



SOFTWARE REQUIREMENT SPECIFICATION FOR AN IMPLEMENTATION OF BANKING SOFTWARE SYSTEM

|  |  |
| --- | --- |
| **Submitted by** | **Registration No** |
| Abdul Rehman | 28833132@students.lincoln.ac.uk |
| Jawad Amin | 28831848@students.lincoln.ac.uk |
| Abdullah Wajid | 28641089@students.lincoln.ac.uk |
| Hammad Shabir | 28761136@students.lincoln.ac.uk |

|  |  |
| --- | --- |
| Semester | 1st |
| Discipline | CMP9134M Advanced Software Engineering |
| Subject | Advanced Software Engineering |

**DEPARTMENT OF COMPUTER ENGINEERING**

**VERSION 1.0**

DOCUMENT HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **DOCUMENT REVISION HISTORY** | | | |
| Date | Document Version | Revision Description | Author |
| 27/04/2024 | 1.0 | Initial Version |  |
| Wdw |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

TABLE OF CONTENTS

[1. Agile Methodology 4](#_Toc166163507)

[1.1. Scenario of Case Study 5](#_Toc166163508)

[**1.2.** **From Customer Point of View** 5](#_Toc166163509)

[**1.2.1.** **User Stories** 5](#_Toc166163510)

[**1.3.** **From Developer Point of View** 6](#_Toc166163511)

[**1.3.1.** **Stories Cards** 6](#_Toc166163512)

[2. Role, Tasks, and Contributions during the Development Process 8](#_Toc166163513)

[**2.1.** **Abdul Rehman’s Role** 8](#_Toc166163514)

[**2.1.1.** **Software System Designer** 8](#_Toc166163515)

[**2.2.** **Tasks and Contributions** 8](#_Toc166163516)

[**2.2.1.** **Requirements Analysis** 8](#_Toc166163517)

[**2.2.2.** **System Design** 8](#_Toc166163518)

[**2.3.** **Jawad Amin’s Role** 8](#_Toc166163519)

[**2.3.1.** **Software Developer and Scrum Master** 8](#_Toc166163520)

[**2.4.** **Tasks and Contributions** 8](#_Toc166163521)

[**2.4.1.** **Implementation** 8](#_Toc166163522)

[**2.4.2.** **Documentation** 9](#_Toc166163523)

[**2.4.3.** **Code Reviews** 9](#_Toc166163524)

[**2.4.4.** **Collaboration and Communication** 9](#_Toc166163525)

[**2.5.** **Abdullah Wajid’s Role** 9](#_Toc166163526)

[**2.5.1.** **Software Quality Engineer** 9](#_Toc166163527)

[**2.6.** **Tasks and Contributions** 9](#_Toc166163528)

[**2.6.1.** **Testing** 9](#_Toc166163529)

[**2.6.2.** **Code Reviews** 9](#_Toc166163530)

[**2.7.** **Hammad Shabir’s Role** 10](#_Toc166163531)

[**2.7.1.** **Software Developer** 10](#_Toc166163532)

[**2.8.** **Tasks and Contributions** 10](#_Toc166163533)

[**2.8.1.** **Implementation** 10](#_Toc166163534)

[**2.8.2.** **Implementation** 10](#_Toc166163535)

[**2.8.3.** **Testing** 10](#_Toc166163536)

[**2.8.4.** **Documentation** 10](#_Toc166163537)

[**2.8.5.** **Code Reviews** 10](#_Toc166163538)

[**2.8.6.** **Collaboration and Communication** 11](#_Toc166163539)

[3. Gantt Chart 12](#_Toc166163540)

[4. System Features 17](#_Toc166163541)

[**4.1.** **Functional Requirements** 17](#_Toc166163542)

[**4.1.1.** **User Authentication and Authorization** 17](#_Toc166163543)

[**4.1.2.** **Account Management** 17](#_Toc166163544)

[**4.1.3.** **Financial Transactions** 17](#_Toc166163545)

[**4.1.4.** **Loan Management** 17](#_Toc166163546)

[**4.1.5.** **Customer Support** 17](#_Toc166163547)

[**4.1.6.** **Account Statements and Reports** 17](#_Toc166163548)

[**4.1.7.** **Online Bill Payment** 17](#_Toc166163549)

[**4.2.** **Non-Functional Requirements** 18](#_Toc166163550)

[**4.2.1.** **Security** 18](#_Toc166163551)

[**4.2.2.** **Performance** 18](#_Toc166163552)

[**4.2.3.** **Scalability** 18](#_Toc166163553)

[**4.2.4.** **Reliability** 18](#_Toc166163554)

[**4.2.5.** **Backup and Recovery** 18](#_Toc166163555)

[**4.2.6.** **Compliance** 18](#_Toc166163556)

[**4.2.7.** **Usability** 19](#_Toc166163557)

[**4.3.** **Design** 19](#_Toc166163558)

[**4.3.1.** **Use Case Diagram** 19](#_Toc166163559)

[**4.3.2.** **Class Diagram** 20](#_Toc166163560)

[**4.3.3.** **Sequence Diagram** 20](#_Toc166163561)

[**4.3.4.** **State chart Diagram** 21](#_Toc166163562)

[**4.3.5.** **DFD0 and DFD1 Level Diagram** 23](#_Toc166163563)

[**4.4.** **Graphical User Interface (GUI) Design** 23](#_Toc166163564)

[**4.4.1.** **Login Module** 23](#_Toc166163565)

[**4.4.2.** **New Account Module** 24](#_Toc166163566)

[**4.4.3.** **Balance Enquiry Module** 24](#_Toc166163567)

[**4.4.4.** **Credit Amount Module** 25](#_Toc166163568)

[**4.4.5.** **Debit Amount Module** 25](#_Toc166163569)

[**4.4.6.** **All Transaction Module** 26](#_Toc166163570)

[**4.4.7.** **Delete Account Module** 26](#_Toc166163571)

[**4.4.8.** **Cash Deposit Module** 27](#_Toc166163572)

[**4.4.9.** **Cash Withdraw Module** 28](#_Toc166163573)

[**4.4.10.** **Transfer Money Module** 28](#_Toc166163574)

[**4.4.11.** **Test Cases** 28](#_Toc166163575)

[5. Advances in Software Processes 31](#_Toc166163576)

[**5.1.** **Agile Processes** 31](#_Toc166163577)

[**5.2.** **Software Engineering Techniques** 31](#_Toc166163578)

[**5.2.1.** **Test Cases** **Project Management** 31](#_Toc166163579)

[**5.2.2.** **Prototype Design** 31](#_Toc166163580)

[**5.2.3.** **Version Control** 32](#_Toc166163581)

[**5.3.** **Impact on Society** 32](#_Toc166163582)

[**5.3.1.** **Ethical Impact** 32](#_Toc166163583)

[**5.3.2.** **Entrepreneurial Impact** 32](#_Toc166163584)

[6. Github Repo Link: 33](#_Toc166163585)

**Section 1**

**Software Development Methodology**

# **Agile Methodology**

I would like to use Agile methodology. Agile methodology is a project management approach. The agile technique anticipates change and provides far more flexibility than conventional methods. Clients may make minor target adjustments without significantly altering the budget or timeline. The technique entails breaking down each project into prioritised needs and delivering them separately throughout an iterative cycle. Iteration is the process of creating little pieces of a project at a time. The development team and the customer examine and appraise each iteration. The evaluation results are utilised to decide the next phase in development. Clients attend prescheduled frequent meetings to discuss work performed during the previous iteration and arrange work for the next iteration. Each iteration meeting includes detailed objectives such as anticipated changes, time estimates, priorities, and budgets.

**Why?**

The agile technique is built on prioritising client input from the beginning of the development cycle. The goal is to keep the customer engaged throughout the process so that they finish up with a product that they are satisfied with. This strategy saves the customer money and time by having them test and approve the product at each stage of creation. If there are any flaws or issues, improvements may be made throughout the manufacturing cycle to address the problem. Traditional project management approaches would not detect flaws as quickly since they do not test as often. Typically (in conventional manufacturing procedures), flaws that are not detected at the various stages might make their way into the final product. This may result in higher overhead costs and client discontent.

Businesses have proved that this project management strategy works by increasing customer satisfaction. This methodology offers many benefits for firms, including lower costs and improved product satisfaction via customer involvement in development choices.

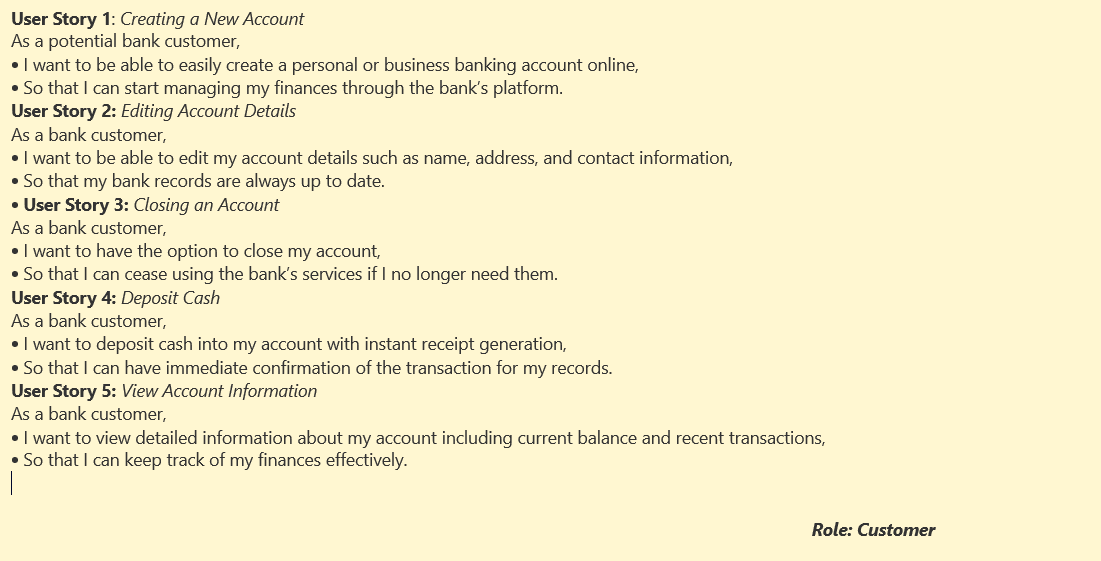
* Encourages free communication between team members and clients.
* Giving teams a competitive edge by identifying problems and implementing adjustments throughout the development process, rather than at the end.
* Reduces assessment time by focusing on certain aspects of the project.
* Consistent assessments throughout the development process enable faster and more efficient adjustments to ensure the product meets desired goals.
* Regular meetings with clients and access to project data provide transparency and accountability.
* This project management style saves time and money for clients and allows for flexible modifications during the development process.

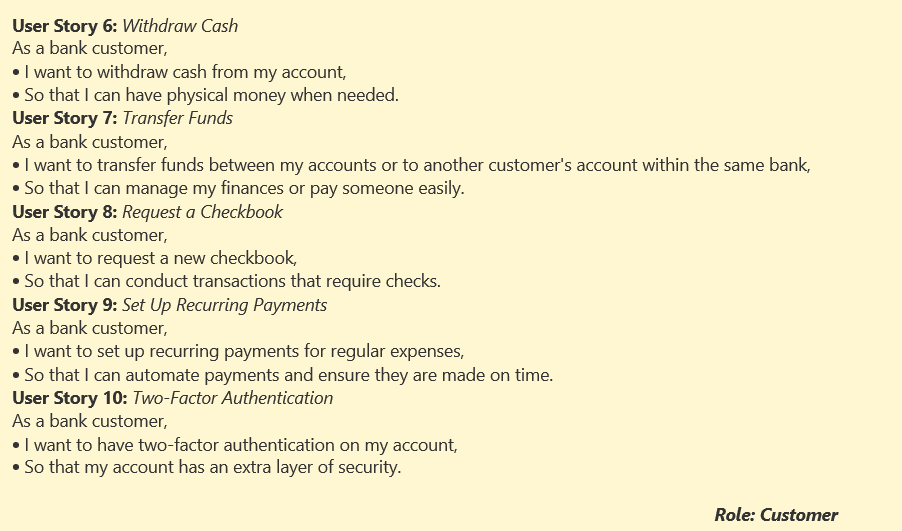
# **Scenario of Case Study**

A customer came to me and wanted to develop a Bank Software System for better management of their customers. As the customer is a non-technical person and did not know about the technicalities of a system so he only provided me with what he wanted and guided me about the requirement of the system in the form of user stories.

## **From Customer Point of View**

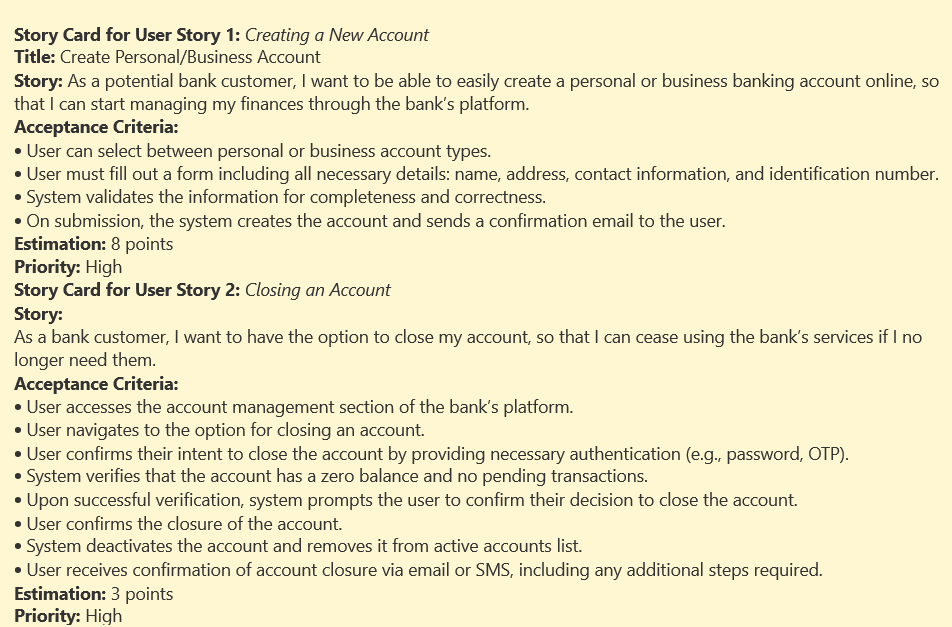
### **User Stories**

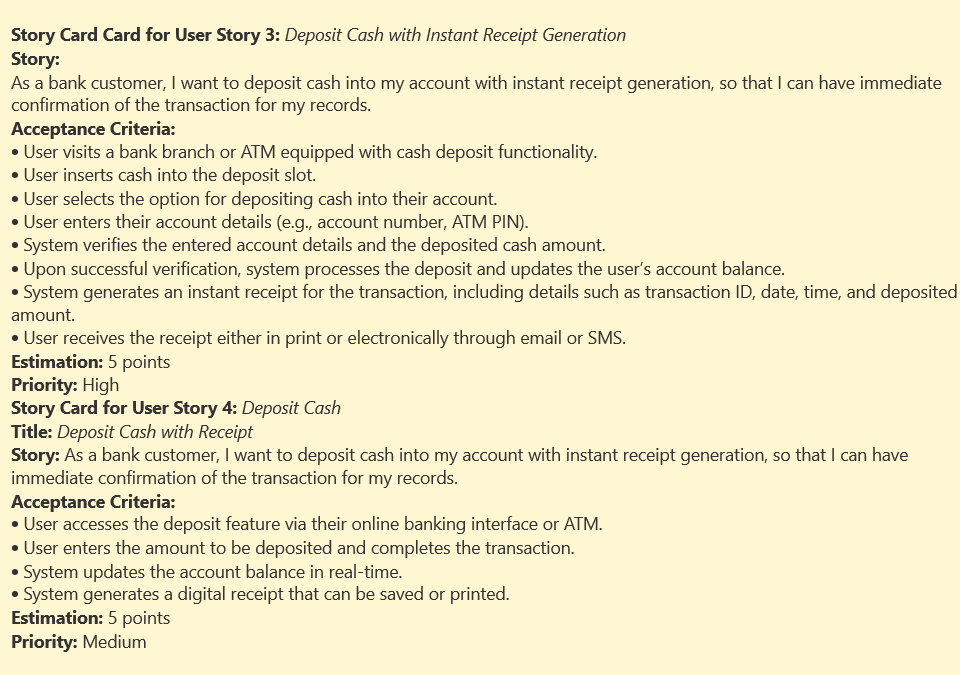


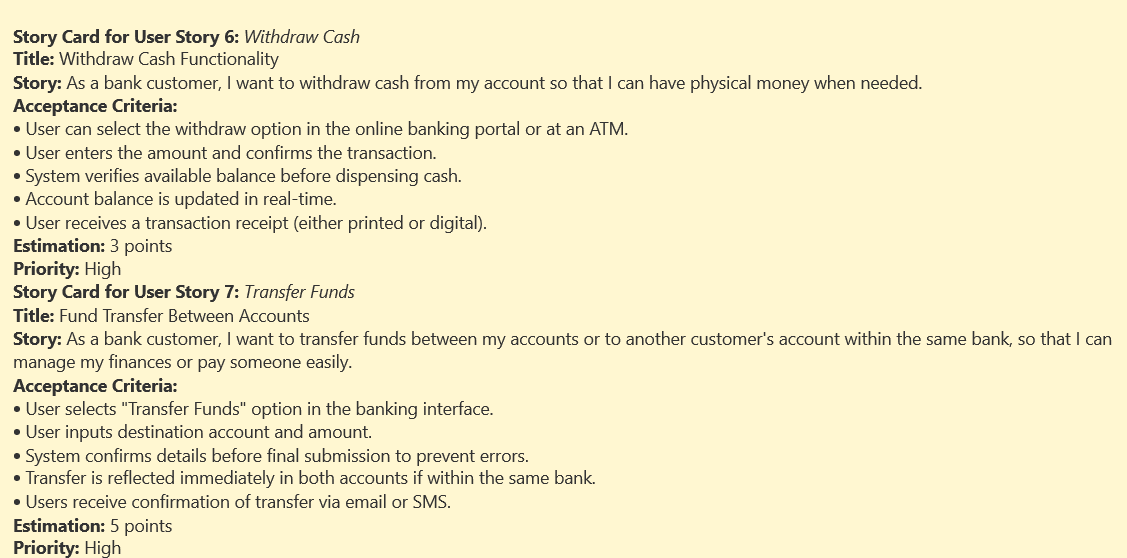


## **From Developer Point of View**

### **Stories Cards**







A yellow paper with black text

Description automatically generated

**Section 2**

**Role, Tasks, and Contributions during the Development Process**

The experience of the development of a banking system encompassed many tasks and contributions from my side of the coin which are imperative for the smooth performance of the software.

# **Role, Tasks, and Contributions during the Development Process**

## **Abdul Rehman’s Role**

### **Software System Designer**

* Abdul Rehman’s major role was the designer of the software system for banking institute.

## **Tasks and Contributions**

### **Requirements Analysis**

* Involved in discussion to understand the requirements of banking system.
* Resolved ambiguities, implementing the suggested features, and identifying challenges encountered along the wave.

### **System Design**

* Worked as a part of the team and involved in the creation of system architecture and database schema design.
* Combined my knowledge in design patterns to improve the scalability, maintainability, and modularity of the code.

## **Jawad Amin’s Role**

### **Software Developer and Scrum Master**

* Jawad’s major role was the creation of the software system as a software developer and acting as a scrum master for the monitoring of project for banking institute.

## **Tasks and Contributions**

### **Implementation**

* We built the banking system and realized the needed figures in line with the stated terms.
* Followed coding best practices, according to the provided code standards.
* On the other hand, Object-Oriented Programming principles, were used for design and implementation of classes and modules.
* I shared workload from pair programming sessions to review each other code and give each other feedback.
* I worked on the development of Credit Amount, Debit Amount, All Transaction, Delete Account, Cash Deposit, Cash Withdraw, Transfer Money Module.

### **Documentation**

* I documented the codebase that contains: Class definitions, method signatures and usage instruction.
* Picked up user manuals and technical documentation writing for end-users and developers.

### **Code Reviews**

* I had the chance to participate in code reviews, during which I was able to offer constructive comments on my colleagues' code.
* Reviewed pull requests to ensure the quality of code, the compliance with the required standards, and consistency with the requirements.

### **Collaboration and Communication**

* Participated extensively in team-level meetings such as sprint planning and daily stand-up and retrospective meetings.

Communicated with team members in response to the progress, difficulty and suggestions.

## **Abdullah Wajid’s Role**

### **Software Quality Engineer**

* Abdullah’s major role was for the testing of the software system for the banking institute as SQA.

## **Tasks and Contributions**

### **Testing**

* Building of the automated unit test suites using the Frameworks like JUnit was done for the purpose of confirming the rightness of the individual parts.
* Participated in module integration to ensure the correctness of working together with other modules.
* Having the user interface and user experience tested sufficiently manual technique was used.

### **Code Reviews**

* I had the chance to participate in code reviews, during which I was able to offer constructive comments on my colleagues' code.

Reviewed pull requests to ensure the quality of code, the compliance with the required standards, and consistency with the requirements.

## **Hammad Shabir’s Role**

### **Software Developer**

* Hammad’s major role was the creation of the software developer for banking institute.

## **Tasks and Contributions**

### **Implementation**

* We built the banking system and realized the needed figures in line with the stated terms.
* Followed coding best practices, according to the provided code standards.
* On the other hand, Object-Oriented Programming principles, were used for design and implementation of classes and modules.
* I shared workload from pair programming sessions to review each other code and give each other feedback.
* I worked on the Login Module, New Account Module, Balance Enquiry Module of the system.

### **Implementation**

* We built the banking system and realized the needed figures in line with the stated terms.
* Followed coding best practices, according to the provided code standards.
* On the other hand, Object-Oriented Programming principles, were used for design and implementation of classes and modules.
* I shared workload from pair programming sessions to review each other code and give each other feedback.

### **Testing**

* Building of the automated unit test suites using the Frameworks like JUnit was done for the purpose of confirming the rightness of the individual parts.
* Participated in module integration to ensure the correctness of working together with other modules.
* Having the user interface and user experience tested sufficiently manual technique was used.

### **Documentation**

* We documented the codebase that contains: Class definitions, method signatures and usage instruction.
* Picked up user manuals and technical documentation writing for end-users and developers.

### **Code Reviews**

* I had the chance to participate in code reviews, during which I was able to offer constructive comments on my colleagues' code.
* Reviewed pull requests to ensure the quality of code, the compliance with the required standards, and consistency with the requirements.

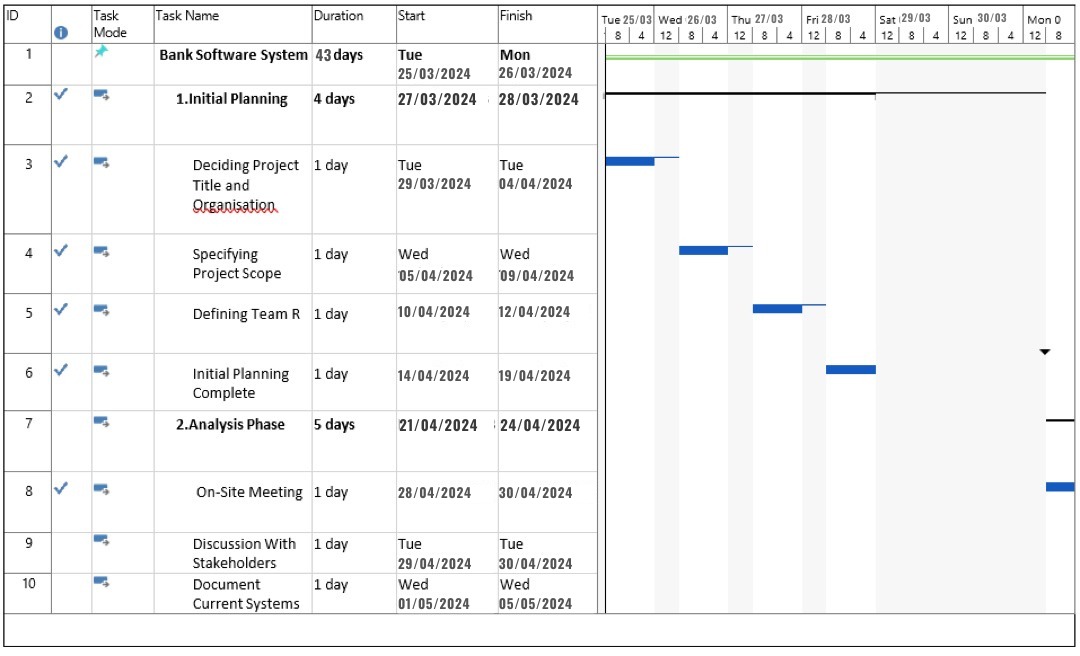
### **Collaboration and Communication**

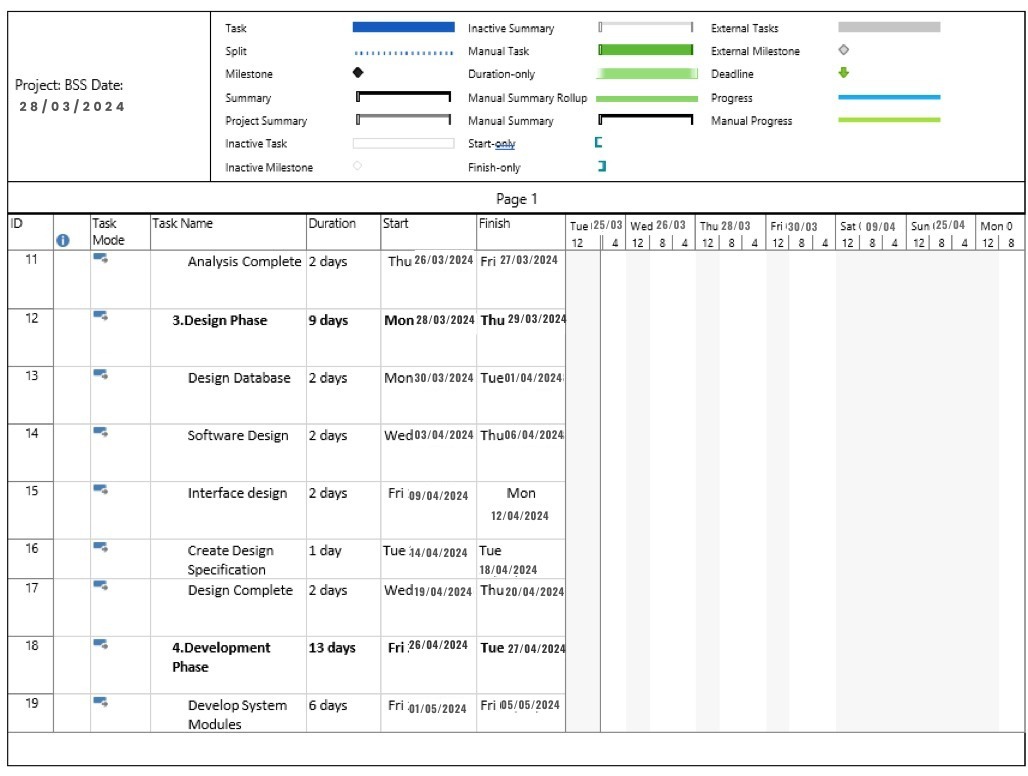
* Participated extensively in team-level meetings such as sprint planning and daily stand-up and retrospective meetings.
* Communicated with team members in response to the progress, difficulty and suggestions.

**Section 3**

**Project Planning**

# **Gantt Chart**

****

****

****

****

****

**Section 4**

**Prototype Design**

# **System Features**

## **Functional Requirements**

### **User Authentication and Authorization**

* The BSS is the system that makes the registration of new users and their account creation possible.
* Login: A user should be able to login with username and the complete password.
* Roles: User privileges and access should be allocated based on the role they are playing in the system.

### **Account Management**

* The BSS offers a lot of possibilities for the customers to open any type of account including saving, and fixed deposit.
* The account holders will easily be able to see both their transaction history and their available balance.
* They will be able to perform the inter-accounts transactions within their own accounts or between other accounts.

### **Financial Transactions**

* Deposit cash with receipt generation.
* Withdraw cash with real-time balance update.
* Transfer funds between different accounts within the same bank.

### **Loan Management**

* Customers can apply for a loan and check their loan status with should be made available.
* The system needs to determine loan interest rates, along with schedules for repayment.
* The loan officers are recommended to have authority to review and approve loan application.

### **Customer Support**

* The system should permit users to gain access to customer support service for instance by email or chat.
* For assisting the customers support representative needs to have access to data customer.

### **Account Statements and Reports**

* Accountholders should be informed how to create and download their account statements on the portal.
* Managers ought to be granted sufficient reports for their operational needs, like daily transaction summaries, account balances, and logs of audit.

### **Online Bill Payment**

* Online bill payments are vital such as paying for gas and electric bills, credit card bills and loans.
* The system should be set up to perfectly do those functions as it should process these payments and provide confirmation of the payments.

## **Non-Functional Requirements**

### **Security**

* The banking system should come up with extra secure layers to assure the safety of customer data from theft, hacks, and frauds. This comprises of the use of encryption for the transmission and the storage of the data so that only appropriate persons can read or keep on record the sensitive material [1].
* Impacts of vulnerabilities including the lost data, damage to the company's reputation, and potential legal fines, therefore, secure software systems are extremely important for both large and small organizations.
* Provision of fine-grained access controls is necessary, locking down those components of the system only for sanctioned users. Furthermore, the system must incorporate safeguards against such usual vulnerabilities like SQL Injection [2].

### **Performance**

* The platform should ensure that the turnaround time for common operations like query for account balance and transfer to an account should be very fast to engender good user experience and efficiency [3].
* It should have a very high capacity of supporting the huge number of users and transactions in the peak period when there is no decrease in its performance [4].

### **Scalability**

* We expect to see the banking system adapt endlessly to match a growing number of participants, accounts, and payments as time passes. This is the case where you can include scalability thus more servers or resources to accommodate growth [5].

### **Reliability**

* The system is expected to ensure a high level of availability minimizing failures or downtimes for maintenance or upgrades. The intention behind that is to prevent disruption of banking services which will lead to lower quality of customer experience [6].
* It is to have backup plans to switchover in the occasion of hardware or software failures to preserve non-stop service [7].

### **Backup and Recovery**

* To prevent data loss caused by system failures, robust data backup and recovery strategies should be implemented. The fault tolerance of the backups should be evaluated on a regular basis [8].
* In the event of a critical failure, the banking institution should have a comprehensive disaster recovery plan in place to ensure that banking operations are not hampered in any way [9].

### **Compliance**

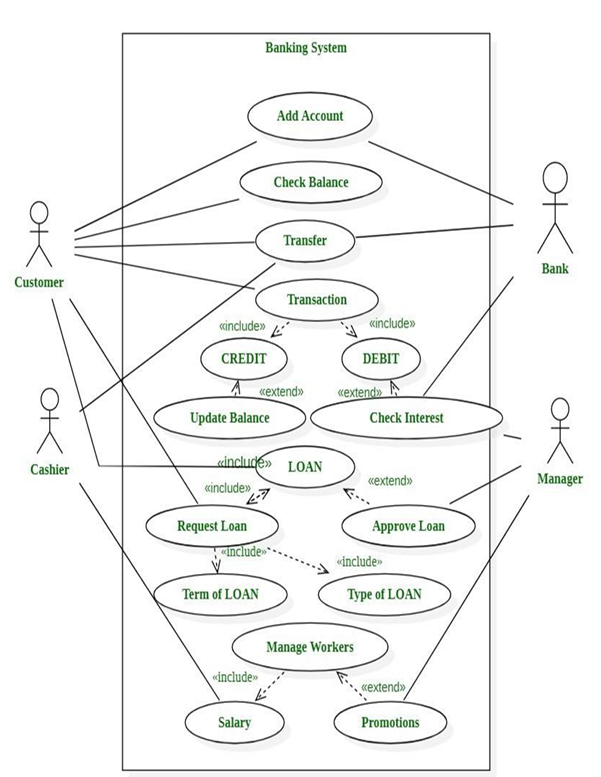
* Such a framework should be consistent with pertinent banking and monetary industry guidelines, including yet not restricted to KYC and AML necessities, for administrative consistence [10].
* The framework should likewise be GDPR and HIPAA consistent to guarantee the security of the client information and protection [11].

### **Usability**

* Banks connection point of the framework ought to be very natural as well as cordial to clients at various specialized capacities to move forward convenience [12].
* Accessibility of openness highlights is expected for the framework to be generally available and agreeable with availability principles like WCAG [13].

## **Design**

### **Use Case Diagram**



### **Class Diagram**

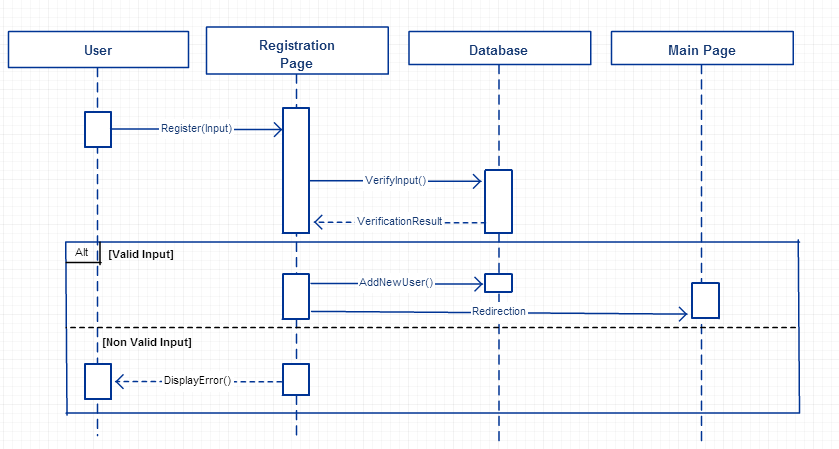
A diagram of a bank account

Description automatically generated

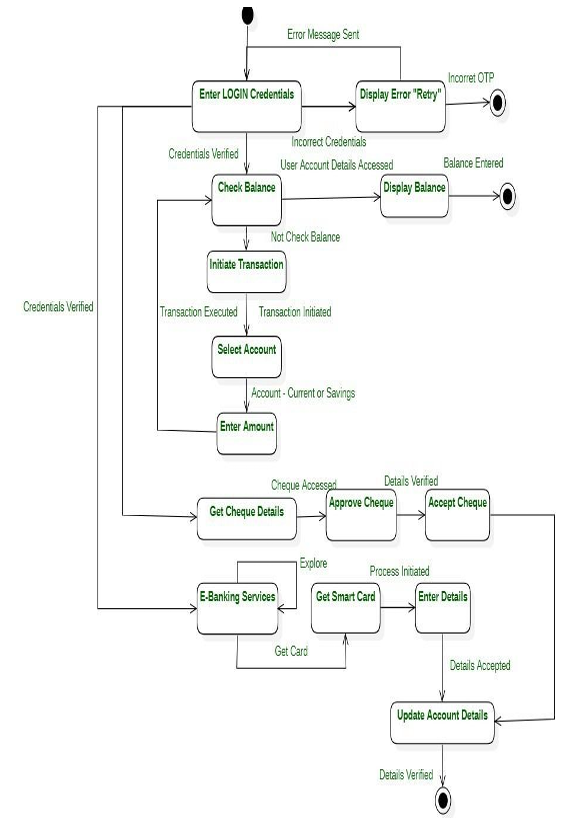
### **Sequence Diagram**

Sequence diagrams illustrate the behaviour of objects in a use case by specifying their messages. It shows a visual picture of item interactions across time. A sequence diagram depicts an actor, as well as the items and components with which they interact throughout the execution of a use case. A sequence diagram depicts a single Use Case ‘scenario' or set of occurrences. Sequence diagrams depict the flow of messages from one object to another, and hence correlate to the methods and events provided by an object.

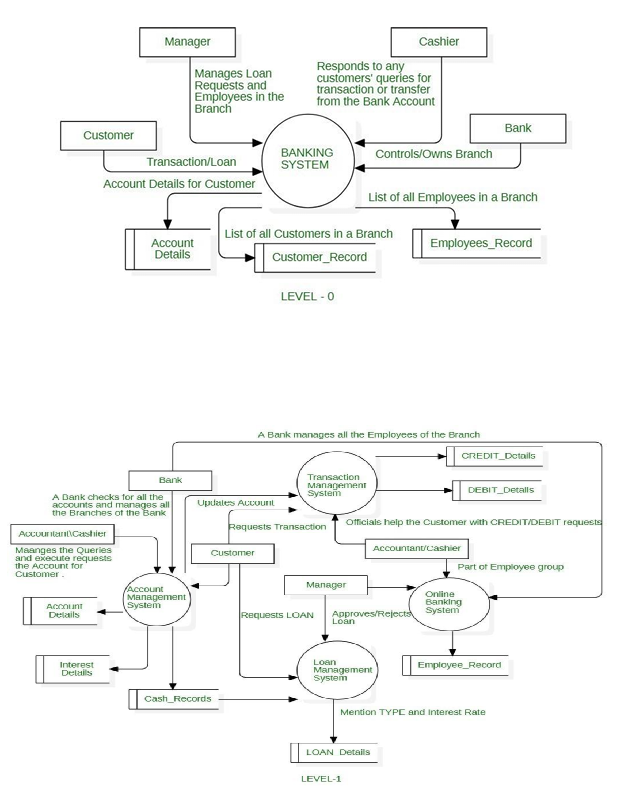
**Login Module**

****

### **State chart Diagram**

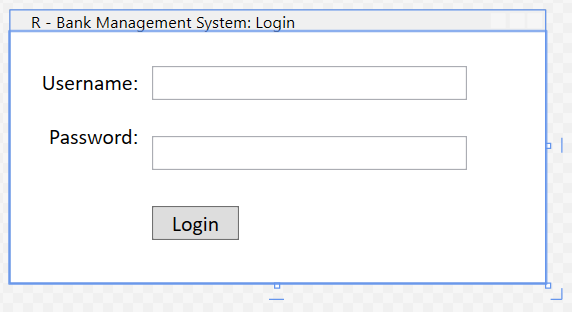


### **DFD0 and DFD1 Level Diagram**

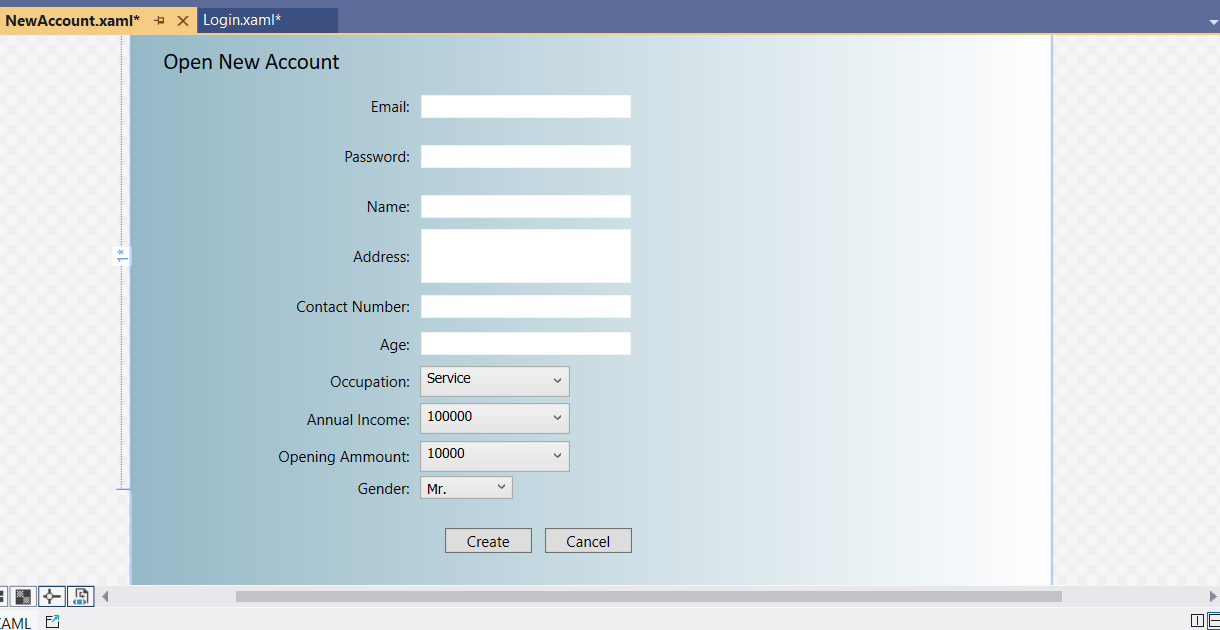


## **Graphical User Interface (GUI) Design**

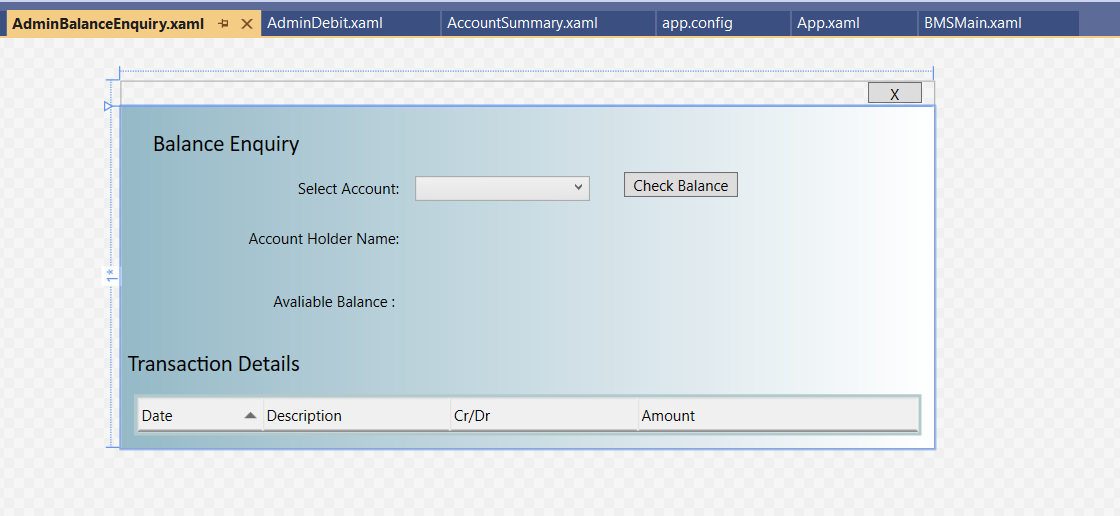
### **Login Module**



### **New Account Module**



### **Balance Enquiry Module**



### **Credit Amount Module**

A screenshot of a login form

Description automatically generated

### **Debit Amount Module**

A screenshot of a computer

Description automatically generated

### **All Transaction Module**

A screenshot of a computer

Description automatically generated

### **Delete Account Module**

A screenshot of a computer

Description automatically generated

### **Cash Deposit Module**

A screenshot of a computer

Description automatically generated

### **Cash Withdraw Module**

A screenshot of a computer

Description automatically generated

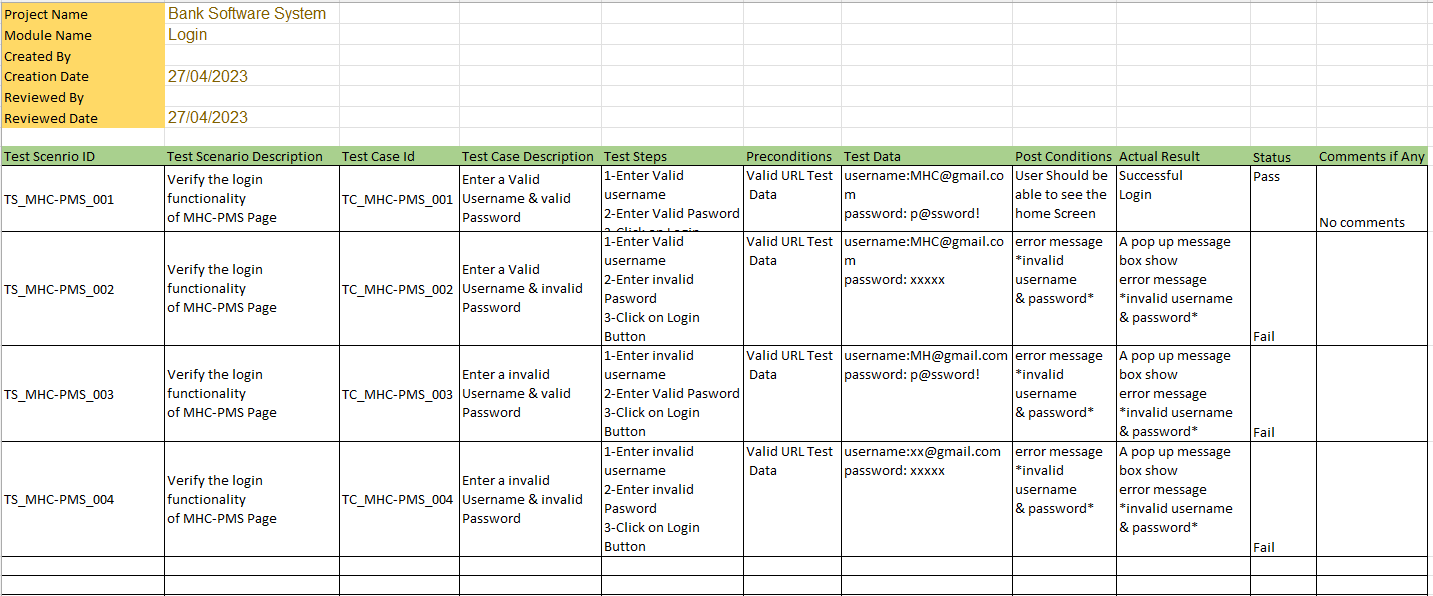
### **Transfer Money Module**

A screenshot of a computer

Description automatically generated

### **Test Cases**

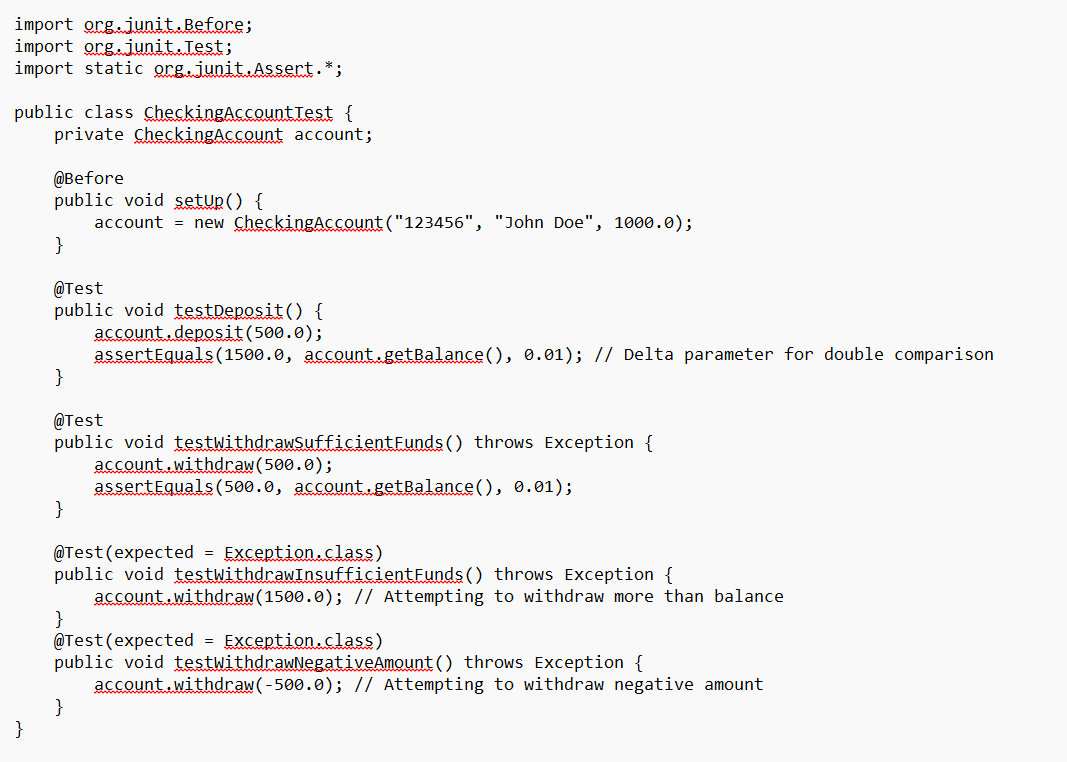
**Login Module Testing**



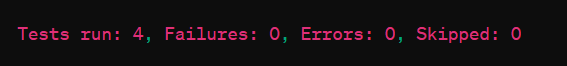
**Checking Account Module Testing**

I implemented a testing framework JUnit to set up a straightforward automated test. I made a test class to confirm the usefulness of the CheckingAccount class I characterized before. The deposit and withdrawal methods will be tested in this test to make sure they work as expected.

**Code:**

****

**Output:**



**Section 5**

**Critical Evaluation of Software Engineering Tools and Techniques and their Impact on Society**

# **Advances in Software Processes**

In our banking system project, we applied the Agile methodology (Scrum specifically) as a way of managing the development process. These kinds of methods are known for their flexibility and adaptation to changing needs. As a result of iterative development cycles, we have been able to prioritize features appropriately and deliver incremental value to all the stakeholders.

## **Agile Processes**

Proof from our project indicates that Agile techniques do help in rendering value to stakeholders. The standups, sprint planning sessions, and retrospectives were key elements of our Agile process. Such methods promoted teamwork, communication and feedback between team participants and other parties. For instance, during sprint reviews, stakeholders would provide feedback on implemented features which eventually would lead to refinement and improvement in the subsequent iterations.

## **Software Engineering Techniques**

### **Test Cases** **Project Management**

Project management tools such as Jira helped us to track the tasks, manage sprints, and monitor the progress. This has enabled us to be organized, prioritize work effectively, and complete tasks in a timely manner. The agile project management principles that consist of transparency, inspection, and adaptation were the fundamentals of our project management strategy. For example, we had sprint planning meetings to set realistic sprint goals and tracked progress against burndown charts.

### **Prototype Design**

Prototype design has been an essential part of the requirements validation and getting feedback concerning the stakeholders.

* **Web Development:** The interface is built using HTML, CSS, JavaScript, and web frameworks, which are supported by server-side languages of ASP.NET.
* **Database Management:** To store data, use databases such as MySQL.
* **Security:** Use encryption, multi-factor authentication, and other security measures.
* **API and Integration:** Create an API to connect with third-party services. Compliance is adhering to financial laws such as KYC and AML.
* **Backup and Recovery:** Automated backup methods to ensure data integrity.
* **Compliance and Maintenance:** Make sure to follow financial requirements, do monthly maintenance, and keep your security up to date.

The Banking Management System needs specified hardware requirements to function properly. To get best system performance, the following hardware requirements should be met:

* For optimal performance, use an Intel Core i5 or similar processor.
* At least 8 GB of RAM is required for seamless functioning with numerous users and huge datasets.
* The system should have at least 500 GB of SSD storage for rapid data retrieval, logging, and backups.
* A reliable internet connection is crucial for accessing web-based components, securely transferring data, and communicating with external financial networks.
* Strong security measures, such as firewalls and antivirus software, are necessary to preserve data integrity and privacy.

### **Version Control**

Version control systems (like Git) were very much needed in the management of codebase changes, in parallel cooperation, and in assurance of code quality. By implementing branching and merging approaches, we could carry out the development of features in parallel, perform code reviews, and keep a clean and stable codebase [14]. Such things as the continuous integration and deployment pipelines were created to automate testing and deployment processes as to make sure that the changes would be integrated in a smooth way.

## **Impact on Society**

The banking system developed by us has a considerable social effect through the provision of services that are both accessible and available. It makes it possible for people to manage their money well, do transactions safely, and even access banking services remotely, in turn, results in financial inclusion and the empowerment of individuals.

### **Ethical Impact**

We always put ethical factors first place, especially data privacy and security. By applying rigorous security procedures to safeguard customers’ confidential information and complying with regulations on data protection, such as GDPR, we also endeavoured to maintain transparency of our data handling processes. Following these ethics helps to gain the trust and confidence of the user, which develops the long-term relationship with customers.

### **Entrepreneurial Impact**

The financial system, which is made up of banking organizations, gives opportunity for creative enterprises to develop their market distinctiveness. Banks would be able to deliver customized financial products and services, improve customer experience, and raise competitiveness by using complex programs and cutting-edge technology. Furthermore, the system's scalability and adaptability allow the bank to adapt to market changes on a regular basis, as well as identify new chances for growth and expansion.

Finally, the use of modern software engineering methods has allowed for the effective construction of the financial system, assuring its performance, dependability, and social involvement. We created a sophisticated and user-centric banking system that meets the needs of stakeholders while also contributing to beneficial social outcomes using an Agile mentality, software engineering approaches, and ethical concerns.

# Github Repo Link:

<https://github.com/CodeWithMaddy709/CMP9134M-Advanced-Software-Engineering>

# 

**References**

[1] "Security Requirements Engineering for Web Applications", Advances in Software Engineering, 2019.

[2] "Enhancing Security of Web Applications Using Model-Driven Engineering", Journal of Systems and Software, 2020.

[3] "Performance Engineering in Agile and DevOps", Advances in Software Engineering, 2018.

[4] "Scalability and Performance Engineering", Journal of Systems and Software, 2017.

[5] "Scalability Patterns in Enterprise Software Systems", Advances in Software Engineering, 2021.

[6] "Reliability Engineering for Distributed Systems", Journal of Systems and Software, 2019.

[7] "Fault Tolerance Techniques for Reliable Software Systems", Advances in Software Engineering, 2020.

[8] "Data Backup and Recovery Strategies in Cloud Computing Environments", Journal of Systems and Software, 2018.

[9] "Disaster Recovery Planning in Banking Systems", Advances in Software Engineering, 2017.

[10] "Regulatory Compliance in Financial Systems", Journal of Systems and Software, 2019.

[11] "Ensuring Data Privacy and Protection in Banking Systems", Advances in Software Engineering, 2021.

[12] "User-Centered Design Principles for Banking Applications", Journal of Systems and Software, 2020.

[13] "Accessibility Design Guidelines for Financial Services", Advances in Software Engineering, 2018.

[14] Chacon, S., & Straub, B. (2014). Pro Git. Apress.