Software Requirement and Design Specifications

Bank Management System

Version: 1.0

Course Code	SE-2002
Instructor	Miss. Rubab Jaffar
Project Team	Muhammad Kazim (K21-3852) Saqib Shaikh (K21-3874) Abdul Rehman Qureshi (K21-3827)
Submission Date	13/3/2023

Table Of Content:

1. Introduction

1.1. Purpose of Document

The purpose of the Software Requirement Specification (SRS) for a Bank Management System is to document the functional and non-functional requirements of the software application. The SRS serves as a comprehensive guide for the development team, providing a clear and concise description of what the software should do and how it should perform. Additionally, the SRS serves as a basis for evaluating the system's compliance with the requirements and serves as a reference for maintenance and future enhancements. Overall, the SRS for a Bank Management System helps to ensure the successful development, implementation, and operation of the software application.

1.2. Intended Audience

Our intended audience for this SRS document is Miss. Rubab Jaffar, instructor of SE-2002 course, Spring 2023 semester.

1.3. Definition of Terms, Acronyms and Abbreviations

Term	Descriptions
BMS	Bank Management System

1.4. Document Convention

We have used font style Times New Roman and font size 14 for heading and 12 for sub-headings.

2. Overall System Description

2.1. Project Background

The BMS will streamline the day-to-day operations of a bank by providing an efficient and user-friendly platform for managing customer accounts, transactions, and staff. The system will integrate various modules such as account opening, deposit and withdrawal processing, and loan processing. The goal is to enhance the speed, accuracy, and security of banking operations while reducing manual errors and increasing customer satisfaction.

2.2. Project Scope

Bank Management System would typically include the essential features required for managing customer accounts, transactions, and staff. The system would facilitate account opening, deposit and withdrawal processing, balance inquiries, transaction recording, and customer relationship management, among other functions. The system would be designed for ease of use, with a userfriendly interface.

Admin can log in, check account-holder details, check recent withdrawals, check recent deposits, check loan requests, check credit card details, and can check cheque book details.

Users can create an account, log in, deposit money, withdraw money, check available balance, make a loan request, check withdrawal history, check their deposit history, make cheque book request, make a credit card request, and can generate withdraw and deposit slip.

2.3. Not In Scope

The system will not include forgotten password features, online payments, investment management, foreign exchange, or wealth management, which are typically offered by larger financial institutions.

2.4. Project Objectives

The objective is to deliver a reliable, scalable, and secure banking software system that can be easily customized to meet the specific needs of different types of banks, thereby improving the efficiency, accuracy, and security of banking operations.

2.5. Stakeholders

- Product Owner
- Project Manager
- Developers
- Designers
- Analyst
- Testers

- Quality Insurance Staff
- Database Administrator
- Architects
- IT Specialist
- Customers
- Bank Staff
- Investors
- Gmail

2.6. Operating Environment

The app will run only on Windows 7 or later and macOS 10 or later.

2.7. System Constraints

The system should have 4 GB RAM.
The system should have a 32GB hard-drive or SSD.
The system should have a anti-virus software.

2.8. Assumptions & Dependencies

- The system is dependent on the other banking system.
- This system is dependent on Gmail.
- It is dependent on the GSM cellular service providers.
- If the user forgot his/her ID password so he/she has to contact the admin.
- The proposed system is to work only with the Windows and macOS operating system. It is assumed that if any other operating system is used the system will fail to launch.

3. External Interface Requirements

3.1. Hardware Interfaces

Our Bank Management System would require a computer with a fast processor, at least 4 GB of RAM, and a reliable hard drive or SSD with ample storage capacity for data and software. The system should have a high-quality display, a keyboard, and a mouse for efficient input and output of information. Additionally, the computer should be equipped with a network interface card for communication with other devices and internet access, and the operating system should be secure and up-to-date with regular updates and patches. To ensure reliability, it is also recommended to have a backup power supply and data backup system in place.

3.2. Software Interfaces

Our Bank Management System would require a comprehensive software package that is capable of managing transactions, customer data, and account information. The software should have robust security features to prevent unauthorized access and ensure data privacy. The software should integrate with other banking systems and adhere to industry standards for data exchange and communication.

3.3. Communication Interfaces

Our Bank Management System requires reliable communication channels to connect with customers, staff, and other banking systems. The system should have secure communication protocols, such as SSL/TLS encryption, to protect sensitive information during transmission. It should support a range of communication channels, including email, and SMS, to provide timely and convenient updates to customers. Additionally, the system should be able to communicate with other banking systems, such as payment gateways and transaction processing systems, to ensure smooth and efficient processing of transactions. To minimize communication downtime, the system should have redundancy measures in place, such as backup communication channels and failover systems.

4. Functional Requirements

4.1. Functional Hierarchy

Account Management:

- The system should allow customers to create accounts, and view their account balances.
- The system should allow bank employees to view and manage customer accounts, including deposits, withdrawals, and transfers.
- The system should allow customers to request for new cheque books and credit cards, and provide an interface for bank employees to process these requests.

Loan Management:

- The system should allow customers to apply for loans.
- The system should allow bank employees to review loan applications, approve or reject them.

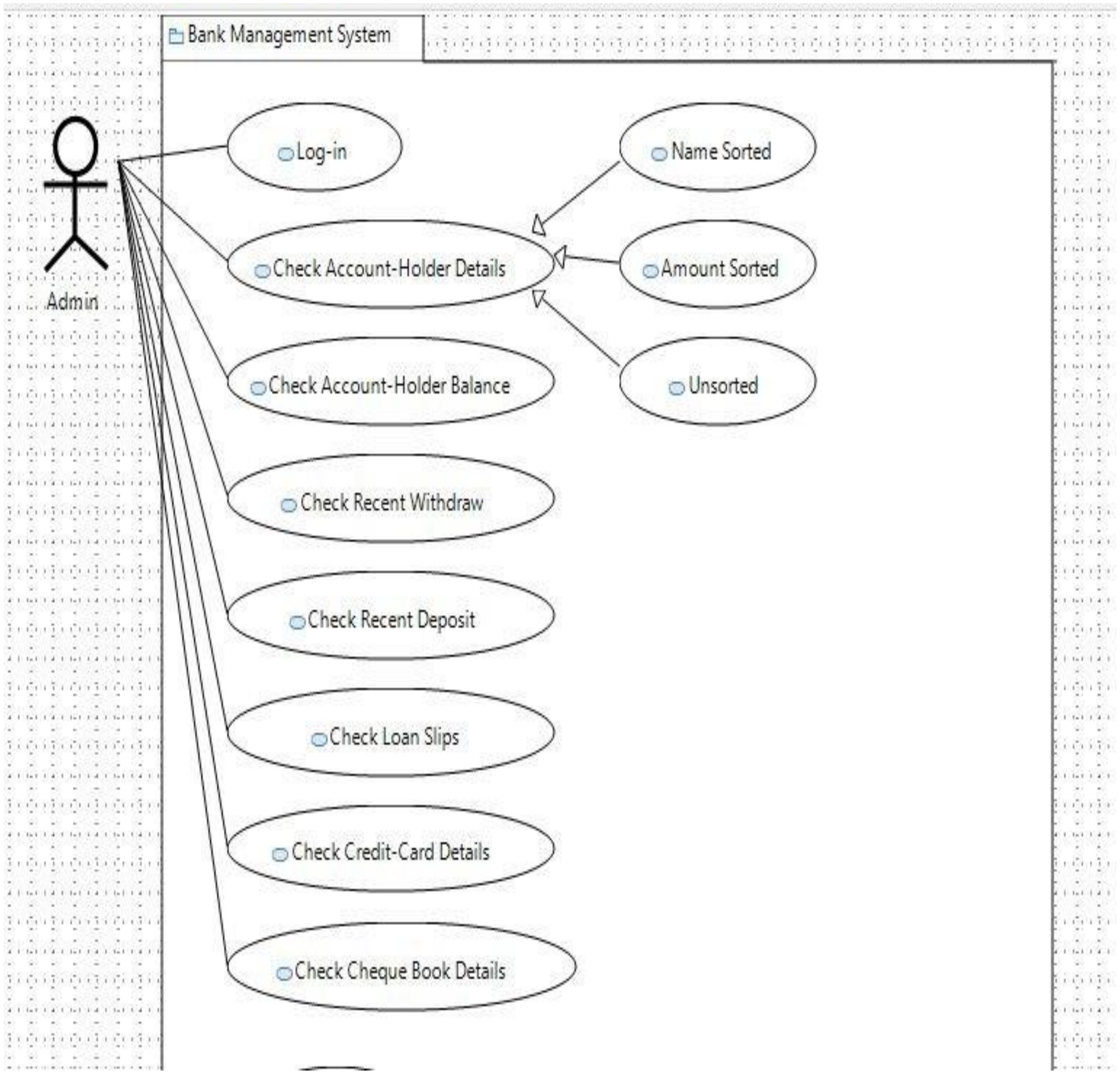
Transaction Processing:

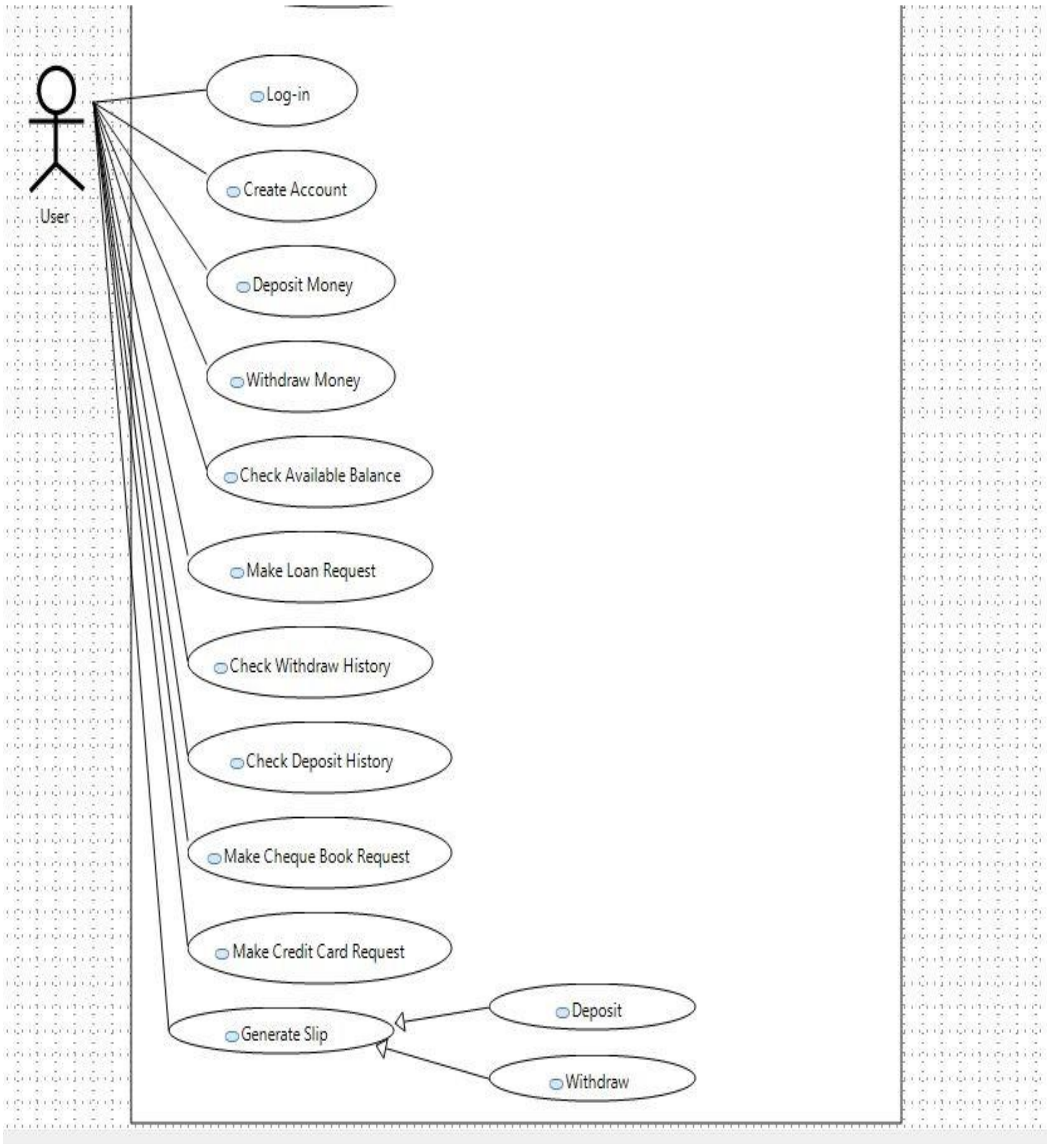
- The system should allow customers to initiate transactions, such as deposits and withdrawals, and generate transaction slips for these transactions.

Fraud Detection and Prevention:

- The system should include fraud detection and prevention mechanisms to identify and prevent fraudulent activities, such as identity theft, account takeover, and money laundering.
- The system should use machine learning and artificial intelligence algorithms to analyze transaction patterns, detect anomalies, and flag suspicious activities.

4.2. Use Cases


Bank Management System
Admin
Log-in
Check Account-Holder Details
Check Account-Holder Balance
Check Recent Withdraw
Check Recent Deposit
Check Loan Slips
Check Credit-Card Details
Check Cheque Book Details
Name Sorted
Amount Sorted
Unsorted


User
Log-in
Create Account
Deposit Money
Withdraw Money
Check Available Balance
Make Loan Request
Check Withdraw History
Check Deposit History
Make Cheque Book Request
Make Credit Card Request
Generate Slip
Deposit
Withdraw

4.2.1. Creating Bank Account

Use-Case Name:	Creating Bank Account
Use-Case Description:	The use case describes the event of a user creating his/her bank account through the system. Once the account is created successfully and updated in the database, a confirmation is shown and an email is sent to the user.
Primary Actor:	Customer
Other Actors:	Gmail Database
Stakeholders:	• Customer • Bank • Gmail • Database
Relationships:	
Preconditions:	The customer should have a compatible device with internet connectivity, valid identification documents, and sufficient funds.
Flow of Events:	1. The customer will run the system. 2. The customer will go to the create account page and enters all his/her details. 3. Customer will submit the details and provides the initial deposit amount. 4. System will verify the customer's details and the funds are transferred from the customer's account to the new bank account. 5. The system will prompt that the account has been created and provides the customer with their account information.
Alternative and Exceptional Flow	**Alt-3:** If the customer enters any wrong information, then he/she will be notified of the discrepancy and prompted to fill in all the fields or invalid information. **Alt-4:** During verification any invalid document is found. Ask customer to re-submit the document.
Postconditions:	The account details are recorded in the database if all the procedures were done successfully.

4.2.2. Take Loan

Use-Case Name:	Take Loan

Use-Case Description:	The use case describes the event of an account holder taking loan from bank through the system. Once the loan process is completed system will provide the user with the loan slip.
Primary Actor:	Account-Holder
Other Actors:	Gmail Database
Stakeholders:	• Bank • Accountant • Bank Manager • Account-Holder • Gmail • Database
Relationships:	
Preconditions:	The user should have an account in the bank.
Flow of Events:	1. The user will log-in in his/her account. 2. The system will validate the user account number and password and will give access to the account page. 3. The user will go to the loan page and click on apply loan. 4. The user will fill the loan form and submit it. 5. The system will send the loan slip through email and direct the user to visit the bank with the loan slip.
Alternative and Exceptional Flow	**Alt-2:** If the user has not provided the correct ID or Password. The user is notified of the discrepancy and prompted to re-enter (ID, Password). **Alt-4:** If the user enters any wrong information, then he/she will be notified of the discrepancy and prompted to fill in all the fields or invalid information.
Postconditions:	The loan request is recorded in the database.

4.2.3. Deposit Money

Use-Case Name:	Deposit Money

Use-Case Description:	The use case describes the event of an account holder depositing money in his/her bank account.
Primary Actor:	Account-Holder
Other Actors:	• Gmail • Database
Stakeholders:	• Account-Holder • Bank • Gmail • Database
Relationships:	
Preconditions:	The user should have an account in the bank.
Flow of Events:	1. The user will log-in in his/her account. 2. The system will validate the user account number and password and will give access to the account page. 3. The user will go the deposit page. 4. The user will enter his/her account number and the money to deposit. 5. The user will click on deposit button. 6. The system will verify details and update his/her balance. 7. An email is sent to the user about the money deposit and current balance.
Alternative and Exceptional Flow	**Alt-2:** If the user has not provided the correct ID or Password. The user is notified of the discrepancy and prompted to re-enter (ID, Password). **Alt-4:** If the user enters any wrong information, then he/she will be notified of the discrepancy and prompted to fill in all the fields or invalid information. **Alt-6:** If the details are not verified then, ask user to re-enter the details.
Postconditions:	The account balance is updated in the database.

4.2.4. Withdraw Money

Use-Case Name:	Withdraw Money
Use-Case Description:	The use case describes the event of an account holder withdrawing money from his/her bank account.
Primary Actor:	Account-Holder
Other Actors:	• Gmail • Database
Stakeholders:	• Account-Holder • Bank • Gmail • Database
Relationships:	
Preconditions:	The user should have an account in the bank.
Flow of Events:	The user will log-in in his/her account. The system will validate the user account number and password and will give access to the account page. The user will go the withdraw page. The user will enter his/her account number and the money to withdraw. The user will click on withdraw button. The system will verify details and update his/her balance. An email is sent to the user about the money withdraw and current balance.

Alternative and Exceptional Flow	**Alt-2:** If the user has not provided the correct ID or Password. The user is notified of the discrepancy and prompted to re-enter (ID, Password). **Alt-4:** If the user enters any wrong information, then he/she will be notified of the discrepancy and prompted to fill in all the fields or invalid information. **Alt-5:** If the user enter amount greater than his/her balance then, user is notified of the discrepancy and prompted to re-enter amount. **Alt-6:** If the details are not verified then, ask user to re-enter the details.
Postconditions:	The account balance is updated in the database.

4.2.5. Check Account Details

Use-Case Name:	Check Account Details
Use-Case Description:	The use case describes the event of the admin checking account holder's details.
Primary Actor:	Admin
Other Actors:	Database
Stakeholders:	• Bank • Database • Admin
Relationships:	Generalization
Preconditions:	The admin should have the ID and password in order to check the account holder's details.
Flow of Events:	The admin will log in to his/her account. The system will validate the user account number and password and will give access to the admin page. The admin will go to the account details page. The admin will enter the account holder name. The system will show the account holder details.

Alternative and Exceptional Flow	**Alt-2:** If the admin has not provided the correct ID or Password. The admin is notified of the discrepancy and prompted to re-enter (ID, Password).
Postconditions:	The system will show the details in the text area.

4.2.6. Checking Loan Slips

Use-Case Name:	Checking Loan Slips
Use-Case	The use case describes the event of the admin checking loan slip and
Description:	verifying them.
Primary Actor:	Admin
Other Actors:	Database
Stakeholders:	• Bank • Database • Admin
Relationships:	
Preconditions:	The admin should have the ID and password in order to check the loan slips.
Flow of Events:	The admin will log in to his/her account. The system will validate the user account number and password and will give access to the admin page. The admin will go to the loan page. The system will automatically show all loan applications.
Alternative and Exceptional Flow	**Alt-2:** If the admin has not provided the correct ID or Password. The admin is notified of the discrepancy and prompted to re-enter (ID, Password).
Postconditions:	The system will show the loan applications in the text area.

5. Non-Functional Requirements

5.1. Performance Requirements

The system must be designed to perform well under heavy loads, handle a large volume of transactions, and respond quickly to user requests.

5.2. Safety Requirements

The system must be designed to provide reliable backup and recovery mechanisms to prevent data loss, security monitoring and auditing tools to detect and respond to security incidents, a comprehensive disaster recovery plan to minimize disruption in the event of a disaster, secure communication protocols to protect sensitive information during transmission, and staff training on security policies and procedures to ensure proper system use.

5.3. Security Requirements

The system must be designed with strict security measures to prevent unauthorized access, data breaches, and cyber-attacks. It should be able to provide encryption of sensitive data, multi-factor authentication, and role-based access controls.

5.4. User Documentation

Tutorials will be added in the system in order to help new users to understand the system.

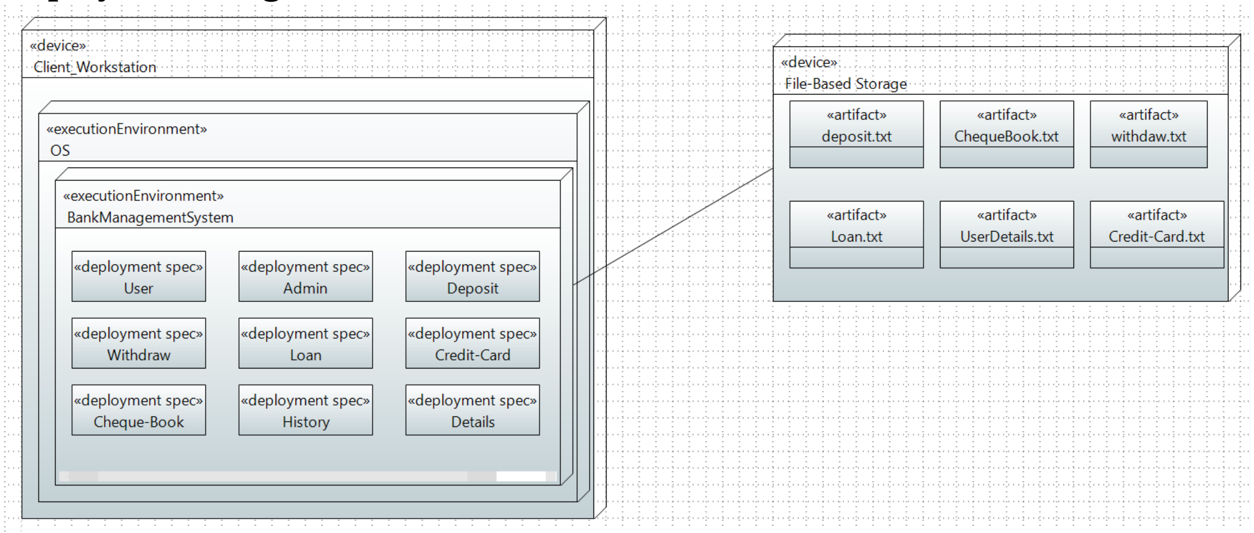
SDS

6. System Architecture

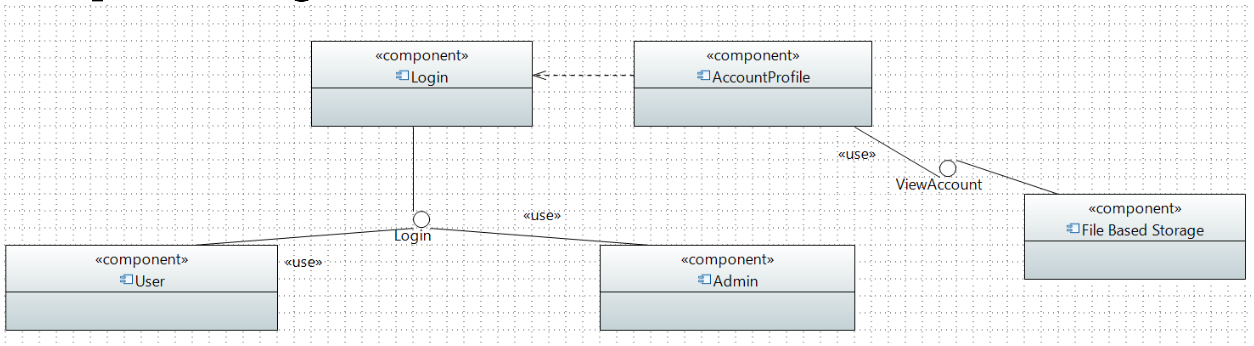
The overview of the system: There are two sections User and Admin and most of the functionality is working separately but the loan, cheque-book or credit-card request is working between the user and admin, aa s user request for the loan, cheque-book or credit card, the request goes to the admin.

6.1. System Level Architecture

Deployment Diagram:



Component Diagram:



6.2. Software Architecture

- User Interface Layer:

This layer provides a graphical user interface (GUI) for customers and bank employees to interact with the system. It includes desktop application that allow users to view their account information, make transactions, and access various banking services.

- File-Based Storage Layer:

This layer stores and manages the bank's data, including customer information, transaction data, account balances, and other financial data. The layer uses file-based storage instead of a database, which can include JSON or other file formats.

- Middleware Layer:

This layer provides connectivity between the application layer and the file-based storage layer. It includes messaging queues, caching mechanisms, and other middleware components that enable communication and data exchange between the various components of the system.

- Security Layer:

This layer provides security measures to protect the system from unauthorized access and ensure data privacy. It includes firewalls, encryption mechanisms, access control policies, and other security components that protect the system from external threats.

- Infrastructure Layer:

This layer provides the underlying infrastructure components such as servers, storage devices, networking devices, and other hardware components that support the operation of the bank management system

7. Design Strategy

7.1. **User-centered design approach**: The bank management system should be designed with the end-users in mind. The system should be user-friendly and intuitive, making it easy for bank employees and customers to use. This requires a deep understanding of user needs, habits, and preferences.

7.2. **Scalability:** The bank management system should be designed to scale as the bank grows. It should be able to handle a large number of users, transactions, and data. This requires a robust architecture and a scalable infrastructure.

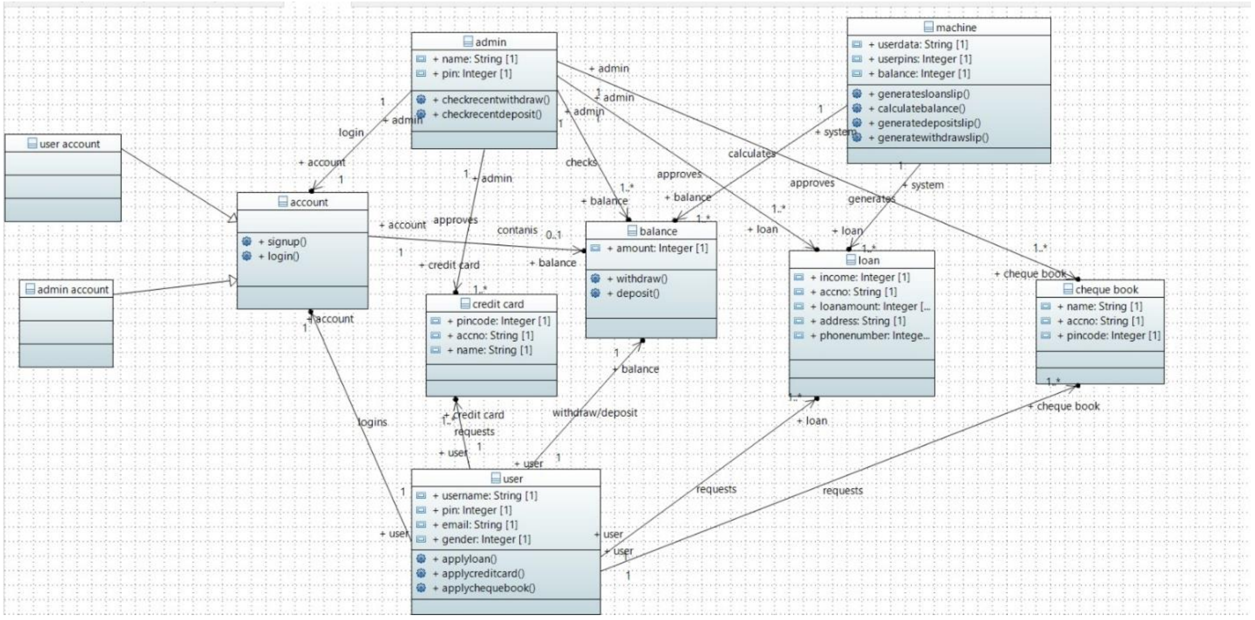
7.3. **Security:** Security is of utmost importance in a bank management system. The system should be designed with multiple layers of security to ensure that customer data and transactions are protected. This requires a thorough understanding of security protocols, encryption techniques, and compliance requirements.

7.4. **Flexibility:** The bank management system should be designed to accommodate changes in business requirements, regulations, and technology. This requires a modular design that can be easily updated and expanded.

7.5. **Performance:** The bank management system should be designed to deliver high performance, with minimal latency and downtime. This requires a robust infrastructure and optimized code.

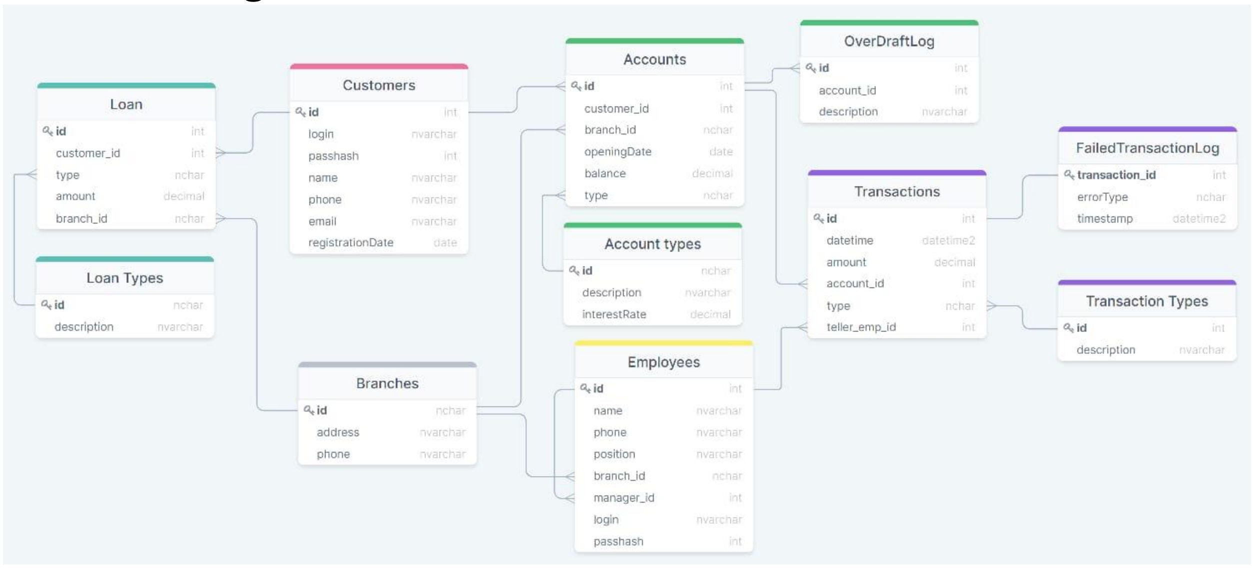
7.6. **Analytics:** The bank management system should be designed to capture and analyze data from various sources, such as customer transactions. This requires a strong data management strategy and advanced analytics tools.

8. Detailed System Design



8.1. Database Design

8.1.1. ER Diagram



8.1.2. Data Dictionary

8.1.2.1 Data 1

<DATA 1>	
Name	User
Alias	customer
Where used/how used	It is used when logging In or signing up and when applying for loan, credit card, cheque book inorder to enter relevant details

Content description	Stores information about user, username, pin email and gender.					
Colunm name	Description	Type	Length	Null able	Default value	Key type
username	User login identity	String	7	no	n/a	FK
pin	Unique security number	int	4	no	n/a	PK
email	Source of contact	String	15	yes	n/a	-
gender	sex	String	n/a	no	male	-
name	User identity	String	n/a	no	n/a	-

8.1.2.2 Data 2

<DATA 2>						
Name	Account					
Alias	-					
Where used/how used	It is used to keep track of users account and stores information about account. Used while deposit and withdrawl.					
Content description	Stores account number, balance and recent withdrawals and deposit.					
Colunm name	Description	Type	Length	Null able	Default value	Key type
Account no	Unique account identification	String	11	no	n/a	PK
balance	Amount of money in account	long	n/a	no	0	-

deposit	Recently made deposit	long	n/a	yes	0	-
withdraw	Recently made withdrawl	long	n/a	yes	0	-

8.1.2.3 Data 3

<DATA 3>						
Name	Loan					
Alias	credit					
Where used/how used	Used when a user applies for loan. Getting filled information from user.					
Content description	Takes input income, account number, loan amount, address & phone number of user applying for loan.					
Colunm name	Description	Type	Length	Nullable	Default value	Key type
income	Salary of applying person	int	n/a	no	n/a	-
accno	Account number of peson applying	String	7	no	n/a	FK
loanamount	Amount the person need as loan	long	10	no	n/a	PK
address	House number, street number,area , city	String	50	no	n/a	-
phoneno	Phone number to contact person applying	long	11	no	n/a	-

8.1.2.4 Data 4

<DATA 4>						
Name	Cheque book					
Alias	-					
Where used/how used	Used when a user applies for loan. Getting filled information from user.					
Content description	Takes input name, account number, pin.					
Colunm name	Description	Type	Length	Null able	Default value	Key type
Name	User identitiy	String	20	no	n/a	-
accno	Unique account identification	String	7	no	n/a	FK
pin	Unique security number	int	4	no	n/a	-

8.1.2.5 Data 5

<DATA 5>	
Name	Credit card
Alias	-
Where used/how used	Used when a user applies for loan. Getting filled information from user.
Content description	Takes input name, account number, pin.

Colunm name	Description	Type	Length	Null able	Default value	Key type
Name	User identitiy	String	20	no	n/a	-
accno	Unique account identification	String	7	no	n/a	FK
pin	Unique security number	int	4	no	n/a	-

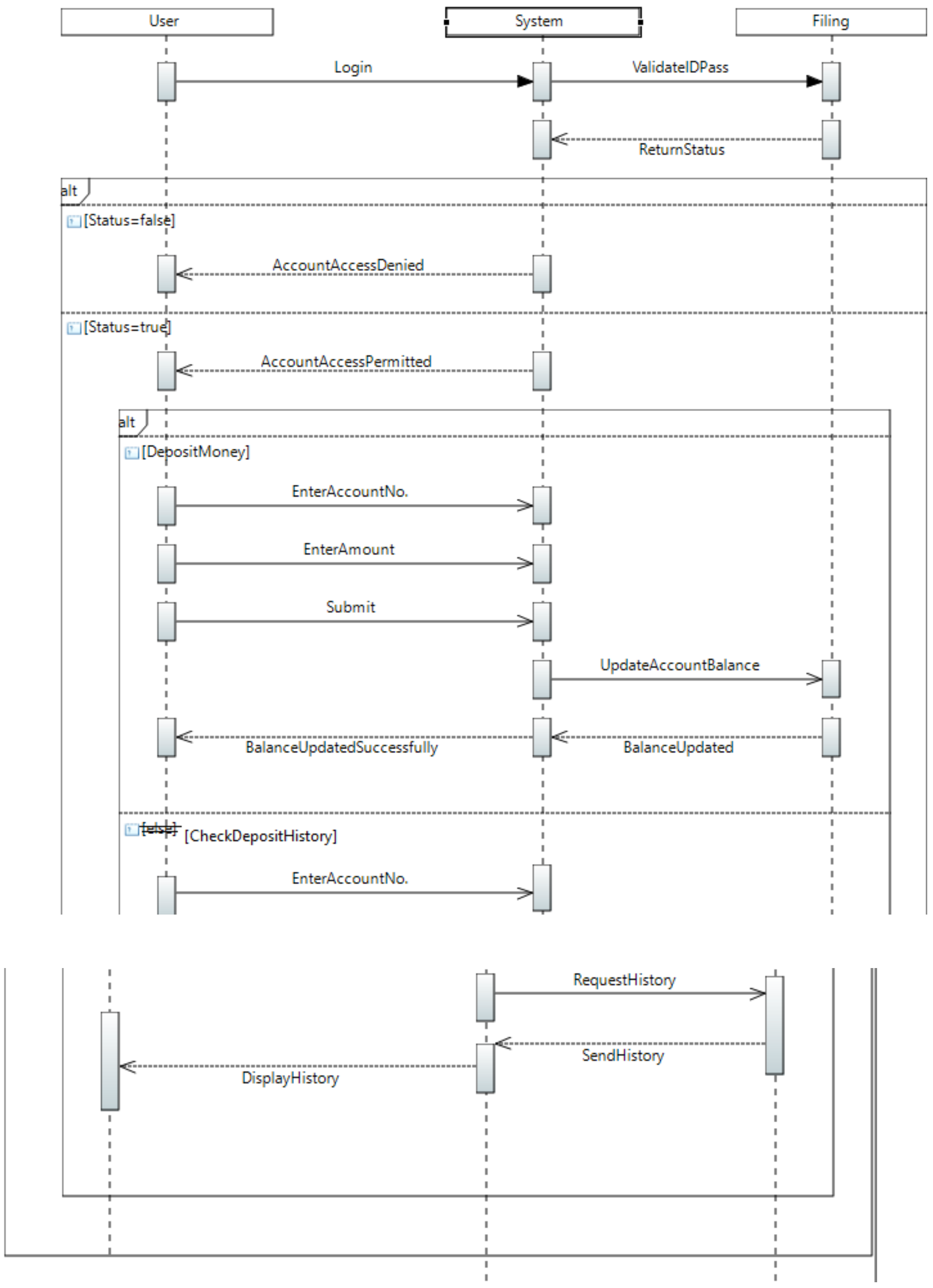
8.1.2.6 Data 6

<DATA 6>						
Name	Admin					
Alias	Incharge					
Where used/how used	Used while logging in for admin. Manages loan, cheque book , credit car applications & check recent withdrawals & deposit.					
Content description	Stores the pin and name for admin.					
Colunm name	Description	Type	Length	Null able	Default value	Key type
Name	User identitiy	String	20	no	n/a	-
pin	Unique security number	int	4	no	n/a	-

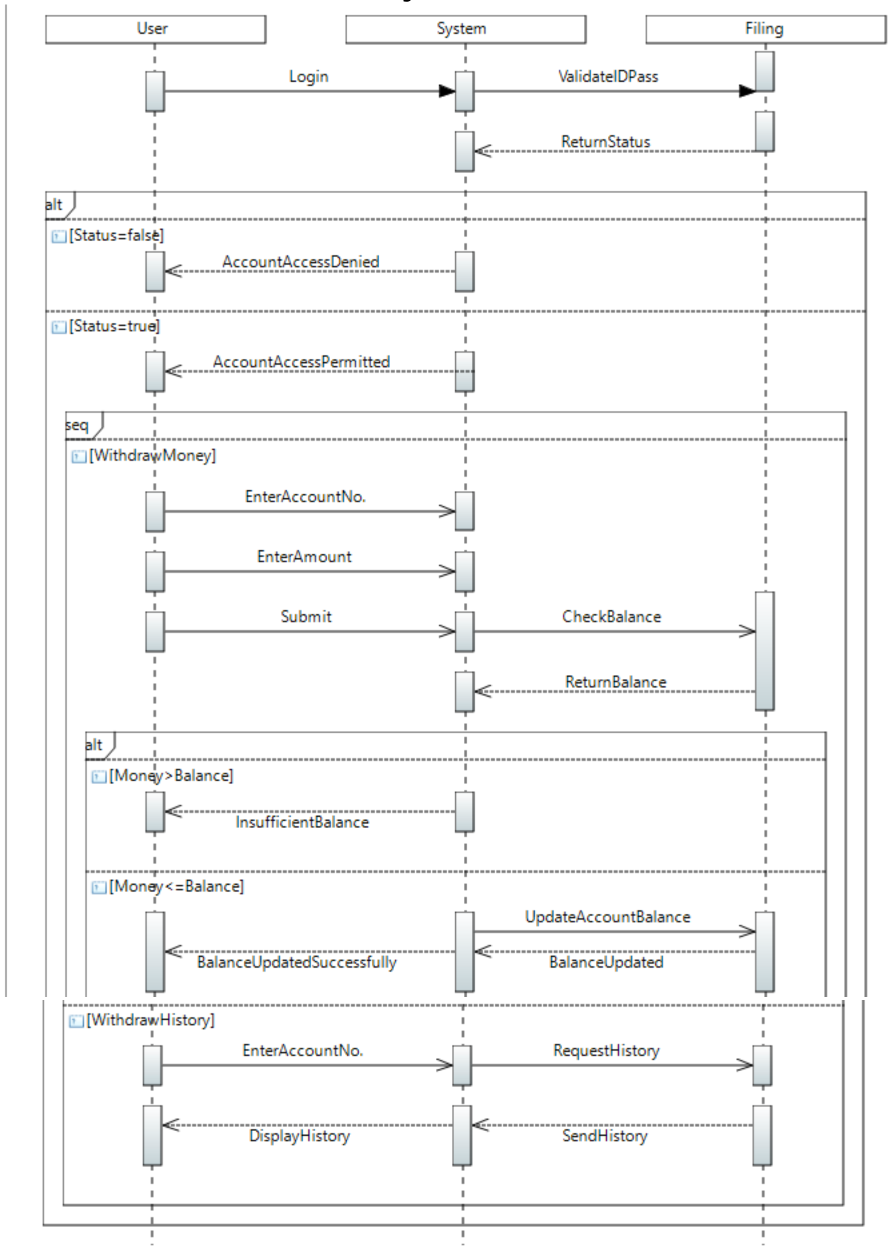
9. Application Design

9.1. Sequence Diagram

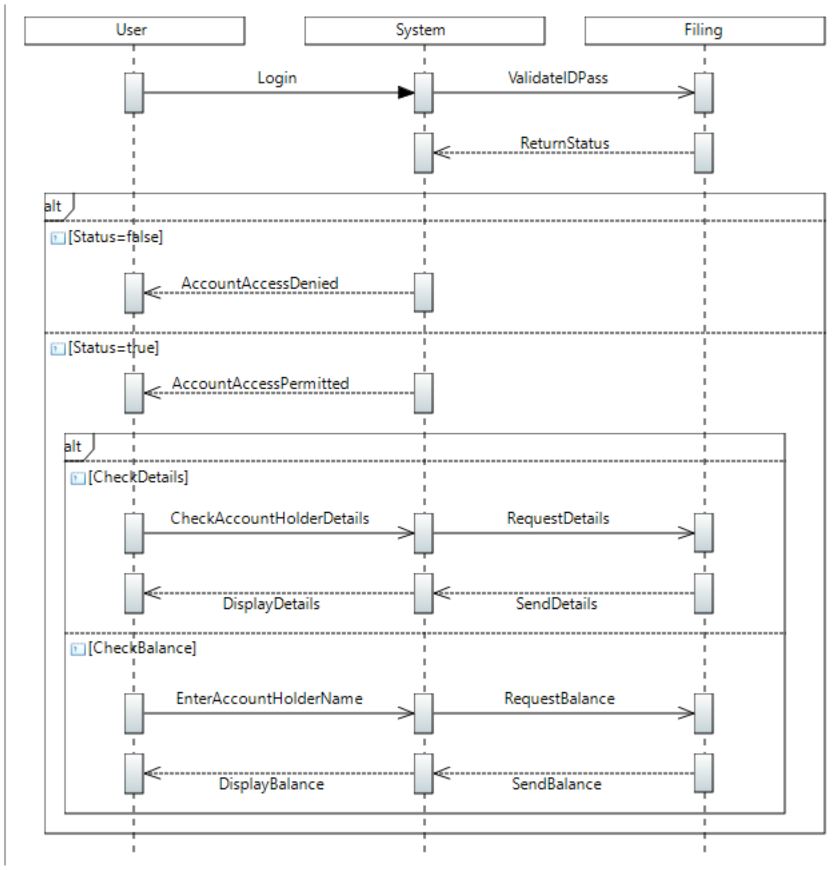
9.1.1 Deposit Money & Check Deposit History



9.1.2 Withdraw Money & Check Withdraw History

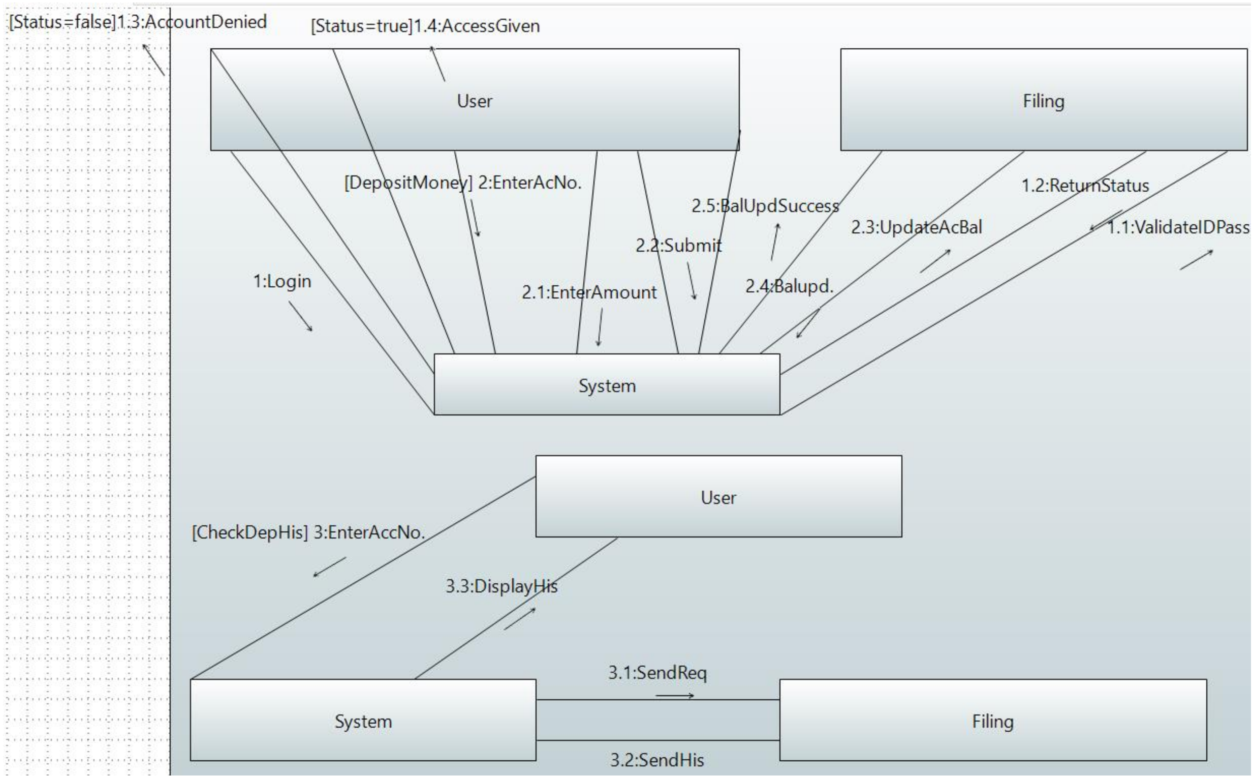


9.1.3 Check Details & Balance

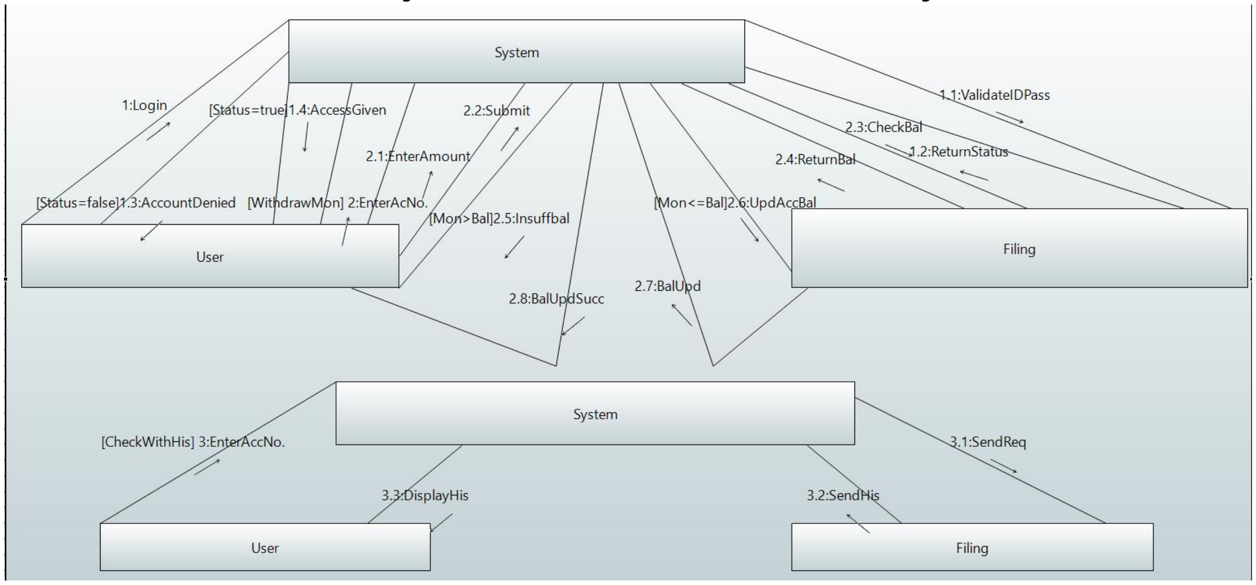


9.2 Collaboration Diagram

9.2.1 Deposit Money & Check Deposit History

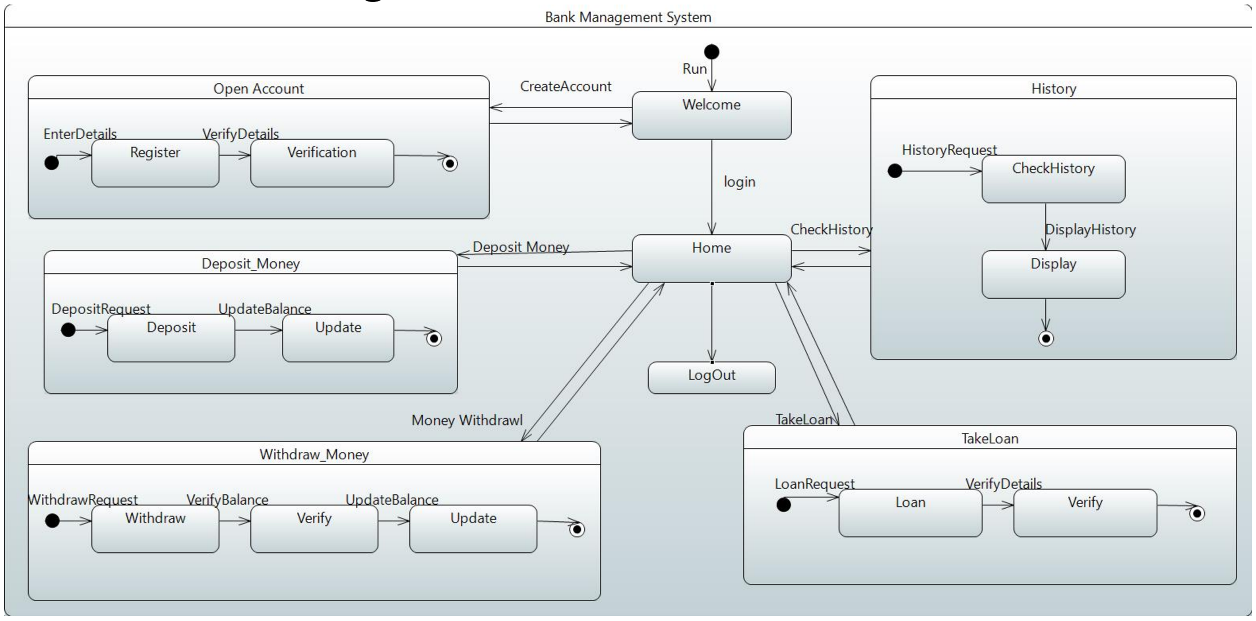


9.2.2 Withdraw Money & Check Withdraw History

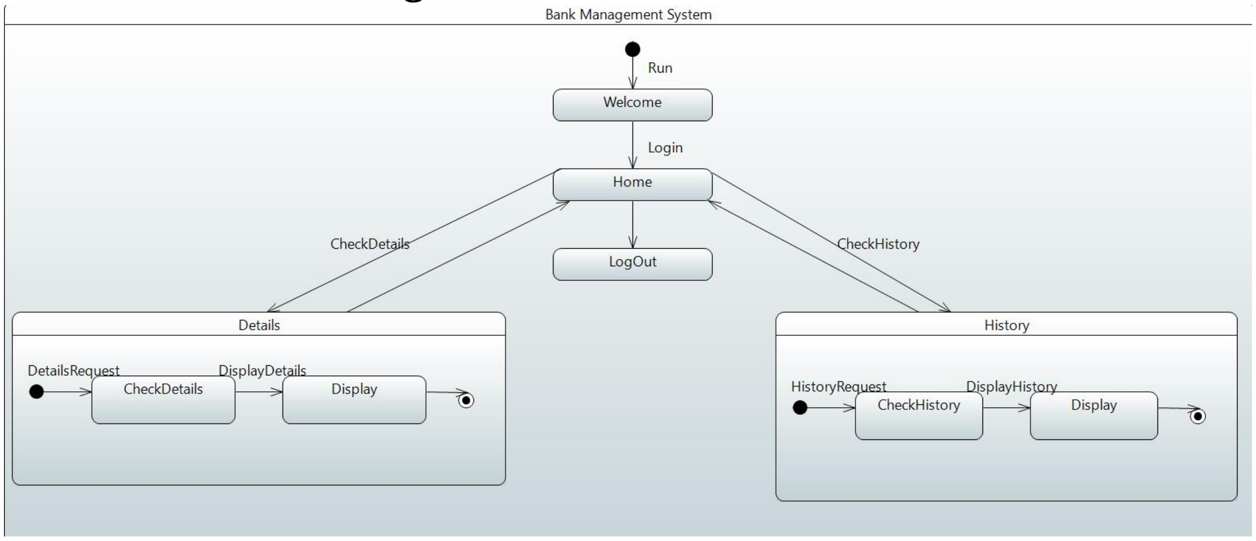


9.3 State Diagram

9.2.1 User State Diagram

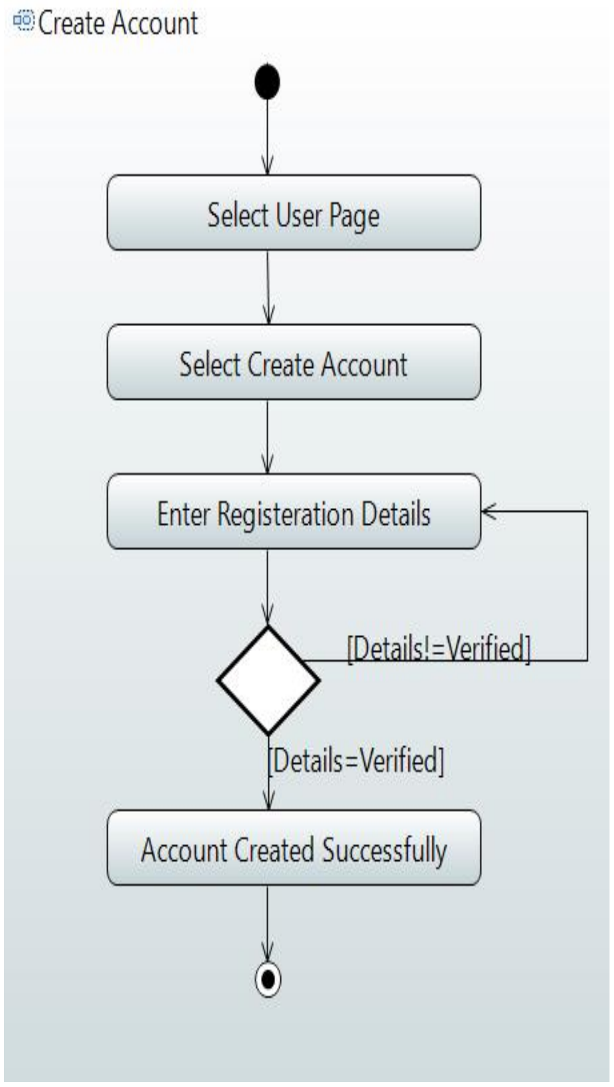


9.2.2 Admin State Diagram

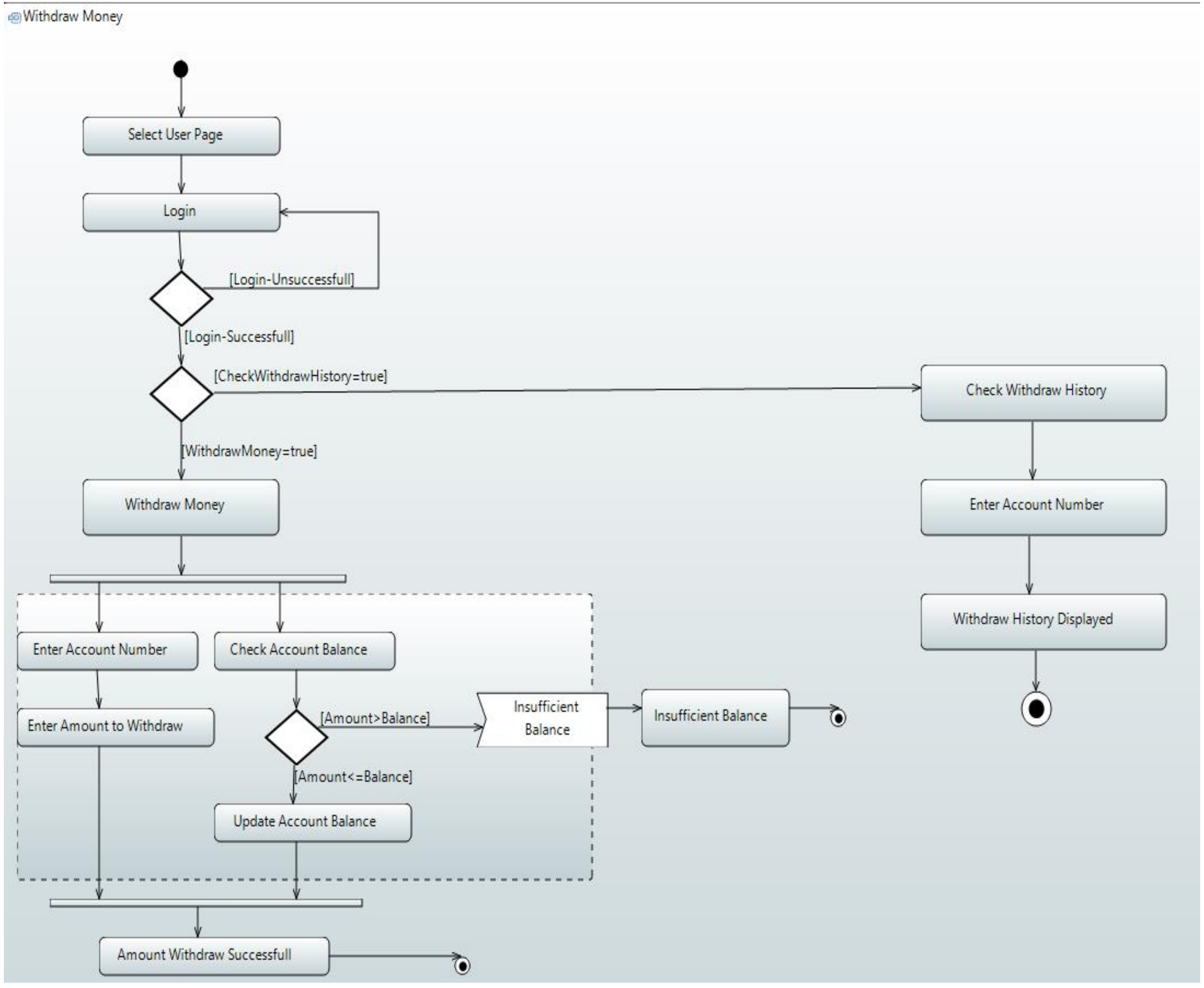


9.4 Activity Diagram

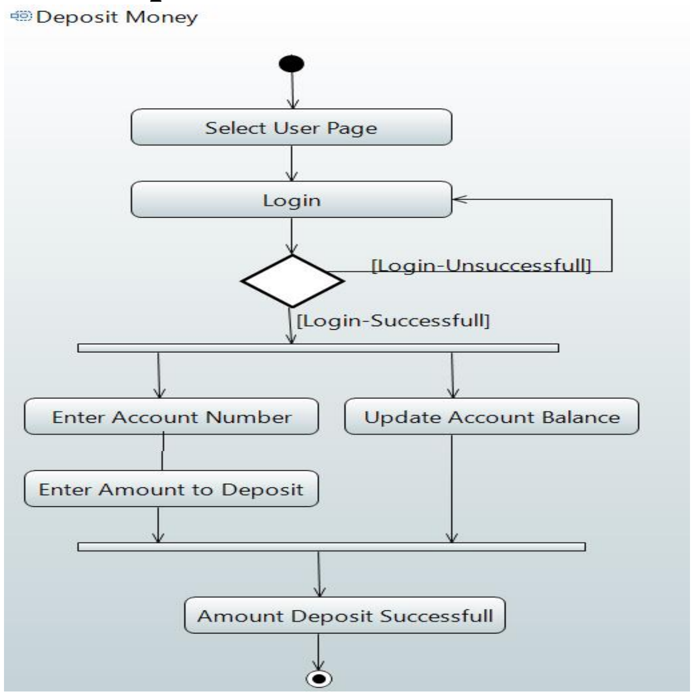
9.4.1 Create Account



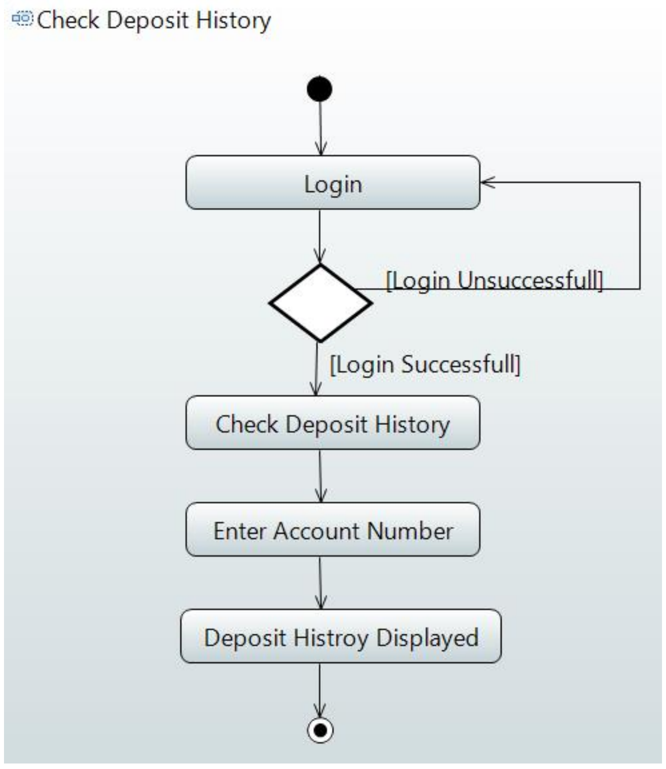
9.4.2 Withdraw Amount and History



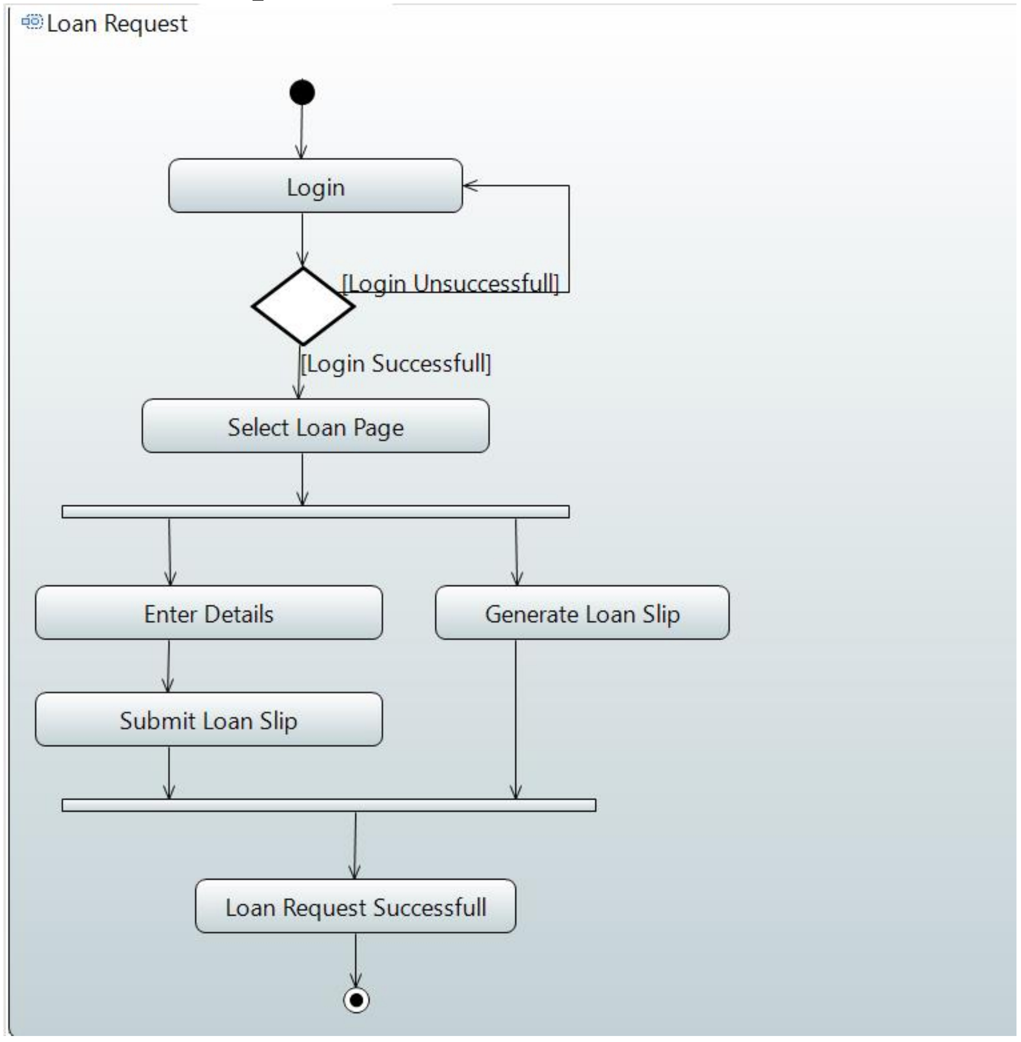
9.4.3 Deposit Amount



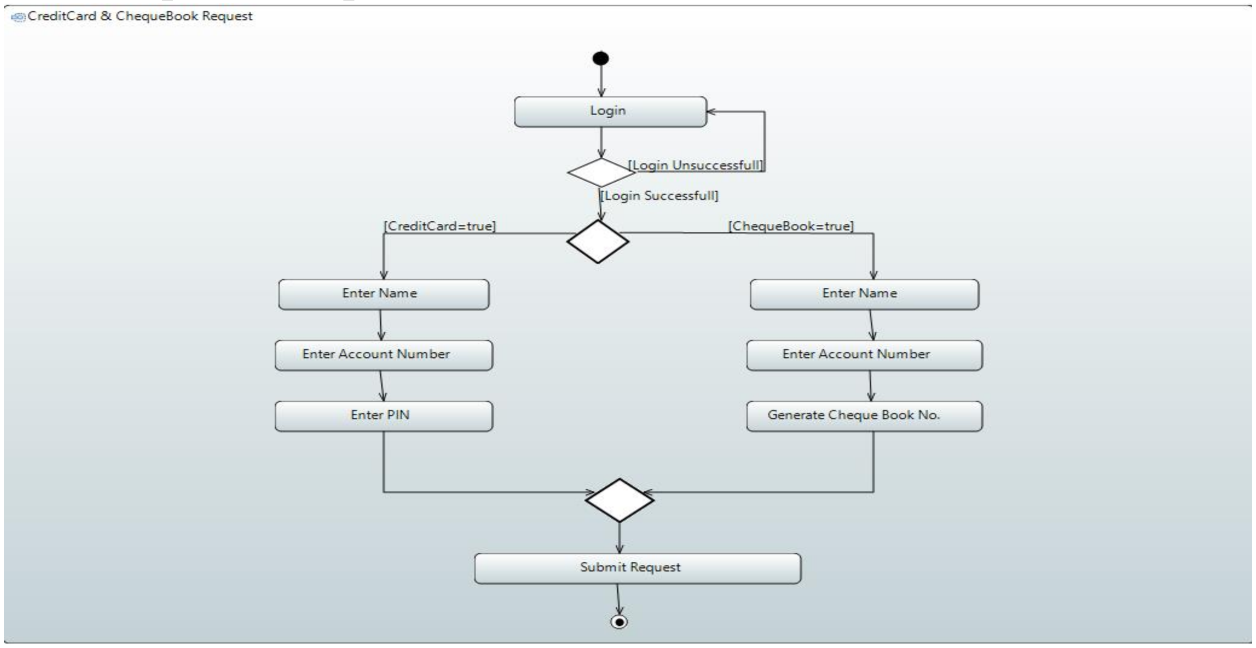
9.4.4 Deposit History



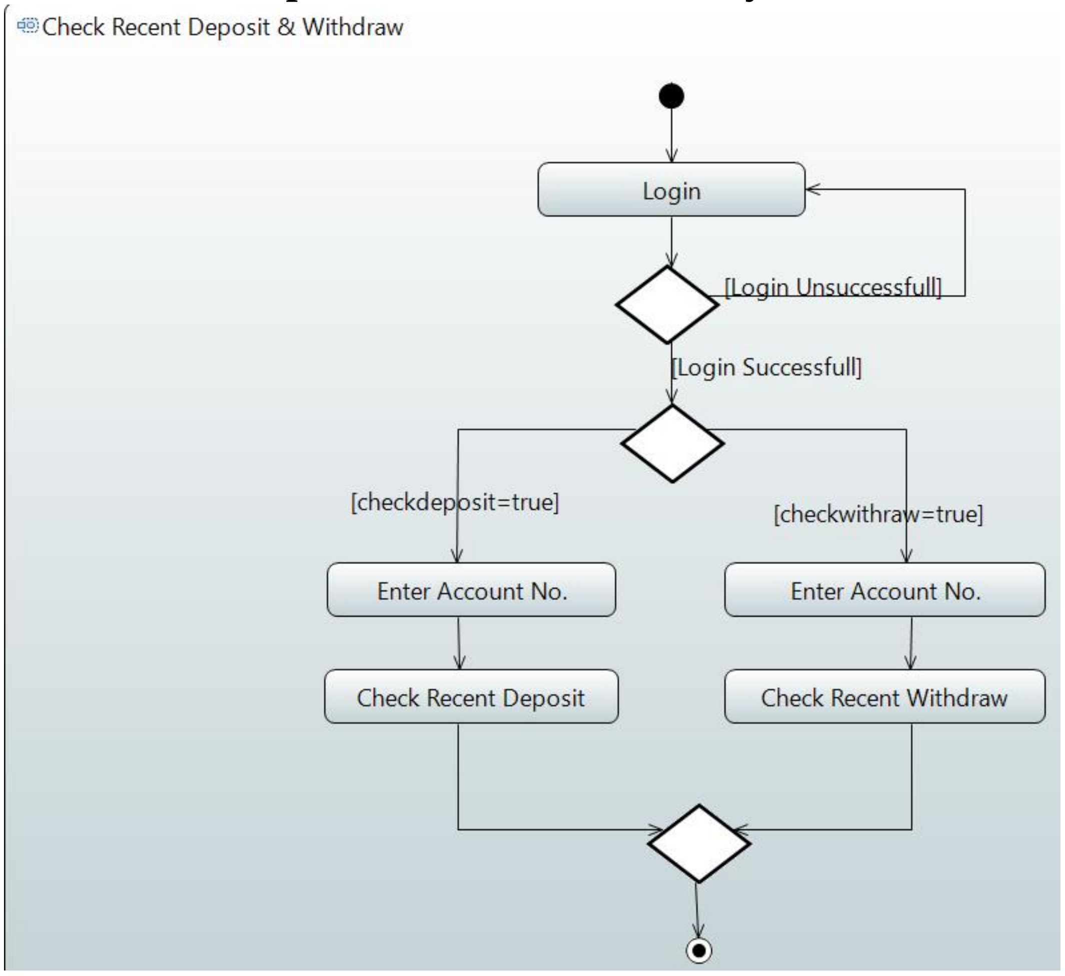
9.4.5 Loan Slip



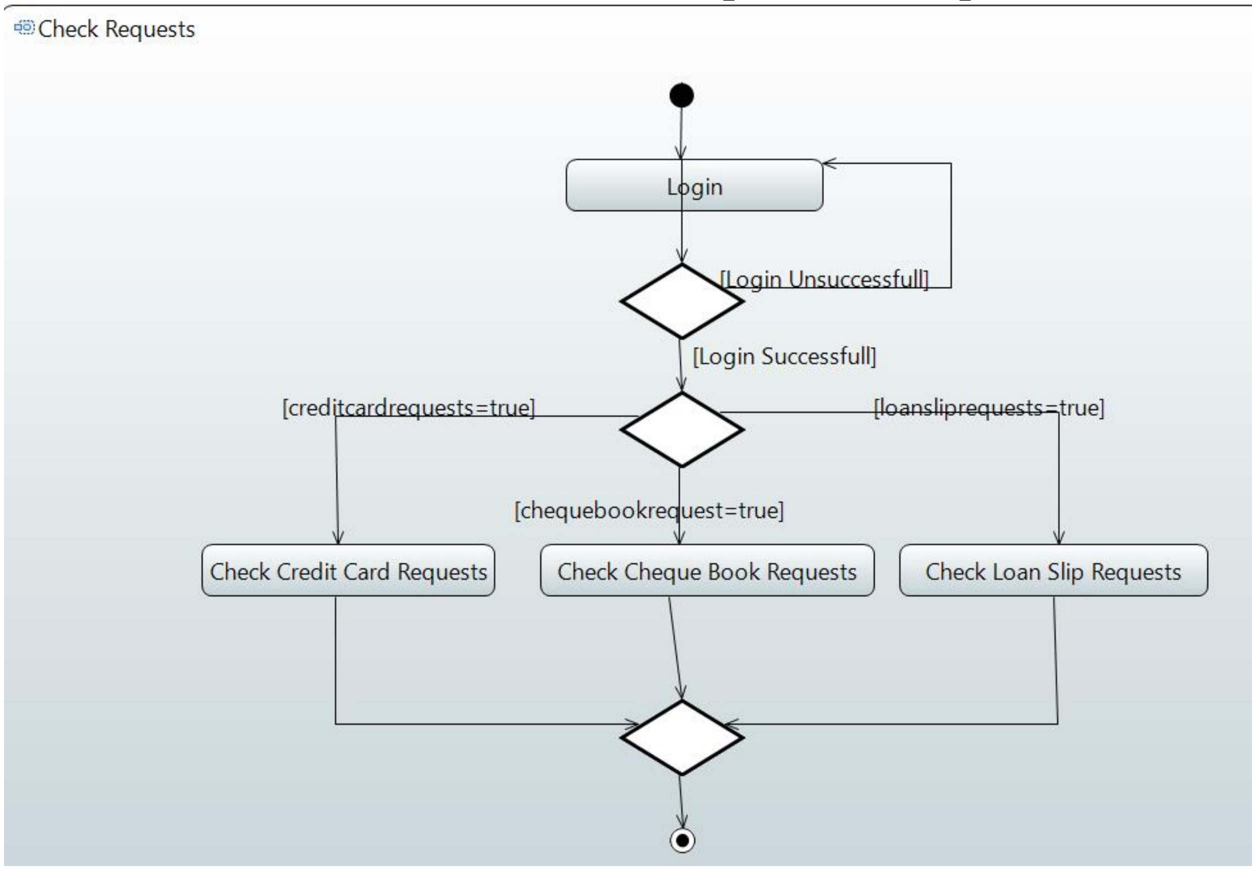
9.4.6 Request Cheques Book & Credit Card



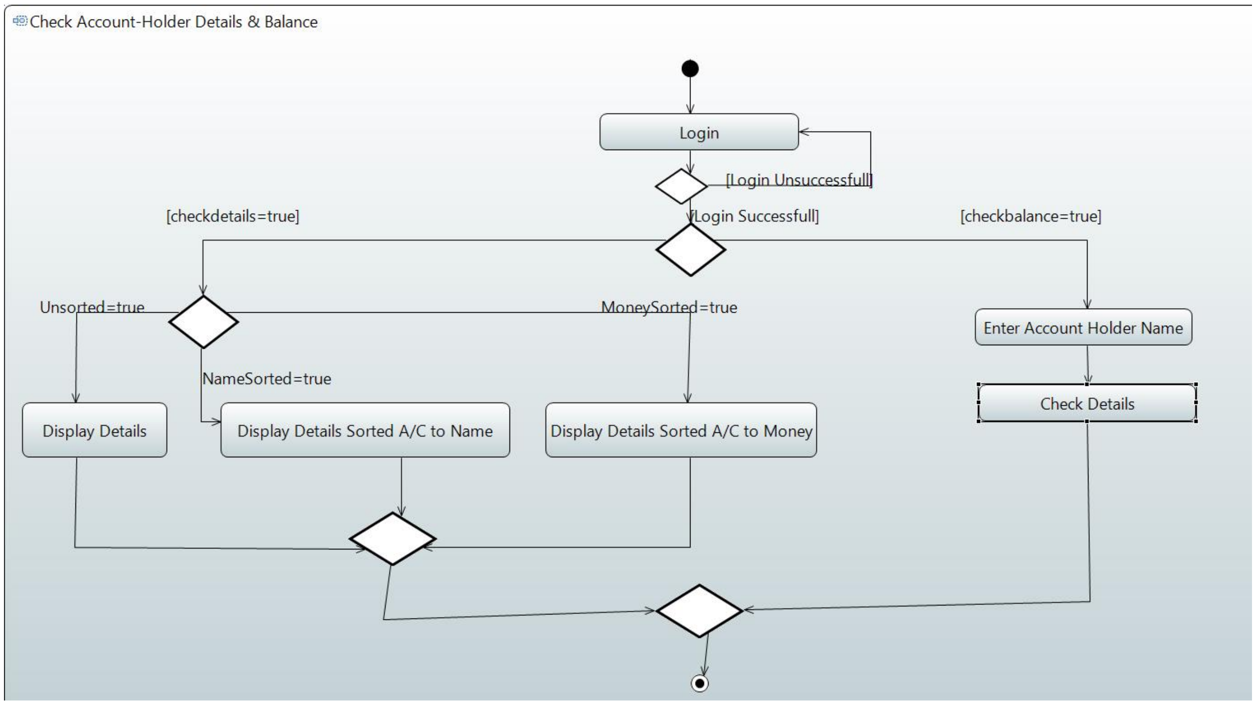
9.4.7 Recent Deposit & Withdraw History



9.4.8 Check Credit-Card, Loan, & Cheque-Book Request



9.4.9 Check Account-Holder Details & Balance



10. References

Not Applicable

11. Appendices

Not Applicable