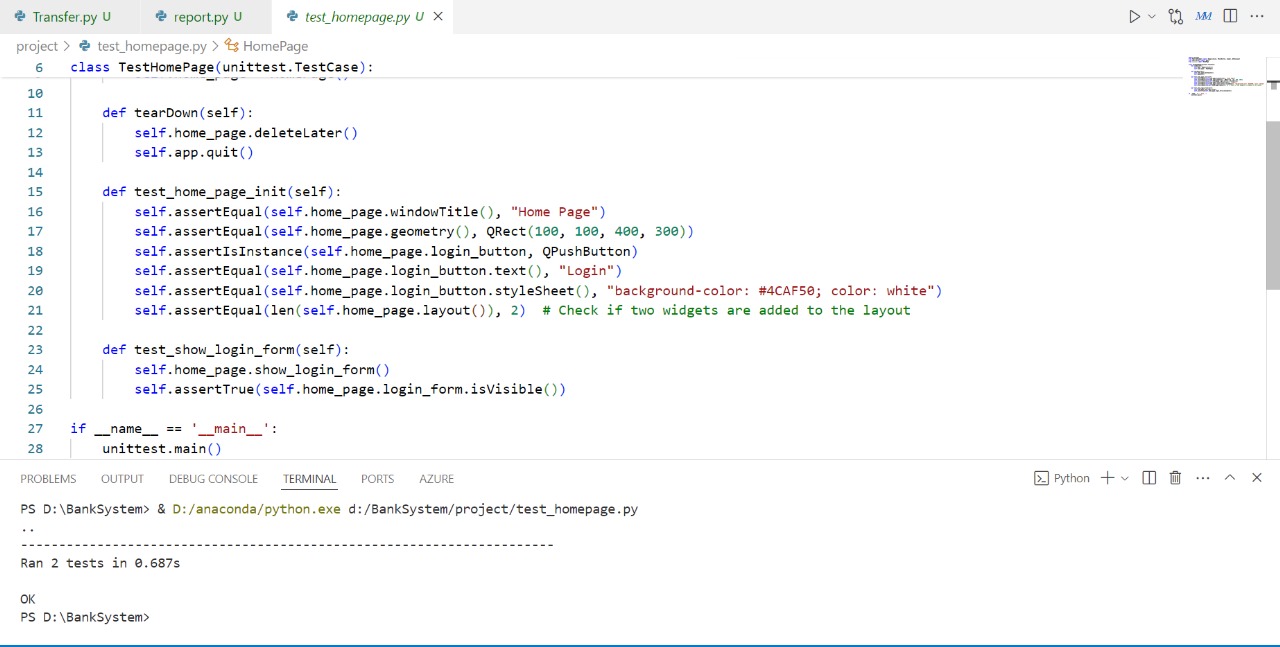
* **Encapsulation:** We limit access by putting variables and methods into classes to keep the data secure and ensure data integrity. For example, we keep the attributes private and methods public to access those private attributes to make sure object consistency and protected from external interference.
* **Polymorphism:** We utilized the concept of polymorphism to allow the interaction between objects of different classes in the same way. We used the concept of polymorphism to let objects of different classes be used in the same way. For instance, we write code for abstract base classes and different subclasses that share a common way of working. Polymorphism enables to write code in such a way that it can also with any subclass without knowing their types.
* **Inheritance:** Inheritance is also utilized while writing the code for the project. Inheritance allows writing parent classes so that child classes can also access their properties and methods (Erinc, 2020). The creation of Parent classes with common attributes and methods helps to increase the reusability of code and also makes it more modular and easy to understand. This approach makes our code base more scalable and maintainable.

By utilizing the pair programming and OOP principles we ensure that our code is strong, maintainable, and can be extended if need to add more functionalities. Working as a team and by following good design principles we were able to make a banking software system in such a way that we met the project’s needs while showing our skills in Agile methods and best practices in software engineering respectively.

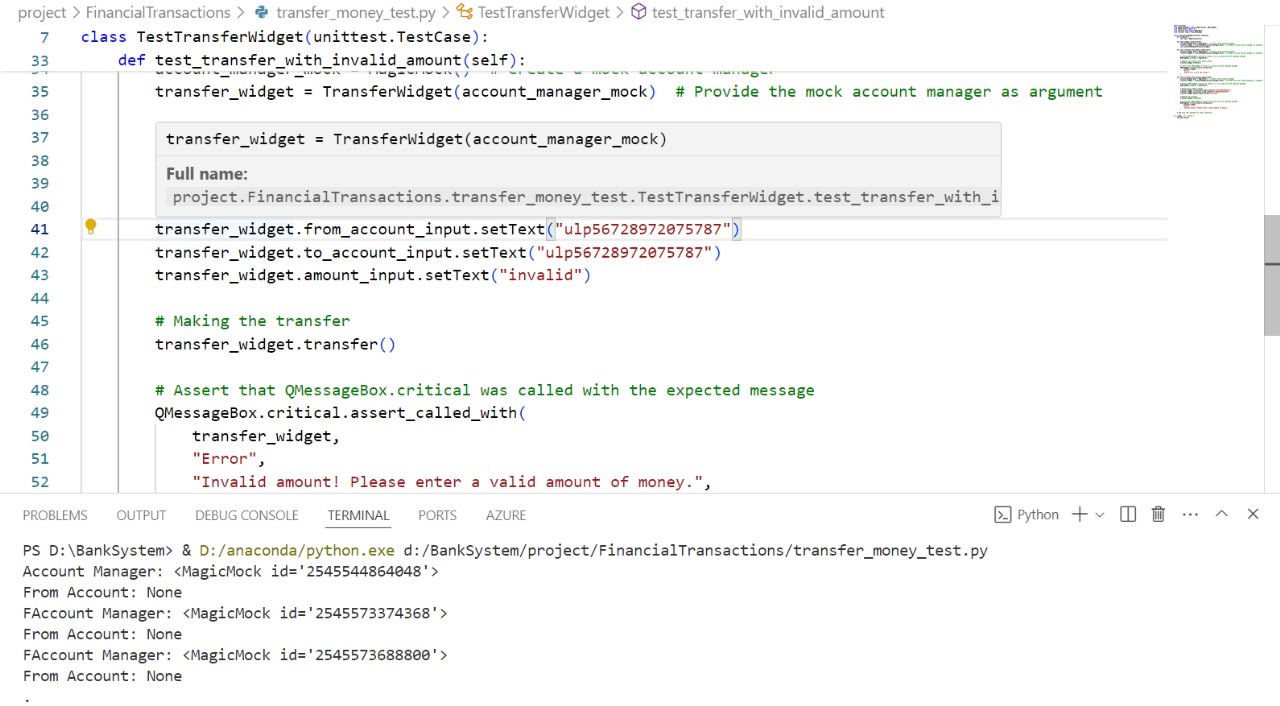
# Implementation and Testing

Unittest tool is used for implementing automated testing to verify and validate our code for various functionalities and to ensure it works as intended. Unittest is a popular testing tool in Python known for its simplicity, scalability, and additional plugins for testing. We used built-in Unittest features to write and execute automated tests efficiently and to perform different types of testing such as unit testing, integration testing, and regression testing to check different functionalities of our software system.

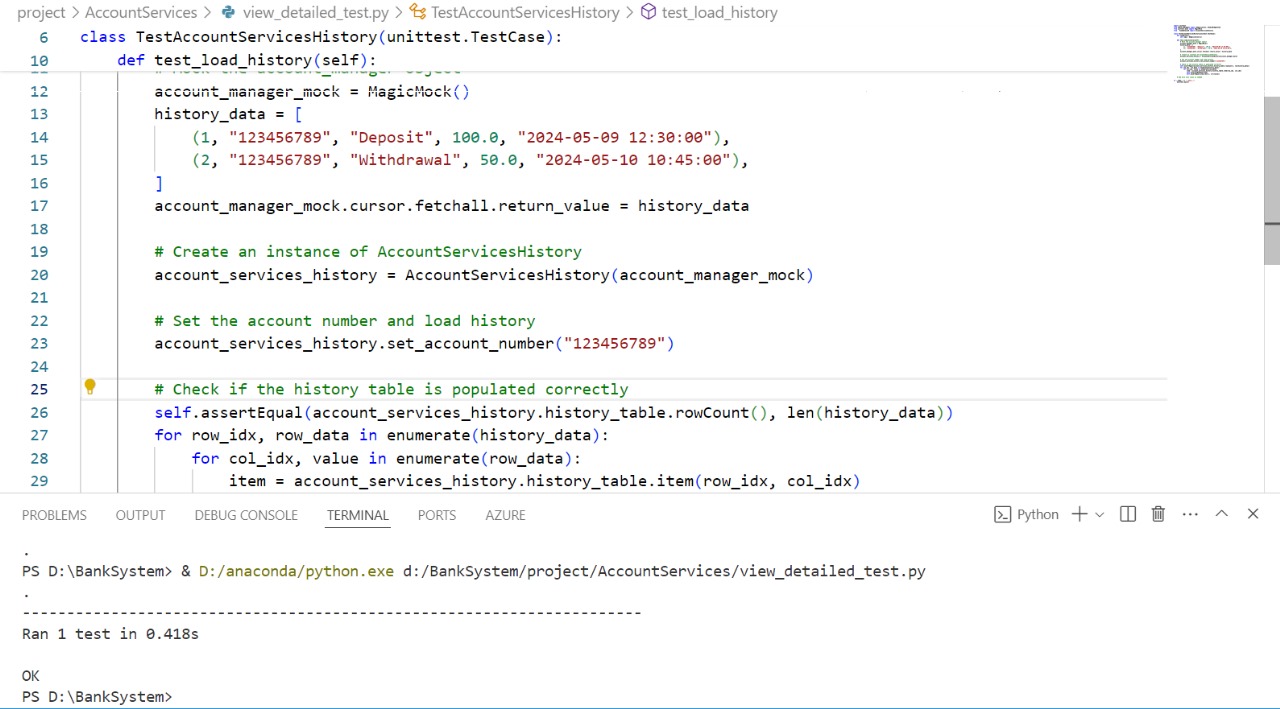
Unittest gave us a flexible and clear way to write test cases letting us create comprehensive tests that covered a lot of scenarios and edge cases (Python, n.d). We used Unittest fixtures, parametrization, and markers to organize our tests and make them easier to write, helping us write clean, maintainable, and reusable test code. By the use of Unittest in our testing we were able to find and fix issues early in the development process resulting in a stronger and more reliable software system. This helped us do continuous testing and integration letting us work quickly and deliver high-quality code to our stakeholders.



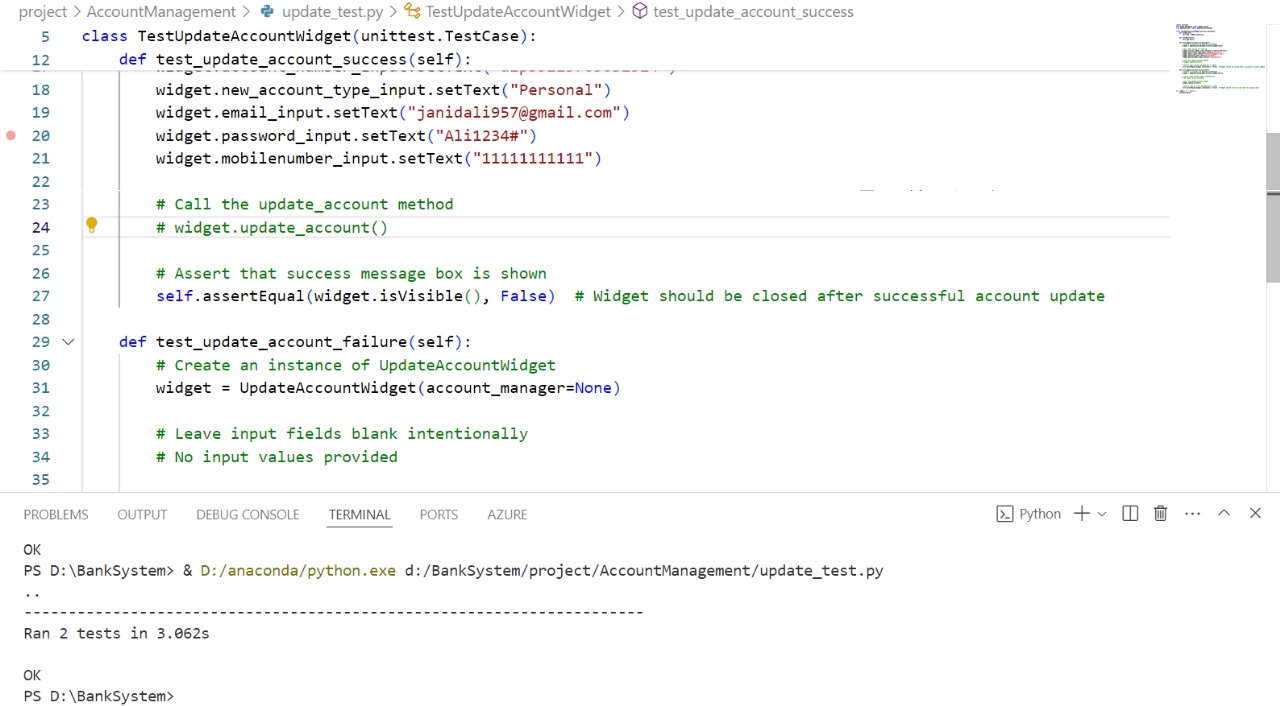
**Figure 14: Home Page**



**Figure 15: Transfer Money Test**



**Figure 16: View Account Detail Test**



**Figure 17: Update Account Test**



**Figure 18: Help Desk Test**

# Conclusion

In conclusion, the use of Agile methods along with effective tools, documentation, working together, and testing practices helped us deal with the complexities of software development and deliver a successful banking software system. By prioritizing the stakeholder value, working together, and using cutting-edge technologies we have set the stage for future innovation and growth in the ever-changing world of software engineering.

# References

Drumond, C. (2022). Scrum - what it is, how it works, and why it’s awesome. [online] Atlassian. Available at: <https://www.atlassian.com/agile/scrum>.

Markup Hero - Blog. (2024). How to Use Trello for Project Management. [online] Available at: https://markuphero.com/blog/how-to-use-trello-for-project-management [Accessed 3 May 2024].

Erinç, Y.K. (2020). The SOLID Principles of Object-Oriented Programming Explained in Plain English. [online] freeCodeCamp.org. Available at: <https://www.freecodecamp.org/news/solid-principles-explained-in-plain-english/>.

Python, R. (n.d.). Effective Python Testing With Unittest – Real Python. [online] realpython.com. Available at: [https://realpython.com/Unittest-python-testing/](https://realpython.com/pytest-python-testing/).

# Appendix

GitHub repository link: <https://github.com/Sukhaman906214/BankSystem.git>