Object Oriented Software Engineering
OOSE
SUBJECT CODE: SE 202a
PROJECT FILE

TOPIC: BANKING SYSTEM

Submitted by:

NAMAN SINGH (2K21/SE/124)

MICHAEL MULUGETA DEMISSIE(2K21/SE/112)

Batch – SE A2 (2nd year)

Submitted to:

Dr. Ruchika Malhotra

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
SHAHBAD DAULATPUR
(FORMERLY DELHI COLLEGE OF ENGINEERING)
MAIN BAWANA ROAD, Delhi 110042

INDEX

PROBLEM STATEMENT

Banking is an industry that handles cash, credit, and other financial transactions for individual consumers and businesses alike. Banking provides the liquidity needed for families and businesses to invest in the future, and is one of the key drivers of the U.S. economy. A bank is a financial institution licensed to receive deposits and make loans. Banks may also provide financial services, such as wealth management, currency exchange and safe deposit boxes. The conventional system doesn’t involve the use of special institutes for money management. The money is present in the hands of all the people individually. People manage their money on their own, money here strictly refers to cash. People store their money in their homes or their wallets and it is used in case of transactions. The conventional system has the following limitations:

- Storing of money locally in homes is not safe.
- Our money becomes more vulnerable to thievery.
- Money stored as cash might get lost in case of accidents like fire, floods and other natural disasters.
- Since no interest is given, our money doesn’t grow with time.
- There is no official provision to apply for loans/borrowing money.
- Exchange of currency poses a lot of problems in terms of distance and time.
- Transactions take a very large time as compared to those done through banks (netbanking system).
- The task of storing huge amount of money becomes very difficult and cumbersome.

So, to overcome the above problems we are proposing a solution which has the following features:

- To provide a safe and secure platform to the customers to store their hard earned money.
- The system manages the money of all the account holders efficiently, thereby reducing the stress and burden on their heads.
- The system also provides additional interests on our amount, which means that our deposited amount keeps increasing slowly with time.
- Additional services like loan applications and credit card management are also provided.
- Services like netbanking makes the transactions much easier.

Initial Requirements Document (IRD)

Title of the project	Banking System
Stakeholders involved in capturing requirements	users (account holders), employees, shareholders, bank manager
Techniques used for requirement capturing	interviewing and brainstorming
Name of the people along with designations	-
Date	January,2023
Version	1.0

Consolidated list of initial requirements:

1. A system is to be implemented to provide a safe and secure platform to the users to store their money.
2. The system should manage all the operations related to money of all the account holders efficiently.
3. All the people who are above 18 years of age should be allowed to create a new account.
4. The user must be allowed to check the existing account balance at any time.
5. The system should enable the account holder to deposit money in the bank account via cash or cheque according to his/her convenience.
6. The user may also demand for withdrawal of money from the account in the form of cash from the bank or the nearest ATM according to his/her convenience.
7. The system should provide additional interests on the amount to benefit the user.
8. The user should be able to apply for loans from the bank at a fixed rate of interest.
9. The system should issue debit and credit cards to the user.
10. The user must be given a cheque book and a passbook which should be updated regularly.
11. System should also provide additional convenience services like netbanking.
12. The user should be able to view previous transaction history, which is sorted based on the date on which the transaction was done.
13. On each successful transaction the system should inform the user via email and text message on the registered mobile number.
14. There must be a provision to report issues, fraud and malicious activity available to the users.
15. The user might also want to close the account.

SOFTWARE REQUIREMENTS SPECIFICATION(SRS) DOCUMENT

1. Introduction

1.1Purpose

The purpose of this document is to specify the requirements and specifications for a banking system. The banking system will be used to manage customer accounts, deposits, withdrawals, transfers, loans and other related transactions. The customer may also report issues in the system

1.2 Scope

The scope of the banking system includes the following functionalities:

- User registration and authentication
- Account creation and management
- Deposit and withdrawal transactions
- Effective loan management
- Fund transfer between accounts
- Transaction history and statement generation
- Security features such as encryption and access control

1.3 Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification
- UI: User Interface
- API: Application Programming Interface
- ATM - Automated Teller Machine
- KYC - Know Your Customer
- PIN - Personal Identification Number
- NFC - Near Field Communication

1.4 Reference

This document is based on the requirements and specifications gathered from various sources, including customer interviews, industry standards, and best practices for banking systems.

1.5 Overview

The banking system is a software application designed to provide users with online access to their bank accounts and enable them to perform various financial operations.

- The system must be able to handle a large number of users and transactions simultaneously while ensuring data privacy and security.
- The system should allow users to create and manage their accounts, deposit and withdraw funds, transfer funds between accounts, view transaction history, apply for loans and generate statements.

- The system should also provide security features such as encryption, access control, and two-factor authentication.
- The non-functional requirements for the system include performance, reliability, usability, and security. The system should be easy to use, reliable, and secure while providing high performance and availability.
- The system must also be compatible with different types of payment gateways, banks and provide APIs for integrating with third-party applications.
- Documentation and training materials should be provided for users and administrators

2. Overall Description

2.1 Product Perspective

- The banking system will be a standalone web application that can be accessed by authorized users through a web browser.
- The system will interact with a database to store and retrieve data related to customer accounts and transactions.
- The system will also integrate with third-party payment gateways to facilitate fund transfers between accounts.

2.2 Product Features

The following are the key features of the banking system:

- User registration and authentication: Users can create an account with the banking system and login using their credentials.
- Account creation and management: Users can create and manage multiple accounts with the banking system, including savings, checking, and other types of accounts.
- Loan management: Users can apply for loans from the bank and the bank after reviewing the transaction history and credit worthiness may approve the request.
- Deposit and withdrawal transactions: Users can deposit or withdraw funds from their accounts using various methods such as cash, check, or online transfer.
- Fund transfer between accounts: Users can transfer funds between their own accounts or to other accounts within the same bank or different banks.
- Transaction history and statement generation: Users can view their transaction history and generate statements for a specified period.
- Security features: The banking system will have security features such as encryption, two-factor authentication, and access control to prevent unauthorized access to user accounts.

2.3 User Classes and Characteristics

The banking system will cater to the following user classes:

- Customers: Customers are the primary users of the banking system. They can create and manage accounts, perform transactions, and view their transaction history.
- Bank employees: Bank employees will have access to administrative features of the system, such as managing customer accounts, generating reports, and performing other maintenance tasks.

Other classes include:

- Account: The bank stores the account details of all the customers.
- Card: The details of cards like card number, CVV and validity are stored in the system.
- Book: Dates of cheques, transaction amount, cheque numbers are also stored in the system.

2.4 Operating Environment

The banking system will be deployed on a web server with the following specifications:

- Operating system: Linux
- Web server: Apache
- Database: MySQL
- Programming language: C++

2.5 Design and Implementation Constraints

The banking system must comply with the following design and implementation constraints:

- The system must be scalable to handle a large number of users and transactions.
- The system must be secure to prevent unauthorized access and data breaches.
- The system must be easy to use and navigate for both customers and bank employees.
- The system must be compatible with various web browsers and devices.

3. Specific Requirements

3.1 Functional Requirements

USE CASE DESCRIPTIONS

1) Register

1	**INTRODUCTION:** This use case documents the steps that must be followed by any new user in order to register and file an application for a new bank account.
2	**ACTORS:** user
3	**PRECONDITION:** None
4	**POSTCONDITION:** An application number and a reference number is generated and emailed to the user.
5	**FLOW OF EVENTS:** **BASIC FLOW:** 1. The user visits the home page of the system's website. 2. Then he/she registers for a new account from the home page. 3. The user is redirected in to a new page where he/she would be required to fill a form containing his/her personal details like name, address, annual income etc. 4. After filling the form, the user then clicks the submit button which is placed at the bottom of the form. 5. Upon successful submission, a message indicating success will be displayed and also sent via email along with an application number and reference number of the user. 6. The user may view the status of application also. **ALTERNATIVE FLOW 1: Incorrect application** If the user is below 18 years of age, his/her application is rejected, a message explaining the cause is displayed and he/she is redirected to the system's website. In case of any incorrect/inconsistent information, the application is rejected, a message explaining the cause is displayed and he/she is redirected to the system's website to fill the form again.
6	**SPECIAL REQUIREMENTS:** none
7	**ASSOCIATED USE CASES:** none

2) Manage account

1	**INTRODUCTION:** This use case enables the actor to create a new account, view account details and delete an existing account.
2	**ACTORS:** user, employee
3	**PRECONDITION:** In case of new account creation, the user must register for the same from the system's website and the employee must be logged in. To view/delete an account the user must already have a functional account in the system and the employee must be logged in.
4	**POSTCONDITION:** Create a new account: Upon successful completion of this use case, a new account will be created and the user will receive a confirmation via email. Additionally, the necessary database entries will be created and stored. View account: The user/employee can successfully view the details of the account. Delete account: The account is permanently removed and database is updated.
5	**FLOW OF EVENTS:** 1)Create a new account: **BASIC FLOW:** 1. The application submitted by the user is received. 2. The employee, utilizing the system, will validate the accuracy of the information provided by the user, determining their eligibility in the process. 3. A new account and its associated database records will be established. 4. A notification indicating successful account creation will be emailed to the user. 5. The user is given an account number, CRN number and is asked to set the account settings, passwords and pins. **ALTERNATIVE FLOW 1: Incorrect registration** In the event that the user does not provide all of the required information in the form, their application will be declined. The user will receive a message indicating the reason for the rejection and is asked to fill the form again to correct any errors and resubmit their application with the necessary information. In case of any incorrect/inconsistent information, the application is rejected, a message explaining the cause is displayed to the user and he/she is asked to fill the registration form again. 2)View or update account: **BASIC FLOW:** 1. The user/employee logs into the system. 2. He/she requests the system to view the account details. 3. All the account details including personal information, account balance, bank statements and card information is shown to the actor. 4. The user may also change the account details if desired.

	ALTERNATIVE FLOW 1: Session expired If the actor is logged into the system and has not done any action in the last 5 minutes, his/her session expires and is required to login again to view the account. 3)Delete account: **BASIC FLOW:** 1. The user logs into the system. 2. The user navigates to the account settings section. 3. The user initiates the request for permanent closure of their bank account by clicking on the "Delete My Account" option. 1. The user gives a valid reason why the account is no longer needed. 2. The user is informed about the procedure to return his/her money. 4. The request is reviewed by the employee. 5. Before deletion, the user will be required to comply with the established conditions and procedures. 6. Upon completion, the account will be deleted and the user will receive a notification via email. 7. The system database is updated. **ALTERNATIVE FLOW 1: The user doesn’t comply with conditions** In this case, the deletion will not be completed and an error message indicating that the deletion was unsuccessful will be displayed. **ALTERNATIVE FLOW 2: Pending bills and dues** If the user has any pending bills and dues against his/her account then the employee rejects the request and the user is asked to clear all the bills and dues. **ALTERNATIVE FLOW 3: Newly created account** If the account was created less than 7 days ago then the employee rejects the request to delete the account and the same is notified to the user.
6	**SPECIAL REQUIREMENTS:** none
7	**ASSOCIATED USE CASES:** none

3) Login

1	**INTRODUCTION:** This use case allows the actor to log in to the system to access the relevant functions that the system provides.
2	**ACTORS:** user, employee
3	**PRECONDITION:** The actor has to have a valid account.
4	**POSTCONDITION:** The actor is successfully logged into the system.The system displays the relevant homepage.
5	**FLOW OF EVENTS:** **BASIC FLOW:** 1. The user goes to the system's website. 2. The user selects the login button. 3. User enters valid account number/username and password. 4. User is now logged in to the system. **ALTERNATIVE FLOW: Actor enters wrong username/password** In the event that the actor enters an incorrect username or password, a message indicating the error will be displayed, stating that an incorrect password or username was entered. If the user tries to log in more than three times and fails, their account will become blocked. To unblock the account, the actor will need to visit a nearby branch in person.
6	**SPECIAL REQUIREMENTS:** none
7	**ASSOCIATED USE CASES:** none

4) Manage transactions

1	**INTRODUCTION:** This use case outlines all the processes involved in withdrawing money, depositing funds, reviewing transaction history, transferring money, and utilizing other Netbanking services.
2	**ACTORS:** user, employee, payment gateway
3	**PRECONDITION:** The user must be logged in to their account.
4	**POSTCONDITION:** The successful completion of the transaction will result in the database being updated with the amount withdrawn, deposited or transferred.

5

FLOW OF EVENTS:

1)Withdraw money:

BASIC FLOW:

1. The user logs in to their account.
2. The user then clicks on the ‘Make Transaction’ button.
3. The user then selects the option to withdraw money.
4. The user enters the amount he/she wish to withdraw.
5. The system verifies that the user has sufficient funds in their account to complete the withdrawal.
6. The user's request to withdraw funds is reviewed and authorized by the system.
7. The funds are transferred from the user's account to the withdrawal location, such as an ATM or bank teller.
8. The user receives a confirmation of the successful withdrawal, including the updated account balance.
9. The database is updated to reflect the withdrawal from the user's account.

ALTERNATIVE FLOW 1:Insufficient funds or frozen account

The user's request to withdraw funds is reviewed and denied by the system due to insufficient funds or a frozen account. In such cases, an error message is displayed to the user indicating that the withdrawal cannot be completed due to insufficient funds or a frozen account. The user may choose to deposit additional funds into their account or resolve the issue with their account before attempting to withdraw funds again.

2) Deposit funds:

BASIC FLOW:

1. The user logs in to their account.
2. The user then selects ‘Make Transaction’ button.
3. The user selects the option to deposit funds.
4. The user enters the amount he/she wish to deposit.
5. The system verifies the user's chosen deposit method, such as cash or cheque.
6. The user's request to deposit funds is reviewed and authorized by the system.
7. The funds are transferred to the user's account and reflected in the account balance.
8. The user receives a confirmation of the successful deposit, including the updated balance
9. The database is updated to reflect the deposit of funds into the user's account.

ALTERNATIVE FLOW 1: Issues with the deposit method

The user's request to deposit funds is reviewed and denied by the system due to an issue with the deposit method, such as fraudulent cheques and cash inconsistencies. Then, an error message is displayed to the user indicating that the deposit cannot be completed due to an issue with the deposit method. The user may choose to correct the issue with their deposit method or select a different deposit method before attempting to deposit funds again.

3)Transfer funds:

BASIC FLOW:

1. The user logs in to their account.
2. The user then selects ‘Make Transaction’ button.

	3. The user selects the option to transfer funds. 4. The user enters the details of the transfer, such as the recipient's account number, Bank name and the transfer amount. 5. The system verifies that the recipient's account exists and is eligible to receive funds. 6. The user's request to transfer funds is reviewed and authorized by the system. 7. The funds are transferred from the user's account to the recipient's account. 8. The user receives a confirmation of the successful transfer, including the updated account balance. 9. The database is updated to reflect the transfer of funds from the user's account to the recipient's account. **ALTERNATIVE FLOW 1: Ineligible account** The user's request to transfer funds is reviewed and denied by the system due to the account being ineligible. In this case, an error message is displayed to the user indicating that the transfer cannot be completed due to the account being ineligible. The user may choose to correct the issue with the account or select a different account before attempting to transfer funds again. **ALTERNATIVE FLOW 2: Insufficient funds or frozen account** The user's request to transfer funds is reviewed and denied by the system due to insufficient funds in the user’s account or a frozen account. In such cases, an error message is displayed to the user indicating that the transfer cannot be completed due to insufficient funds or a frozen account. The user may choose to use a different account or deposit funds to his/her account before attempting to transfer funds again. 4)View Transaction History: **BASIC FLOW:** 1. The user logs in to their account. 2. The user selects the option to view their transaction history. 3. The user specifies the time period for which he/she want to view their transaction history. 4. The system retrieves the transaction history for the specified time period from the database. 5. The retrieved transaction history is displayed to the user, including the date, amount, and type of each transaction. 6. The user may choose to sort or filter the transaction history based on different criteria, such as type or amount. **ALTERNATIVE FLOW 1: No transaction found** The system determines that no transactions have been recorded for the specified time period. In this case, an error message is displayed to the user indicating that no transactions have been found for the specified time period. The user may choose to select a different time period or view transactions from a different account in case of multiple accounts.
6	**SPECIAL REQUIREMENTS:** none
7	**ASSOCIATED USE CASES:** none

5) Loan Management

1	**INTRODUCTION:** This document outlines the procedures to be followed by one party (the loan applicant) and how the loan application process is carried out by other parties involved. It details the steps to be taken by the applicant while applying for a loan and the execution of the loan application and processing by the other actors.
2	**ACTORS:** user, employee
3	**PRECONDITION:** The user must be logged into the system.
4	**POSTCONDITION:** Upon the completion of this use case, the loan application will be received and reviewed. If approved, the funds will be provided to the user’s bank account with a specified rate of interest.
5	**FLOW OF EVENTS:** **BASIC FLOW:** 1. The user logs in to their account. 2. The user accesses the "Apply for Loan" section. 3. The user submits a loan request by filling the application form. 4. The system verifies the information provided by the applicant. 5. The system performs a credit check to assess the applicant's creditworthiness. 6. The employee reviews the loan application and all supporting documentation. 7. The employee then makes a decision to approve/reject the loan application. 8. If the application is rejected, the user is notified about the same along with the issues. 9. The user may choose to apply again after a certain period of time. 10. If the loan is approved, the funds will be provided to the user’s bank account with a specified rate of interest. 11. The system updates the loan information in the bank's records. **ALTERNATIVE FLOW 1: Credit history not meeting standards** If the applicant's credit history or employment status is deemed unsatisfactory, or if he/she do not meet the minimum income requirements set by the bank, the loan application would be denied, and the applicant would be informed of the reasons for the denial.
6	**SPECIAL REQUIREMENTS:** none
7	**ASSOCIATED USE CASES:** Manage transactions

6) Manage account cards and books

1	**INTRODUCTION:** This use case outlines the process a user follows when making a request for a credit card, debit card, passbook, or cheque book, as well as the actions taken by other actors to fulfil the request.
2	**ACTORS:** user, employee
3	**PRECONDITION:** The user must be logged into the system
4	**POSTCONDITION:** The user now has a valid credit card, debit card, cheque book, and passbook which can be used to manage transactions.
5	**FLOW OF EVENTS:** **BASIC FLOW:** 1. The user logs in to their account. 2. The user accesses the option to apply for account cards or books. 3. The display of the available options for cards and account books will be presented to the user, which includes credit card, debit card, passbook, and cheque book. 4. Upon selecting a card or book, the user will be redirected to a new page where he/she will complete a form. 5. The applicant submits the application form. 6. The employee verifies the user's account information and eligibility. 7. The employee approves the request and issues the account cards or books to the user. 8. The user receives the same and the system updates the user's account information to reflect the issuance of the credit card, debit card, passbook, and cheque book. **ALTERNATIVE FLOW 1: Ineligible user** If the user does not meet the eligibility criteria, the request will be denied and the user will be informed about the reasons for the denial. This may include inactive account, insufficient balance or poor transaction history.
6	**SPECIAL REQUIREMENTS:** none
7	**ASSOCIATED USE CASES:** Manage transactions

7) Manage Issues

1	**INTRODUCTION:** This use case describes the steps involved in reporting and resolving an issue.
2	**ACTORS:** user, employee, system administrator
3	**PRECONDITION:** The actor must be logged into the system.
4	**POSTCONDITION:** The issue is successfully reported and resolved, and the user is satisfied with the solution.
5	**FLOW OF EVENTS:** **BASIC FLOW:** 1. The user logs in to their account. 2. The user selects the option to report an issue. 3. The system displays a form for the user to complete, including description of the issue, the date the issue occurred, and any other relevant details. 4. The user fills the form and submits the issue. 5. The system generates a report of the issue and notifies it to an employee. 6. The system administrator is notified about the same. 7. The system administrator reviews the issue and determines a solution. The employee implements the solution and updates the issue report with the resolution and any other relevant information. 8. The system updates the status of the issue to reflect that it has been resolved. **ALTERNATIVE FLOW 1: Administrator unable to solve the issue** If the administrator is unable to resolve the issue, the issue may be escalated to a higher level of support or to a specialist team. If the issue cannot be resolved, the employee may offer alternative solutions to the user.
6	**SPECIAL REQUIREMENTS:** none
7	**ASSOCIATED USE CASES:** none

8) Logout

1	**INTRODUCTION:** This use case enables the user to safely end their session and log out of their account.
2	**ACTORS:** user, employee
3	**PRECONDITION:** The user/employee must be logged into the system.
4	**POSTCONDITION:** The user's session is terminated and the user is safely out of the system.
5	**FLOW OF EVENTS:** **BASIC FLOW:** 1. The user clicks the logout button or option in the user interface. 2. The system receives the logout request and terminates the user's session. 3. The system returns the user to a secure landing page or login page. **ALTERNATIVE FLOW 1: Automatic logout/session expired** This occurs when the user does not voluntarily log out of the system, but instead their session is terminated due to inactivity for a certain amount of time. This helps to ensure the security of the system and the protection of sensitive user data.
6	**SPECIAL REQUIREMENTS:** none
7	**ASSOCIATED USE CASES:** none

3.2Non-functional Requirements

3.2.1 Performance

The following requirements must be met for system performance:

- The system must be able to handle a large number of users and transactions simultaneously.
- The system response time must be within acceptable limits for all user actions.
- The system must be able to recover from failures and errors within a reasonable time.

3.2.2 Reliability

The following requirements must be met for system reliability:

- The system must be available 24/7, except for scheduled maintenance windows.
- The system must be able to recover from hardware or software failures without losing any user data.
- The system must provide backups and disaster recovery mechanisms to ensure data integrity and availability.

3.2.3 Usability

The following requirements must be met for system usability:

- The system must be easy to use and navigate for both customers and bank employees.
- The system must have a responsive and intuitive user interface that is accessible from different devices and browsers.
- The system must provide clear and concise instructions for all user actions.

3.2.4 Security

The following requirements must be met for system security:

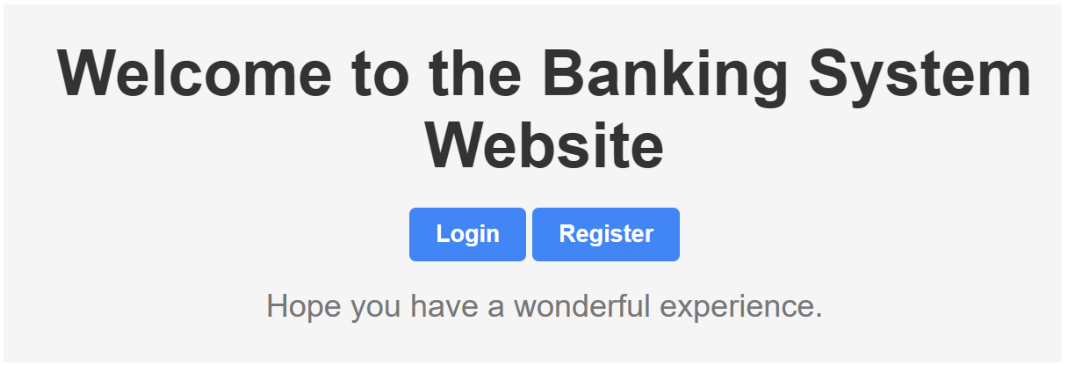
- The system must comply with applicable security standards such as PCI DSS, GDPR, and ISO 27001.
- The system must use encryption to protect user data in transit and at rest.
- The system must implement access control and authentication mechanisms to prevent unauthorized access to user accounts and data.

3.3 Other Requirements

- The system must be compatible with different types of payment gateways and banks.
- The system must provide APIs for integrating with third-party applications.
- The system must provide documentation and training materials for users and administrators.

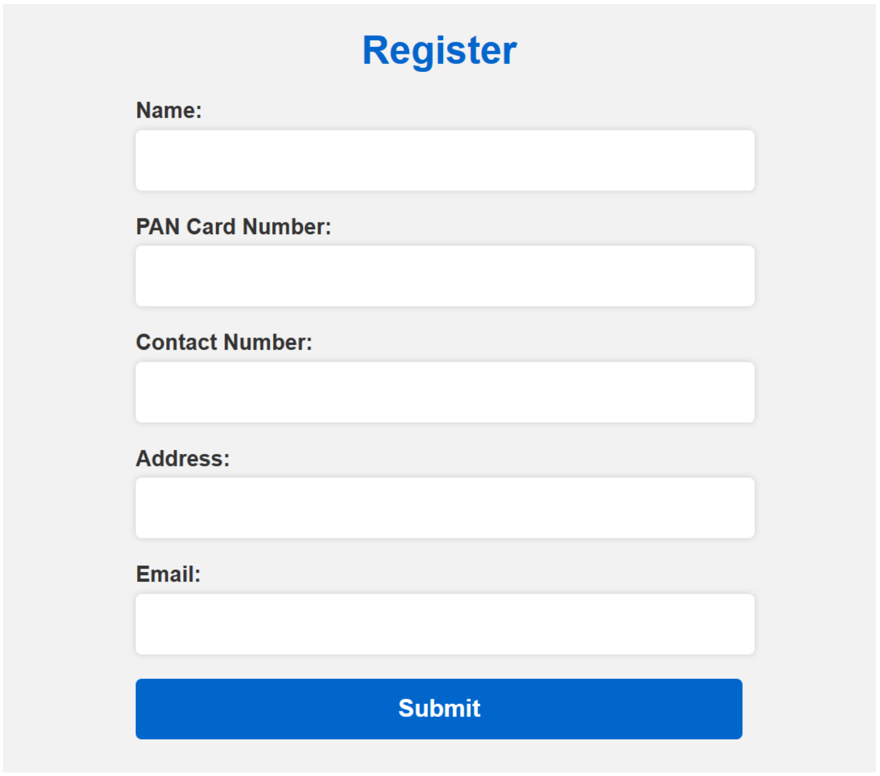
3.4 External Interfaces

Home page

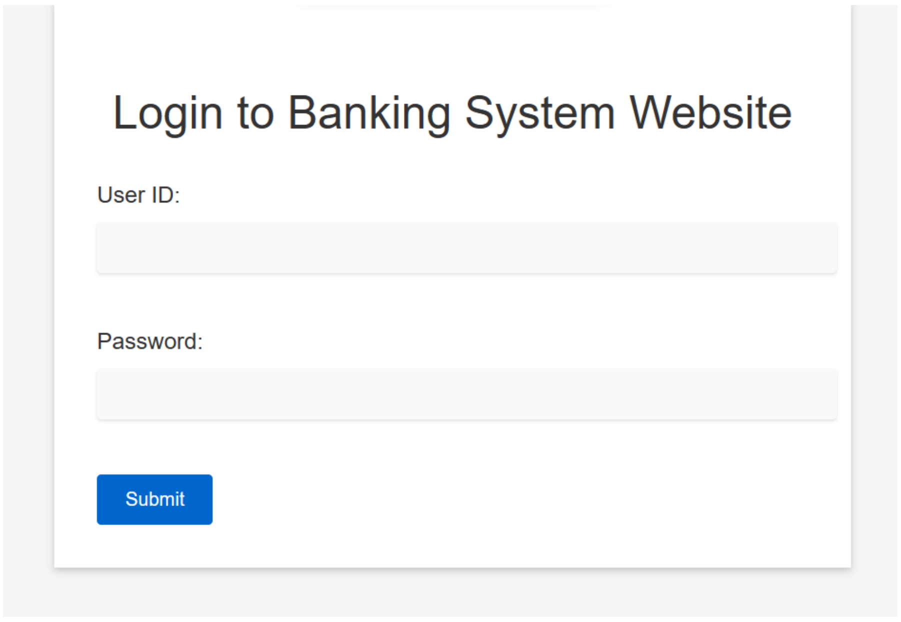

Welcome to the Banking System Website

Hope you have a wonderful experience.

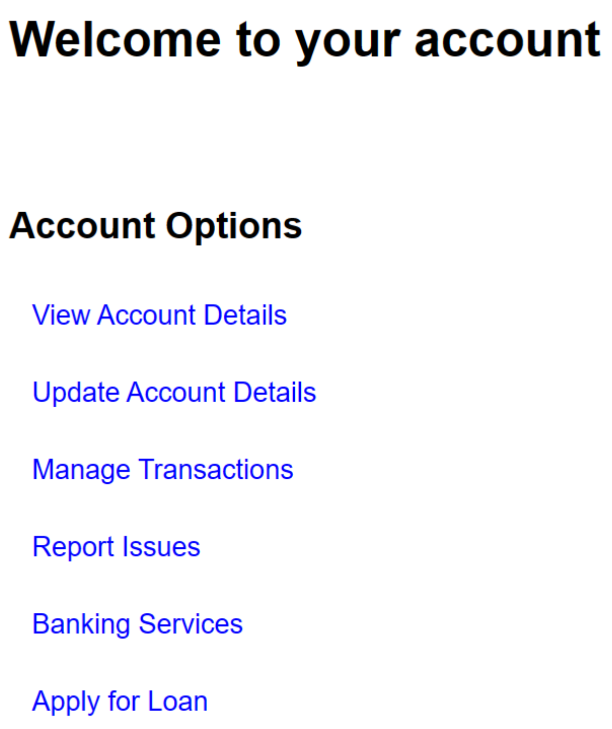
Register page



login page



Account page



Welcome to your account

Account Options

View Account Details

Update Account Details

Manage Transactions

Report Issues

Banking Services

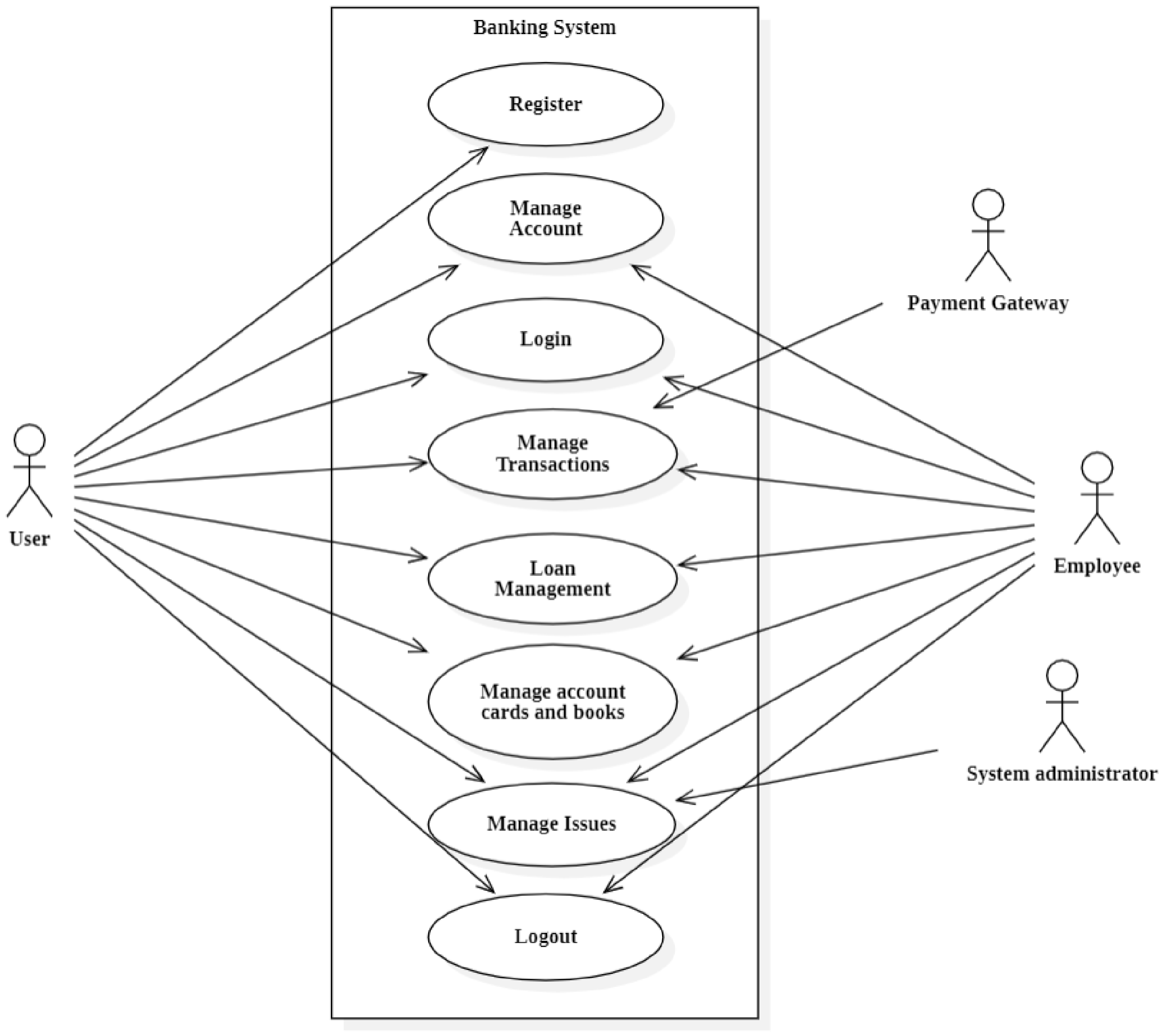
Apply for Loan

4. Appendix

CONTENTS

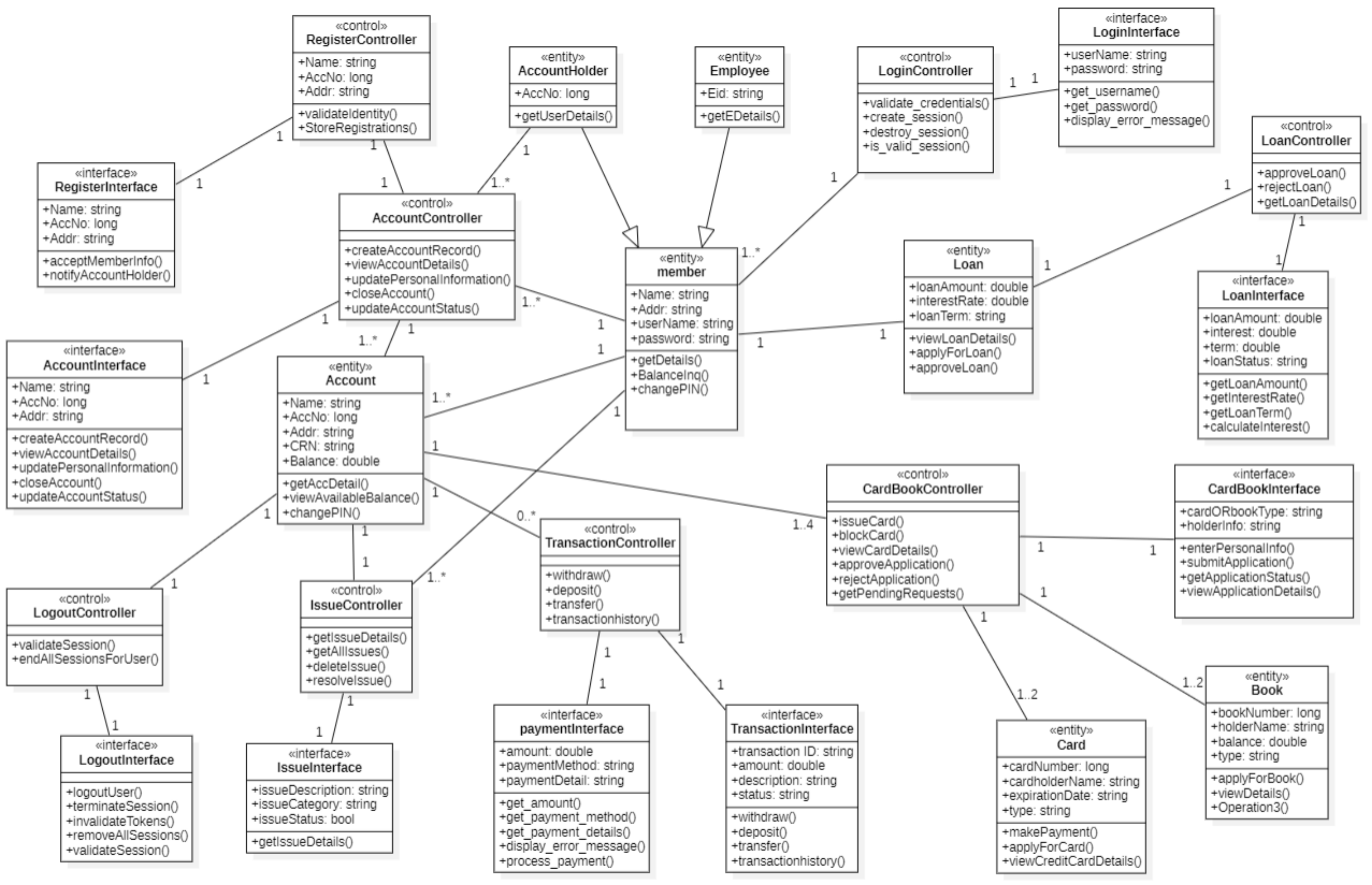
- Use Case Diagram
- Class Diagram
- Sequence Diagrams
- Activity Diagrams
- Statechart diagrams

USE CASE DIAGRAM



CLASS DIAGRAM

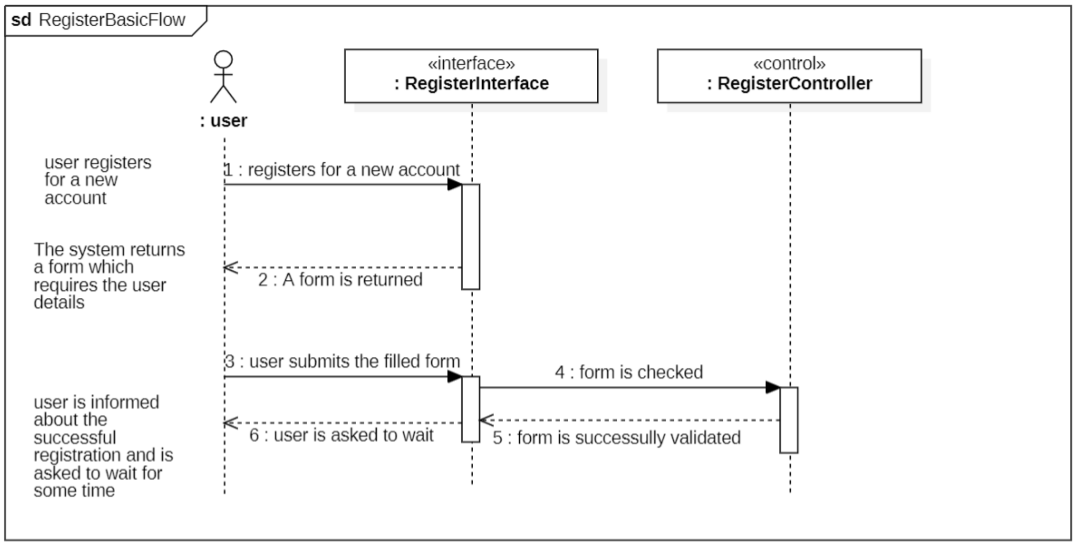
S. No.	Use case	Entity classes	Interface classes	Control classes
1	Register	Member	RegisterInterface	registerController
2	Manage Account	Member, Account	AccountInterface	AccountController, registerController
3	Login	Member, Account	LoginInterface	LoginController
4	Manage Transaction	Member, Account	TransactionInterface, Payment Interface	TransactionController
5	Loan Management	Member, Account, loan	LoanInterface	LoanController
6	Manage account cards and books	Member, Account, Card, Book	CardBookInterface	CardBookController
7	Manage Issues	Member, Account	IssueInterface	IssueController
8	Logout	Member, Account	LogoutInterface	LogoutController



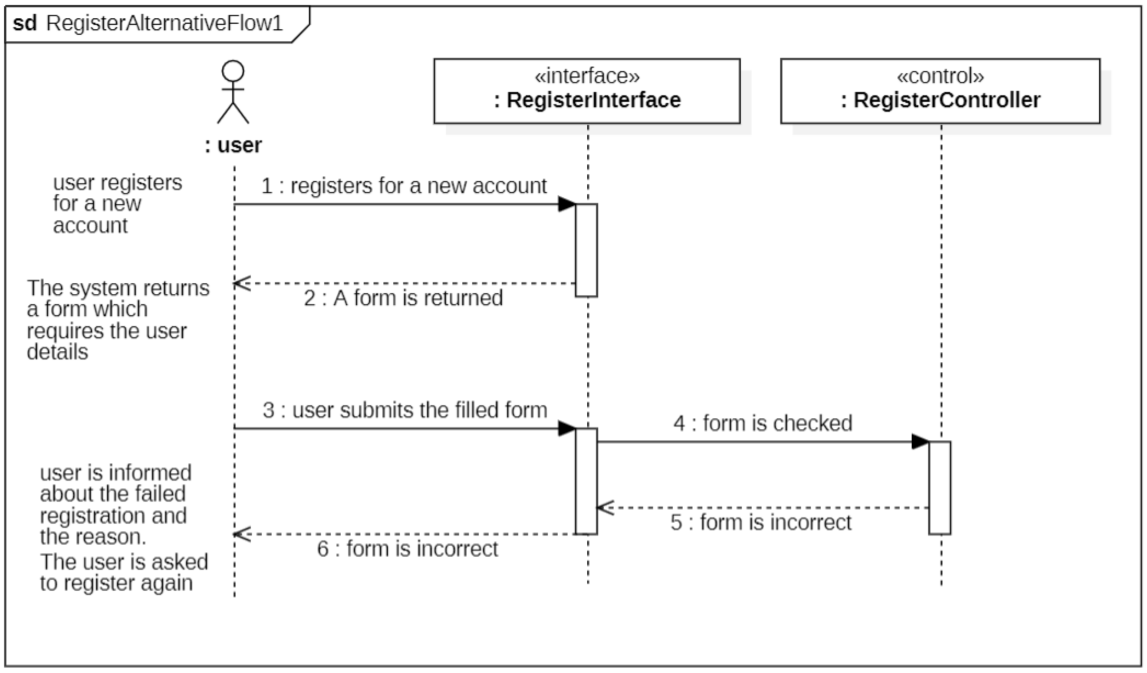
SEQUENCE DIAGRAMS

1)Register

BASIC FLOW


sd RegisterBasicFlow
: user
«interface»
: RegisterInterface
«control»
: RegisterController
user registers for a new account
1 : registers for a new account
The system returns a form which requires the user details
2 : A form is returned
3 : user submits the filled form
4 : form is checked
user is informed about the successful registration and is asked to wait for some time
6 : user is asked to wait
5 : form is successully validated

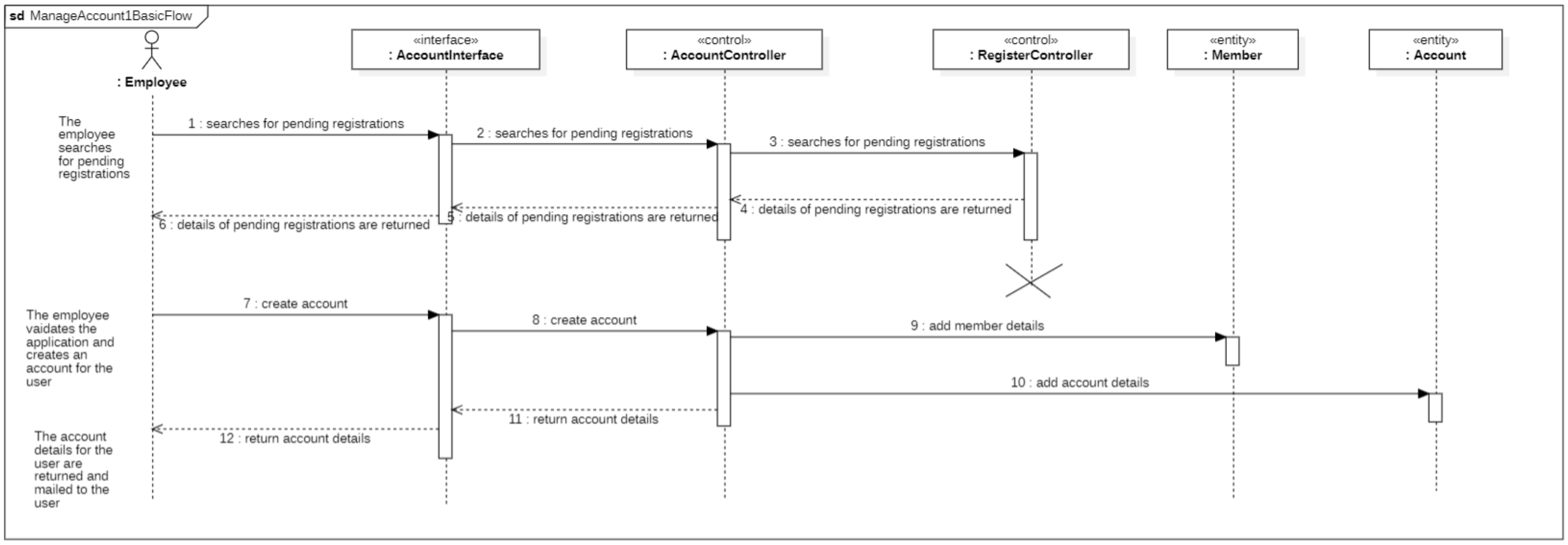
ALTERNATIVE FLOW 1: Incorrect application


sd RegisterAlternativeFlow1
: user
«interface»
: RegisterInterface
«control»
: RegisterController
user registers for a new account
1 : registers for a new account
The system returns a form which requires the user details
2 : A form is returned
3 : user submits the filled form
4 : form is checked
user is informed about the failed registration and the reason.
The user is asked to register again
5 : form is incorrect
6 : form is incorrect

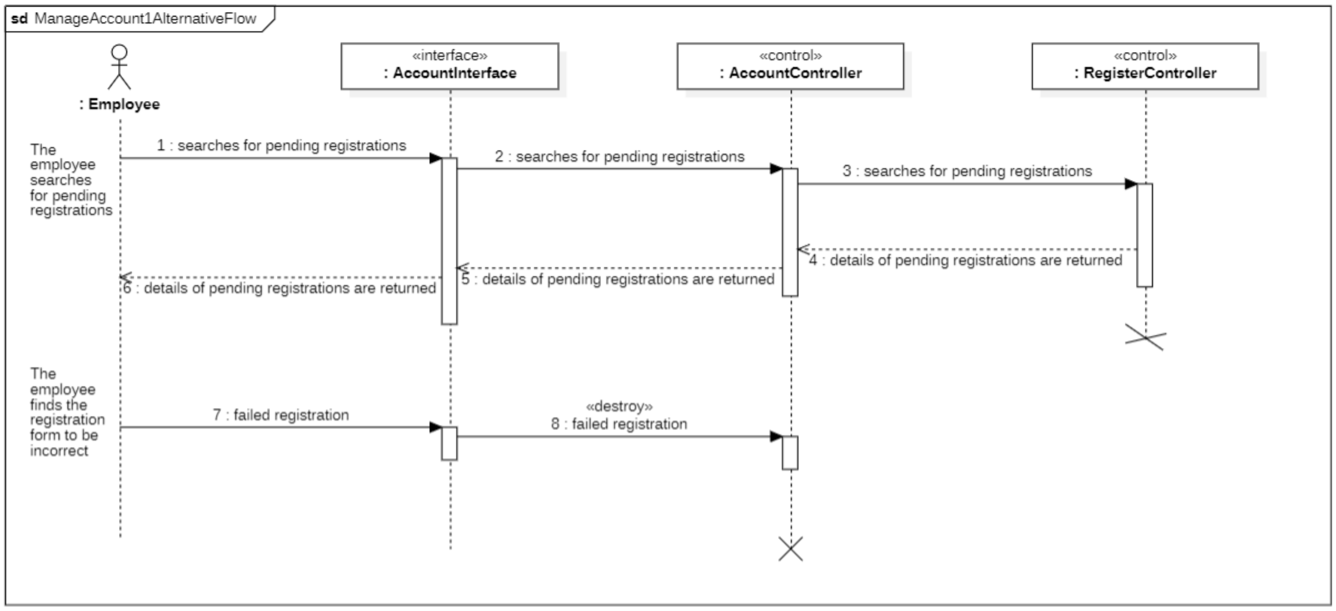
<u>2) Manage account</u>

Create a new account:

BASIC FLOW

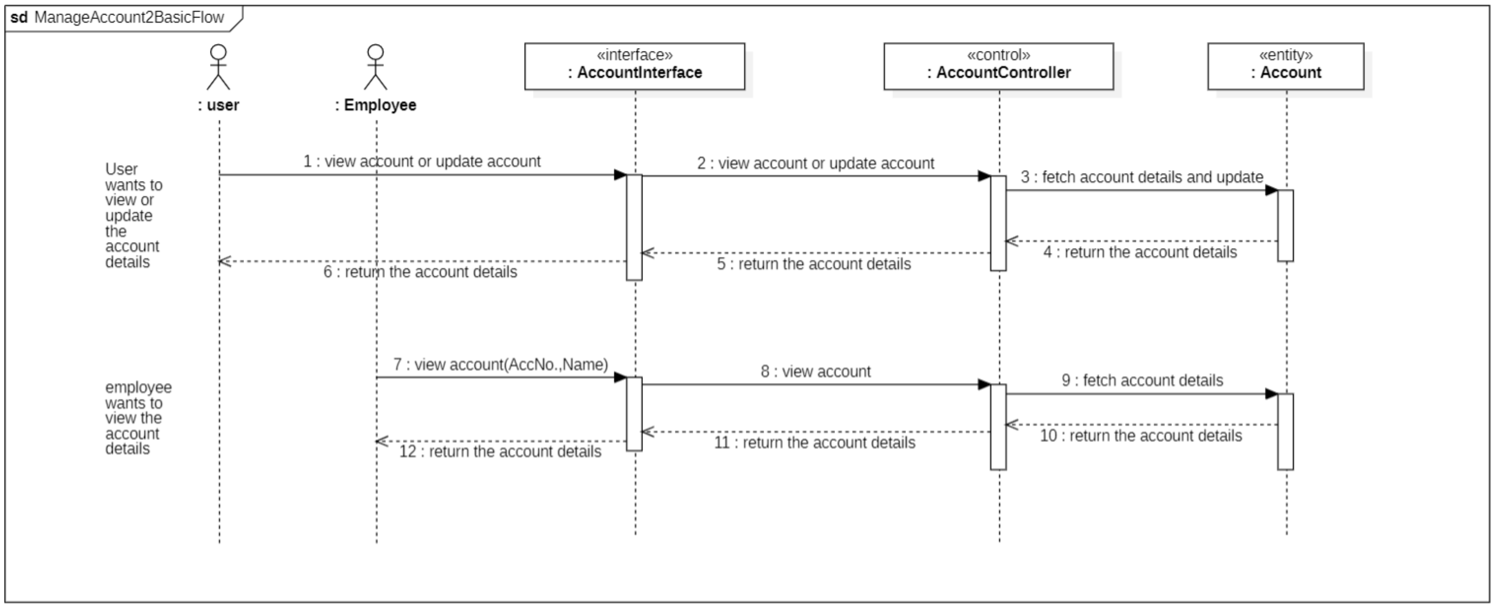

sd ManageAccount1BasicFlow
: Employee
«interface» : AccountInterface
«control» : AccountController
«control» : RegisterController
«entity» : Member
«entity» : Account
The employee searches for pending registrations
1 : searches for pending registrations
2 : searches for pending registrations
3 : searches for pending registrations
4 : details of pending registrations are returned
5 : details of pending registrations are returned
6 : details of pending registrations are returned
The employee vaidates the application and creates an account for the user
7 : create account
8 : create account
9 : add member details
10 : add account details
11 : return account details
12 : return account details
The account details for the user are returned and mailed to the user

ALTERNATIVE FLOW 1: Incorrect registration

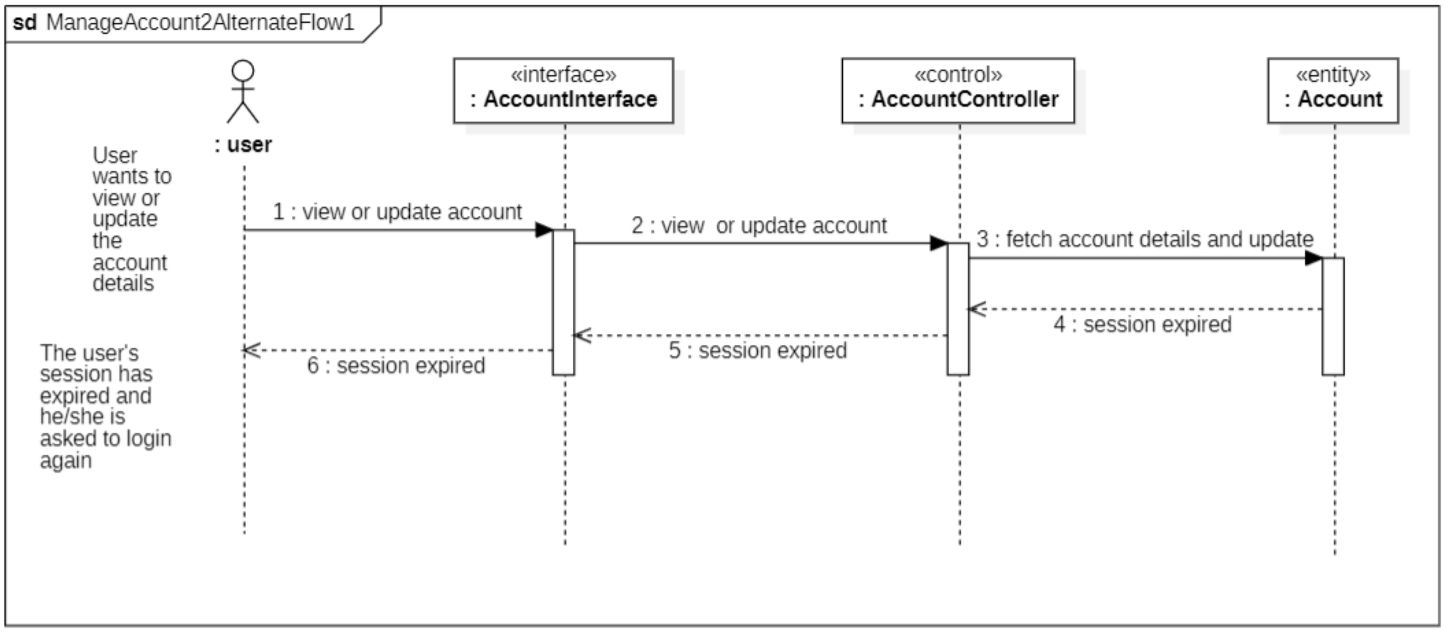

sd ManageAccount1AlternativeFlow
: Employee
«interface» : AccountInterface
«control» : AccountController
«control» : RegisterController
The employee searches for pending registrations
1 : searches for pending registrations
2 : searches for pending registrations
3 : searches for pending registrations
4 : details of pending registrations are returned
5 : details of pending registrations are returned
6 : details of pending registrations are returned
The employee finds the registration form to be incorrect
7 : failed registration
«destroy» 8 : failed registration

View or update account:

BASIC FLOW

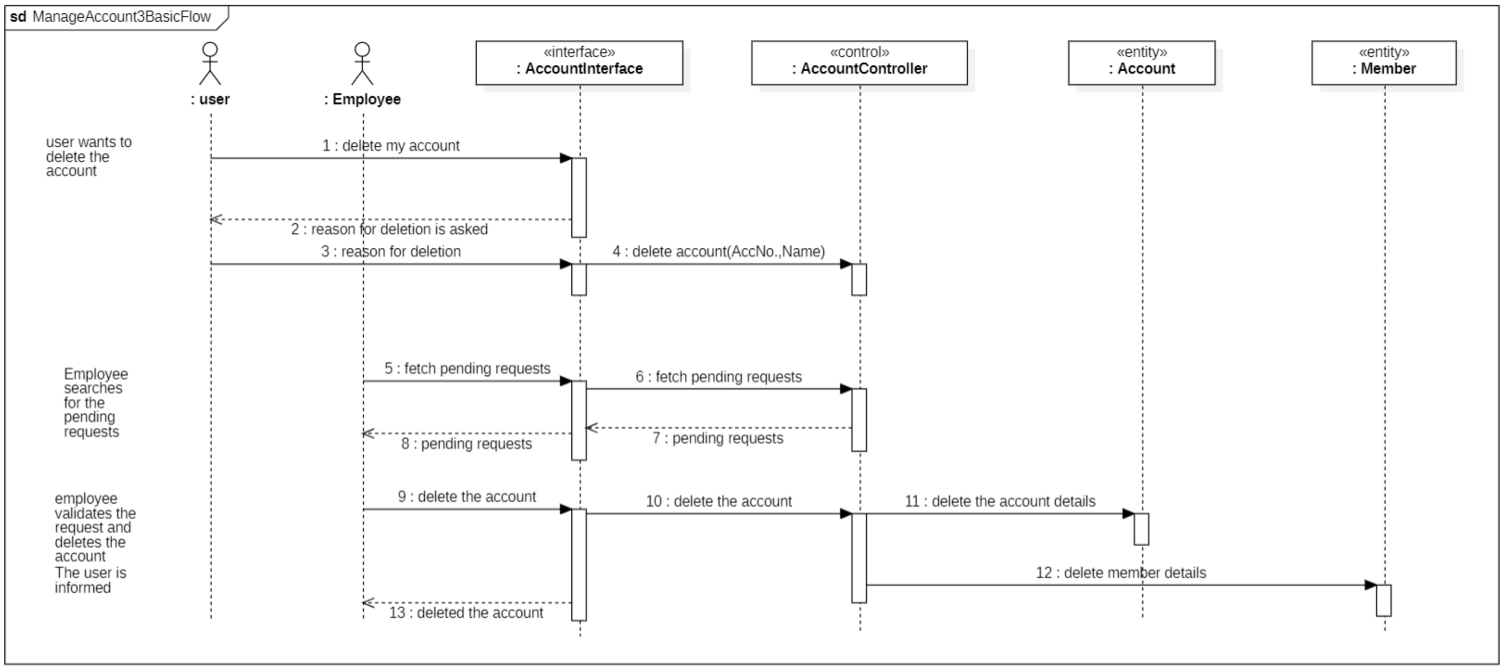

sd ManageAccount2BasicFlow
: user
: Employee
«interface» : AccountInterface
«control» : AccountController
«entity» : Account
User wants to view or update the account details
1 : view account or update account
2 : view account or update account
3 : fetch account details and update
4 : return the account details
5 : return the account details
6 : return the account details
employee wants to view the account details
7 : view account(AccNo.,Name)
8 : view account
9 : fetch account details
10 : return the account details
11 : return the account details
12 : return the account details

ALTERNATIVE FLOW 1: Session expired

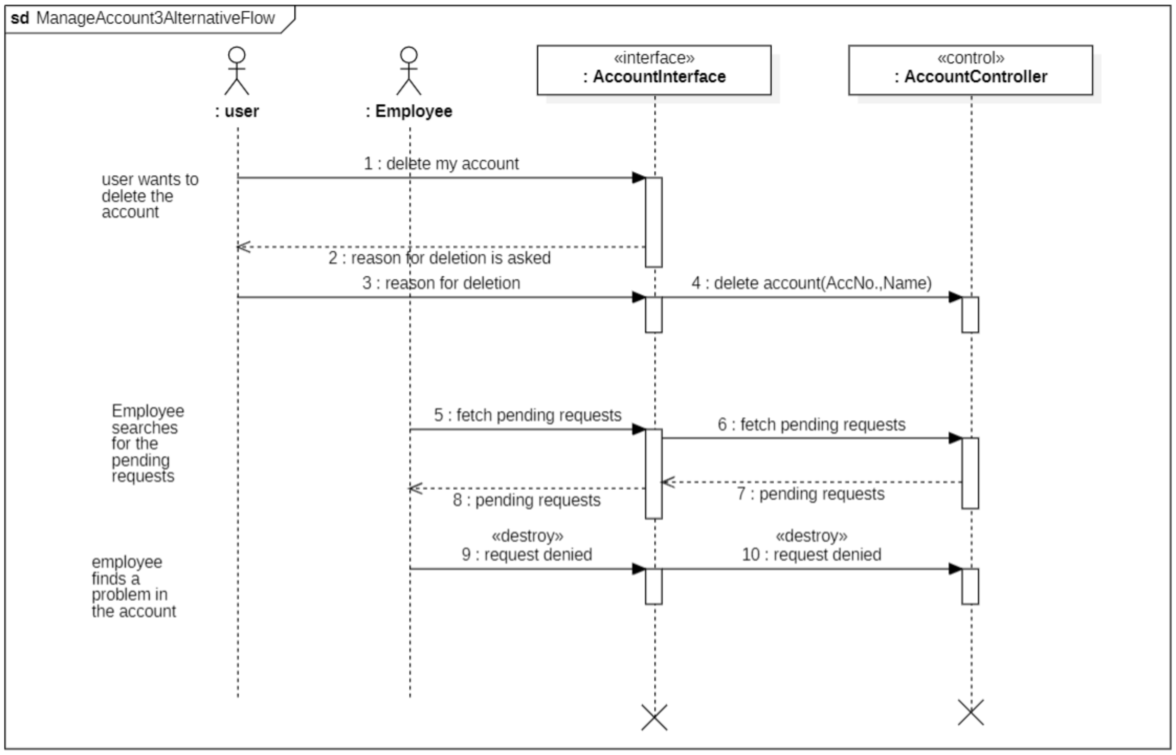

sd ManageAccount2AlternateFlow1
: user
«interface» : AccountInterface
«control» : AccountController
«entity» : Account
User wants to view or update the account details
1 : view or update account
2 : view or update account
3 : fetch account details and update
4 : session expired
5 : session expired
6 : session expired
The user's session has expired and he/she is asked to login again

Delete account:

BASIC FLOW

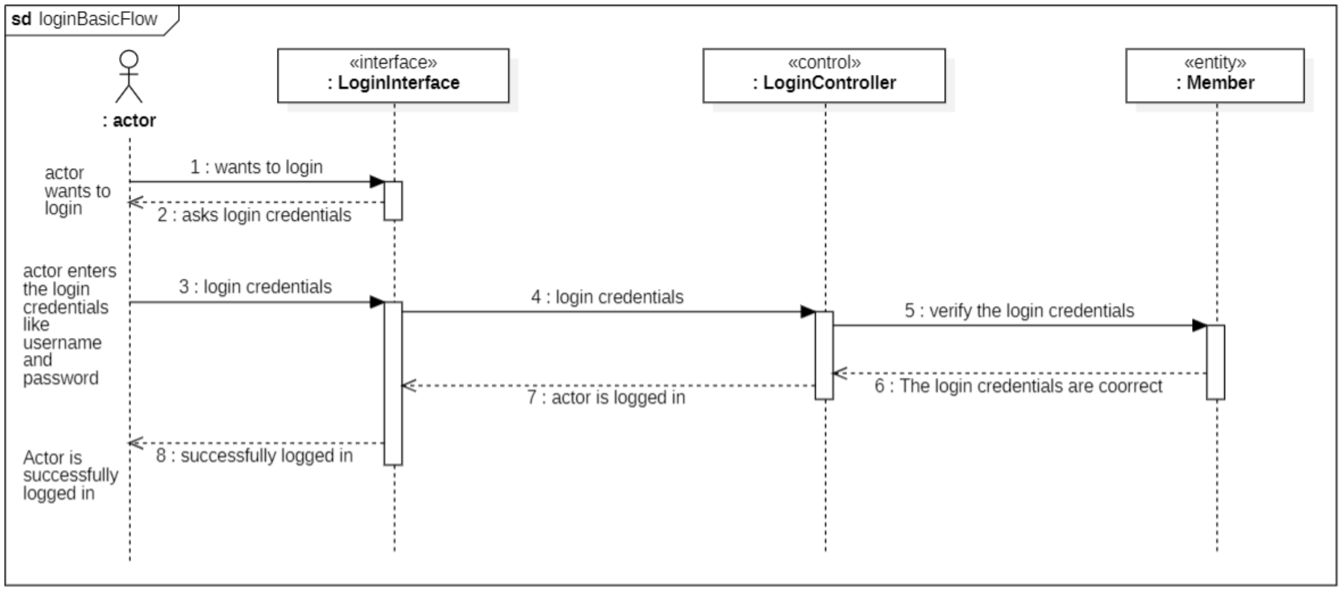

sd ManageAccount3BasicFlow
: user
: Employee
«interface» : AccountInterface
«control» : AccountController
«entity» : Account
«entity» : Member
user wants to delete the account
1 : delete my account
2 : reason for deletion is asked
3 : reason for deletion
4 : delete account(AccNo.,Name)
Employee searches for the pending requests
5 : fetch pending requests
6 : fetch pending requests
7 : pending requests
8 : pending requests
employee validates the request and deletes the account The user is informed
9 : delete the account
10 : delete the account
11 : delete the account details
12 : delete member details
13 : deleted the account

ALTERNATIVE FLOW: Problem with the account

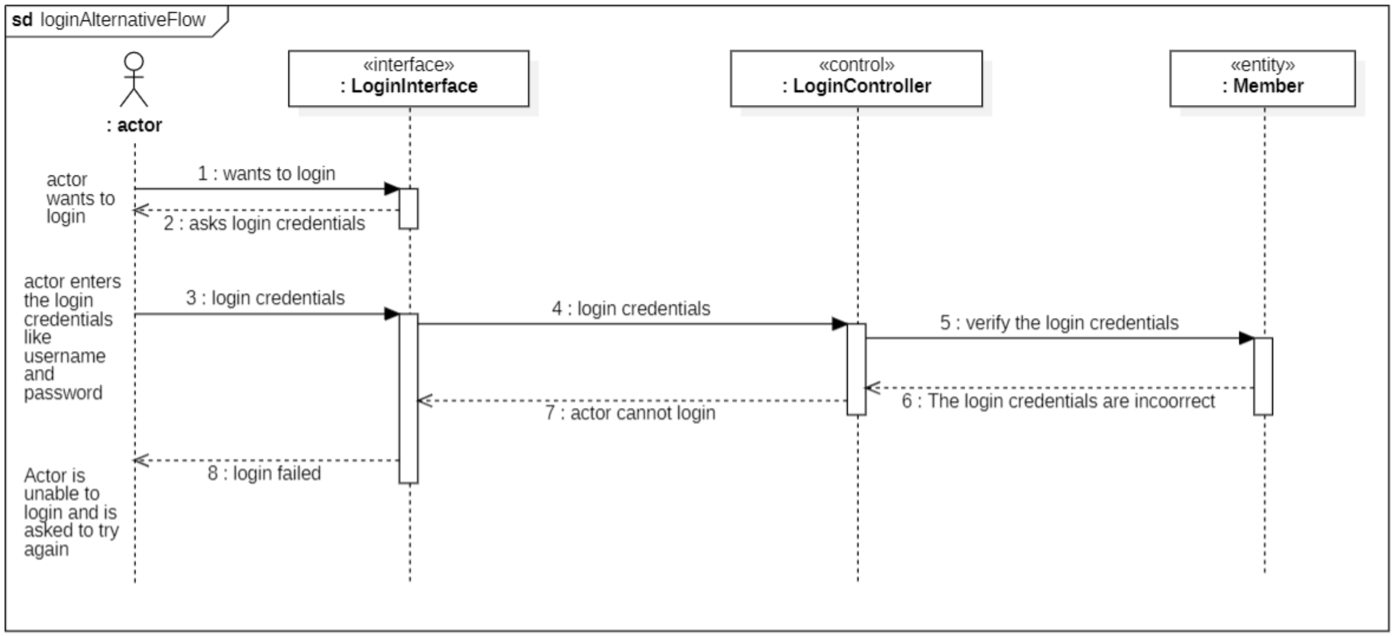

sd ManageAccount3AlternativeFlow
: user
: Employee
«interface» : AccountInterface
«control» : AccountController
user wants to delete the account
1 : delete my account
2 : reason for deletion is asked
3 : reason for deletion
4 : delete account(AccNo.,Name)
Employee searches for the pending requests
5 : fetch pending requests
6 : fetch pending requests
7 : pending requests
8 : pending requests
employee finds a problem in the account
«destroy» 9 : request denied
«destroy» 10 : request denied

3) Login

BASIC FLOW


sd loginBasicFlow
: actor
«interface» : LoginInterface
«control» : LoginController
«entity» : Member
actor wants to login
1 : wants to login
2 : asks login credentials
actor enters the login credentials like username and password
3 : login credentials
4 : login credentials
5 : verify the login credentials
6 : The login credentials are coorrect
7 : actor is logged in
8 : successfully logged in
Actor is successfully logged in

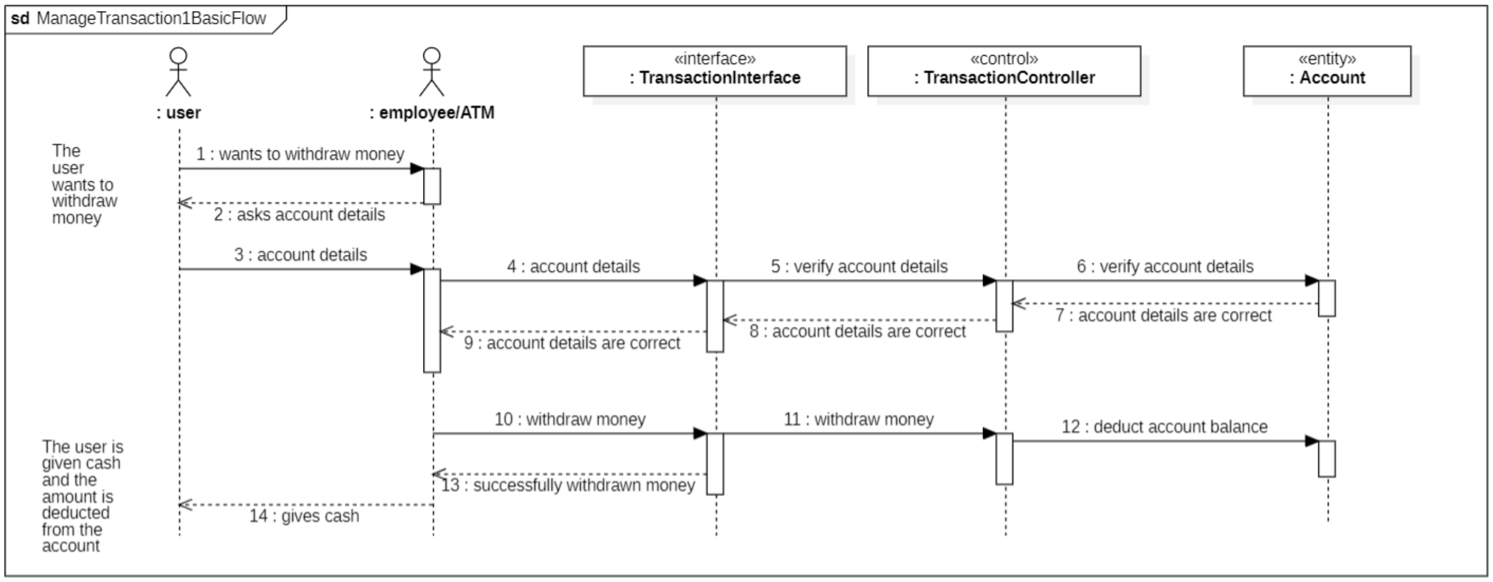
ALTERNATIVE FLOW: actor enters wrong username/password


sd loginAlternativeFlow
: actor
«interface» : LoginInterface
«control» : LoginController
«entity» : Member
actor wants to login
1 : wants to login
2 : asks login credentials
actor enters the login credentials like username and password
3 : login credentials
4 : login credentials
5 : verify the login credentials
6 : The login credentials are incoorrect
7 : actor cannot login
8 : login failed
Actor is unable to login and is asked to try again

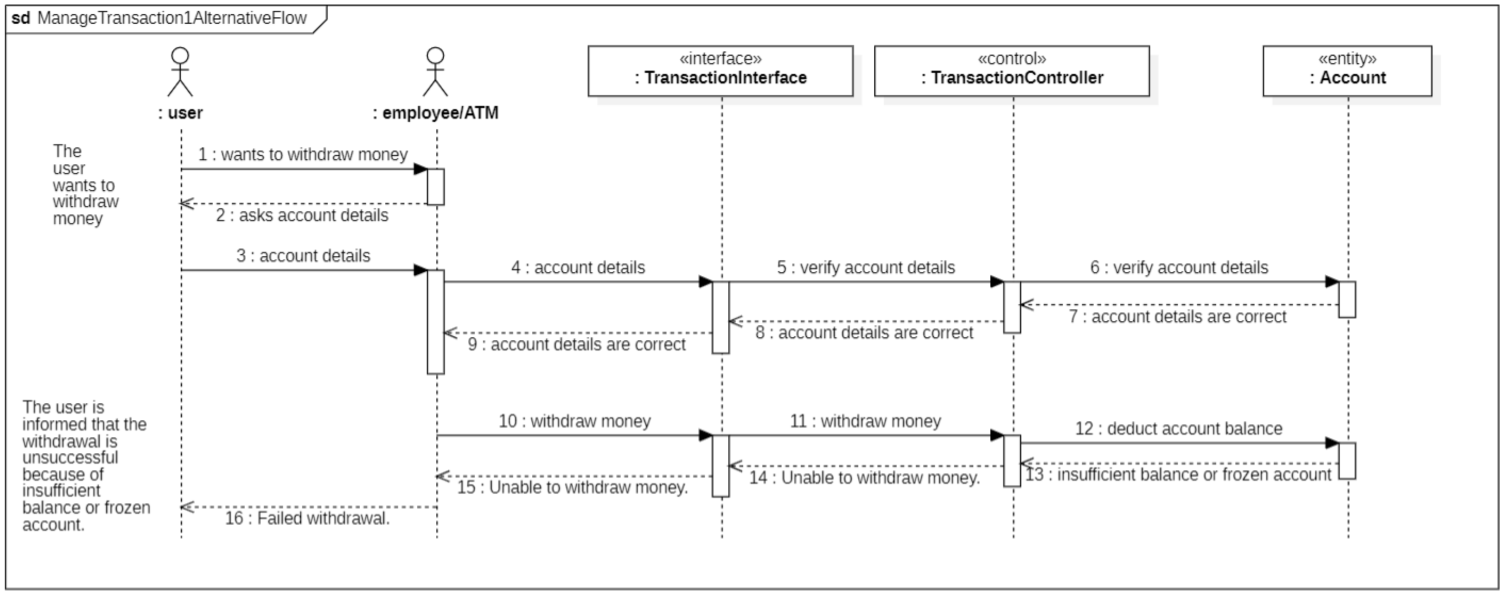
4) Manage transactions

Withdraw money:

BASIC FLOW

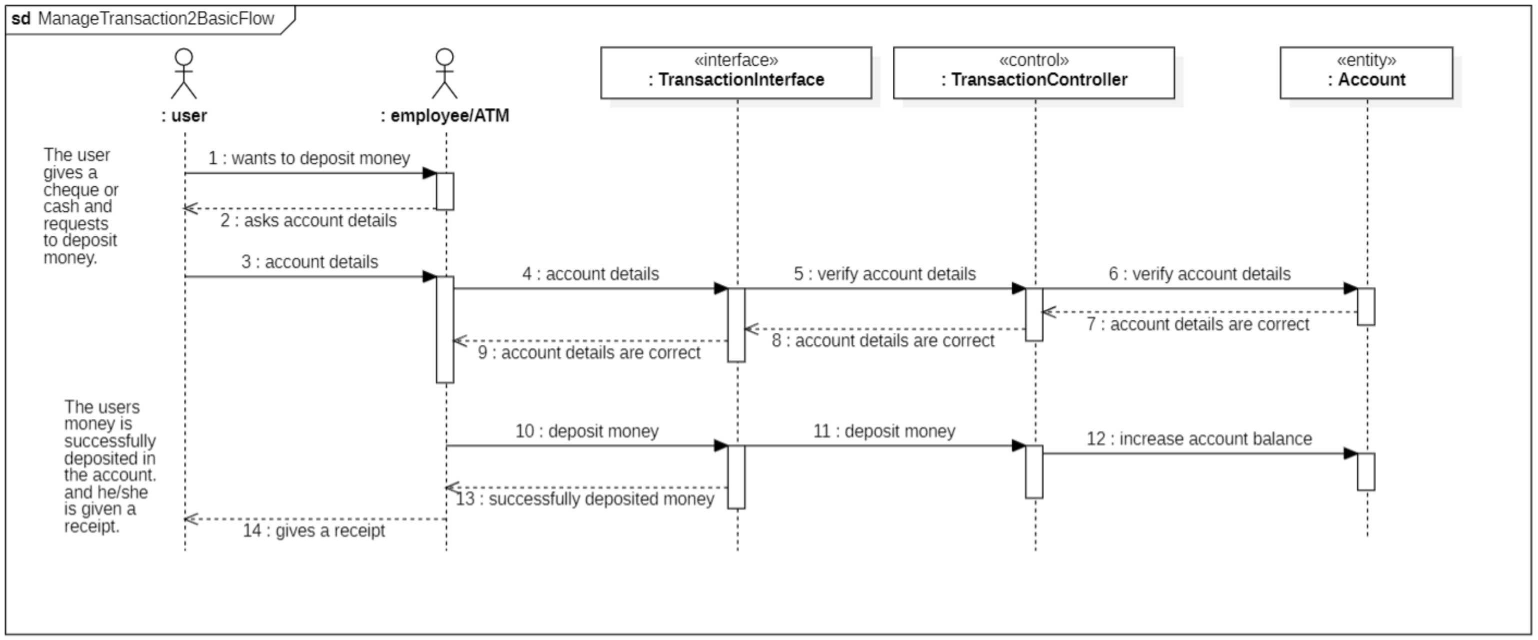

sd ManageTransaction1BasicFlow
: user
: employee/ATM
«interface» : TransactionInterface
«control» : TransactionController
«entity» : Account
The user wants to withdraw money
1 : wants to withdraw money
2 : asks account details
3 : account details
4 : account details
5 : verify account details
6 : verify account details
7 : account details are correct
8 : account details are correct
9 : account details are correct
10 : withdraw money
11 : withdraw money
12 : deduct account balance
13 : successfully withdrawn money
14 : gives cash
The user is given cash and the amount is deducted from the account

ALTERNATIVE FLOW 1: Insufficient funds or frozen account

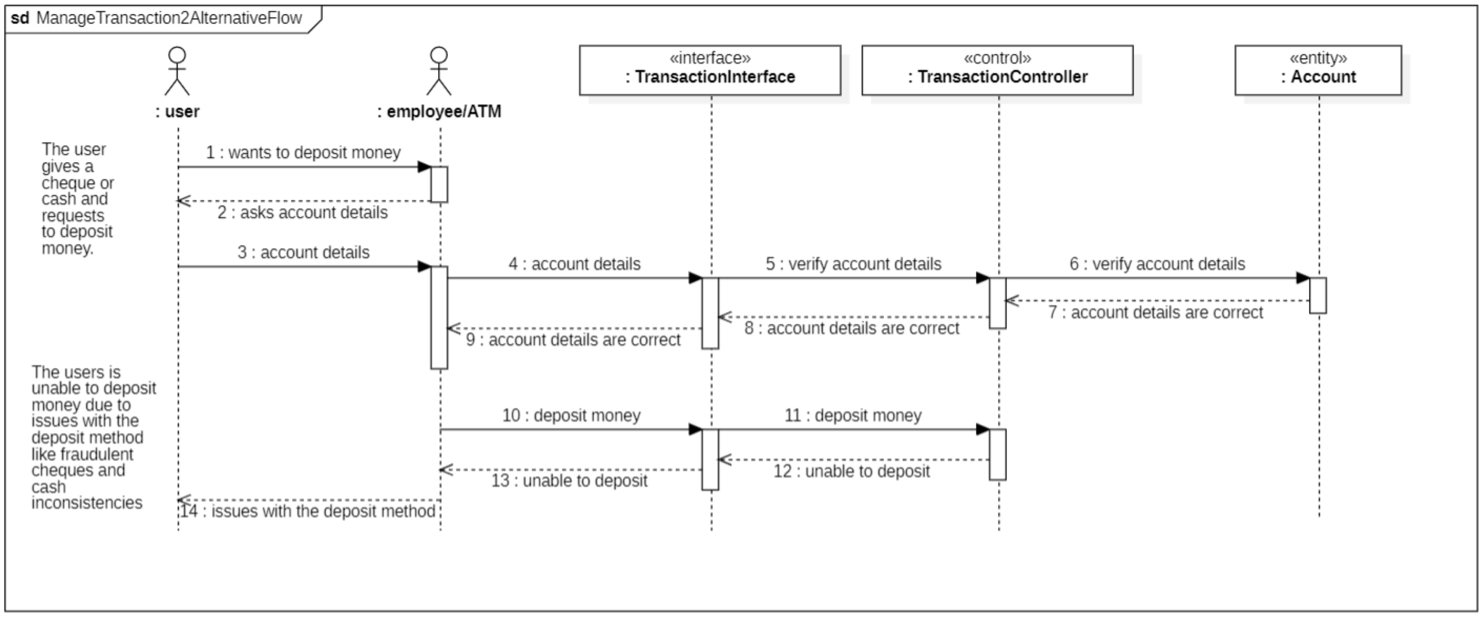

sd ManageTransaction1AlternativeFlow
: user
: employee/ATM
«interface» : TransactionInterface
«control» : TransactionController
«entity» : Account
The user wants to withdraw money
1 : wants to withdraw money
2 : asks account details
3 : account details
4 : account details
5 : verify account details
6 : verify account details
7 : account details are correct
8 : account details are correct
9 : account details are correct
10 : withdraw money
11 : withdraw money
12 : deduct account balance
13 : insufficient balance or frozen account
14 : Unable to withdraw money.
15 : Unable to withdraw money.
16 : Failed withdrawal.
The user is informed that the withdrawal is unsuccessful because of insufficient balance or frozen account.

Deposit funds:

BASIC FLOW

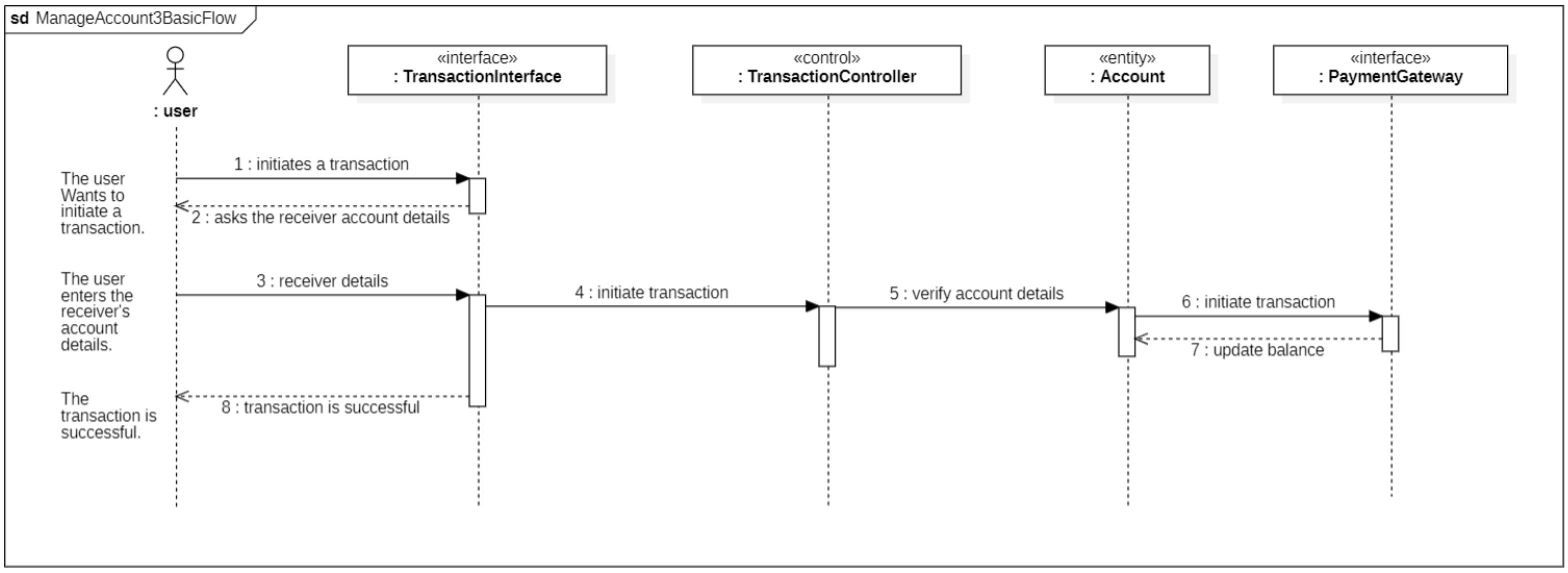

sd ManageTransaction2BasicFlow
: user
: employee/ATM
«interface» : TransactionInterface
«control» : TransactionController
«entity» : Account
The user gives a cheque or cash and requests to deposit money.
1 : wants to deposit money
2 : asks account details
3 : account details
4 : account details
5 : verify account details
6 : verify account details
7 : account details are correct
8 : account details are correct
9 : account details are correct
The users money is successfully deposited in the account. and he/she is given a receipt.
10 : deposit money
11 : deposit money
12 : increase account balance
13 : successfully deposited money
14 : gives a receipt

ALTERNATIVE FLOW 1: Issues with the deposit method

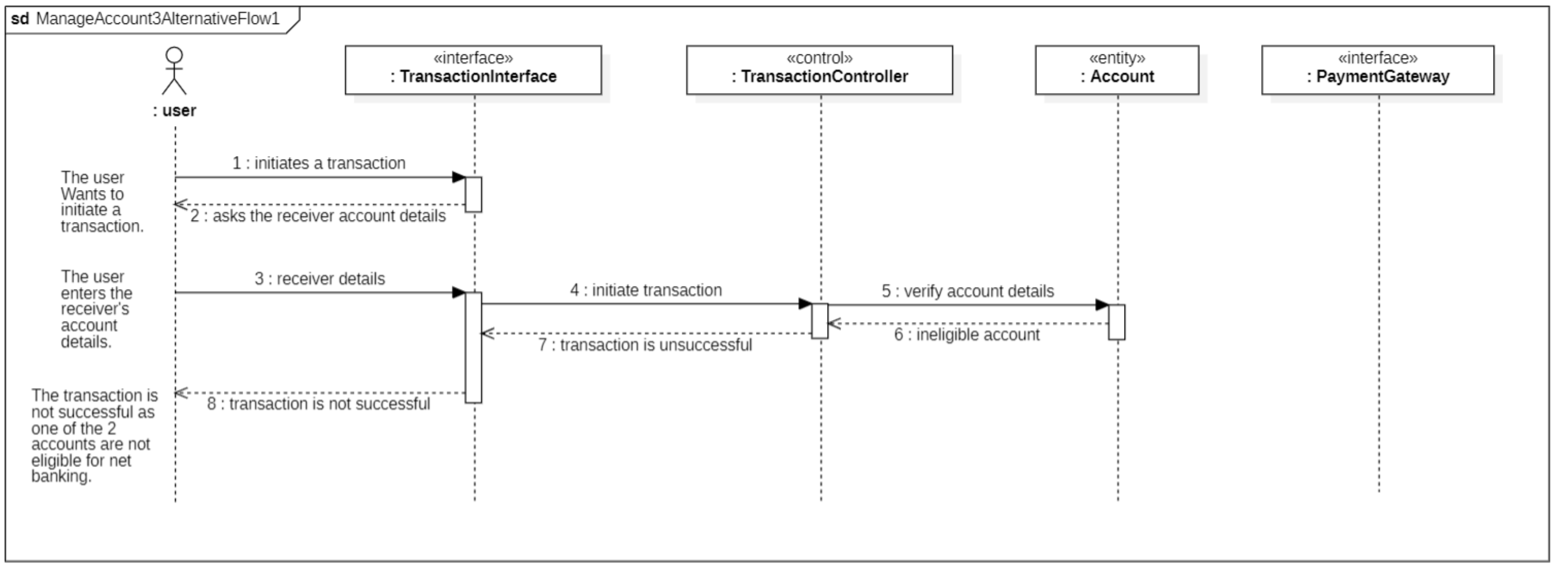

sd ManageTransaction2AlternativeFlow
: user
: employee/ATM
«interface» : TransactionInterface
«control» : TransactionController
«entity» : Account
The user gives a cheque or cash and requests to deposit money.
1 : wants to deposit money
2 : asks account details
3 : account details
4 : account details
5 : verify account details
6 : verify account details
7 : account details are correct
8 : account details are correct
9 : account details are correct
The users is unable to deposit money due to issues with the deposit method like fraudulent cheques and cash inconsistencies
10 : deposit money
11 : deposit money
12 : unable to deposit
13 : unable to deposit
14 : issues with the deposit method

Transfer funds:

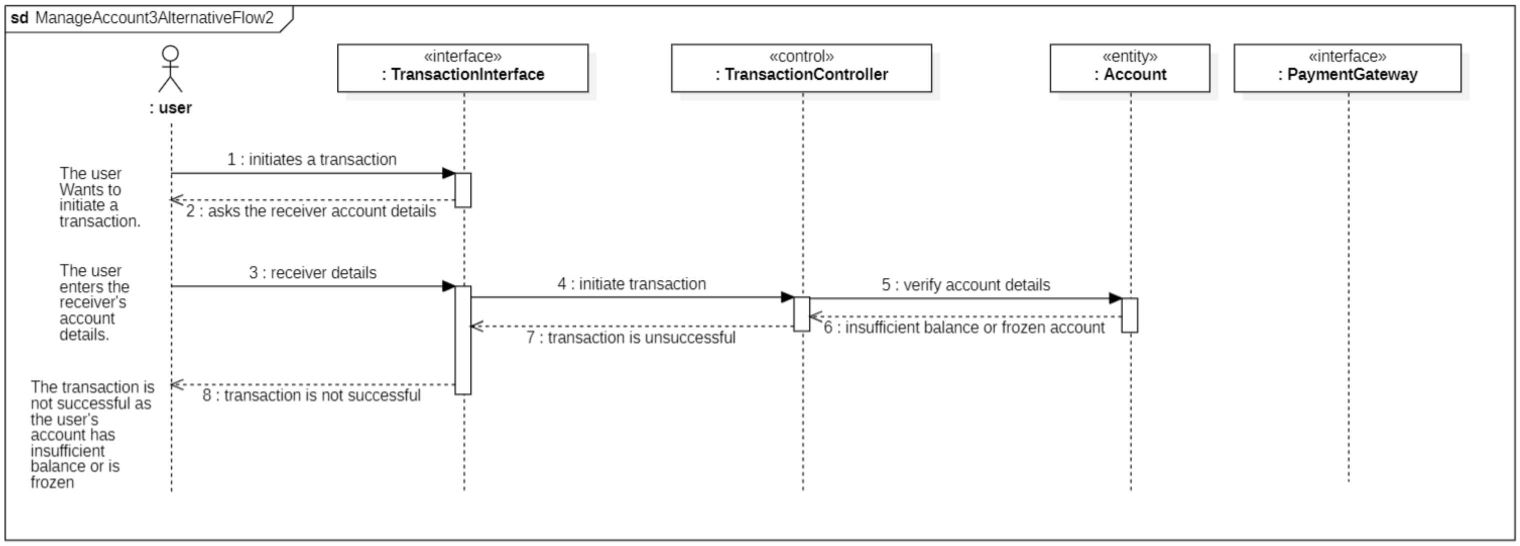
BASIC FLOW


sd ManageAccount3BasicFlow
: user
«interface» : TransactionInterface
«control» : TransactionController
«entity» : Account
«interface» : PaymentGateway
The user Wants to initiate a transaction.
1 : initiates a transaction
2 : asks the receiver account details
The user enters the receiver's account details.
3 : receiver details
4 : initiate transaction
5 : verify account details
6 : initiate transaction
7 : update balance
The transaction is successful.
8 : transaction is successful

ALTERNATIVE FLOW 1: Ineligible account

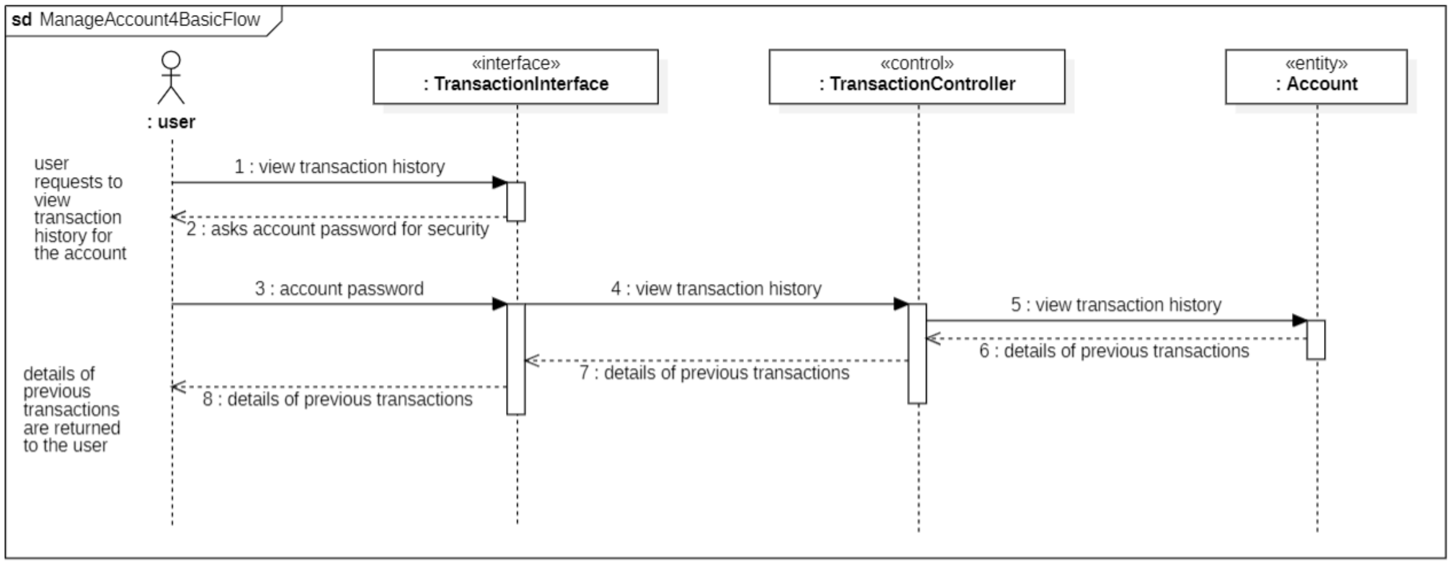

sd ManageAccount3AlternativeFlow1
: user
«interface» : TransactionInterface
«control» : TransactionController
«entity» : Account
«interface» : PaymentGateway
The user Wants to initiate a transaction.
1 : initiates a transaction
2 : asks the receiver account details
The user enters the receiver's account details.
3 : receiver details
4 : initiate transaction
5 : verify account details
6 : ineligible account
7 : transaction is unsuccessful
The transaction is not successful as one of the 2 accounts are not eligible for net banking.
8 : transaction is not successful

ALTERNATIVE FLOW 2: Insufficient funds or frozen account

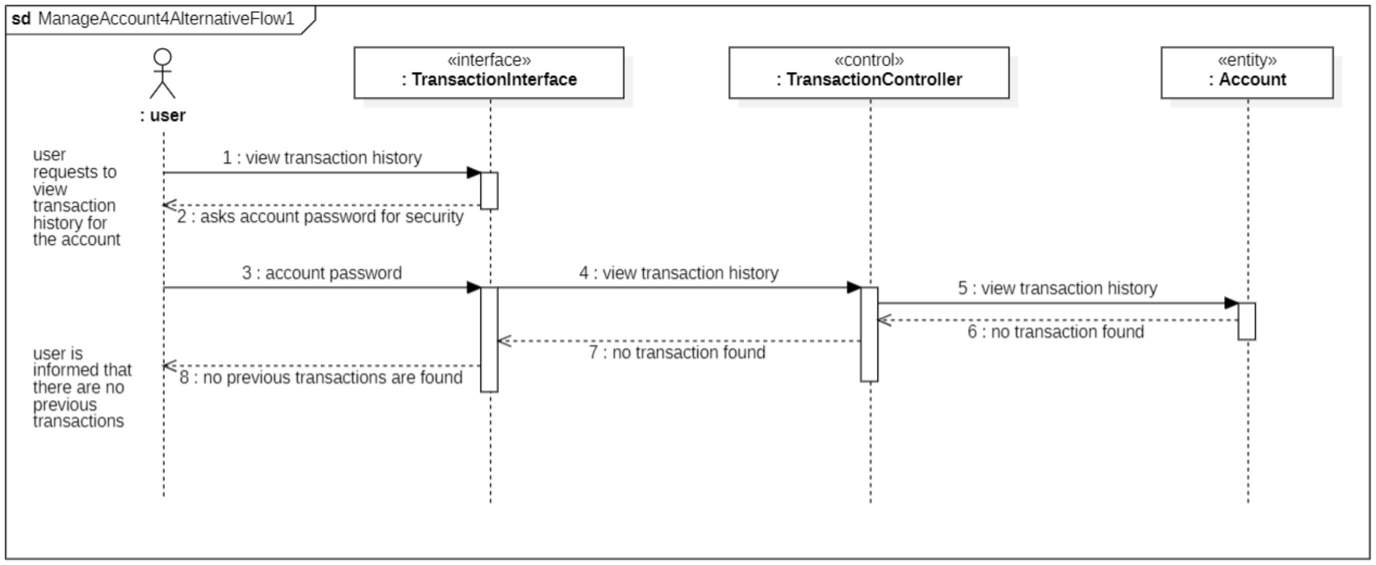

sd ManageAccount3AlternativeFlow2
: user
«interface» : TransactionInterface
«control» : TransactionController
«entity» : Account
«interface» : PaymentGateway
1 : initiates a transaction
2 : asks the receiver account details
3 : receiver details
4 : initiate transaction
5 : verify account details
6 : insufficient balance or frozen account
7 : transaction is unsuccessful
8 : transaction is not successful
The user Wants to initiate a transaction.
The user enters the receiver's account details.
The transaction is not successful as the user's account has insufficient balance or is frozen

4)View Transaction History:

BASIC FLOW

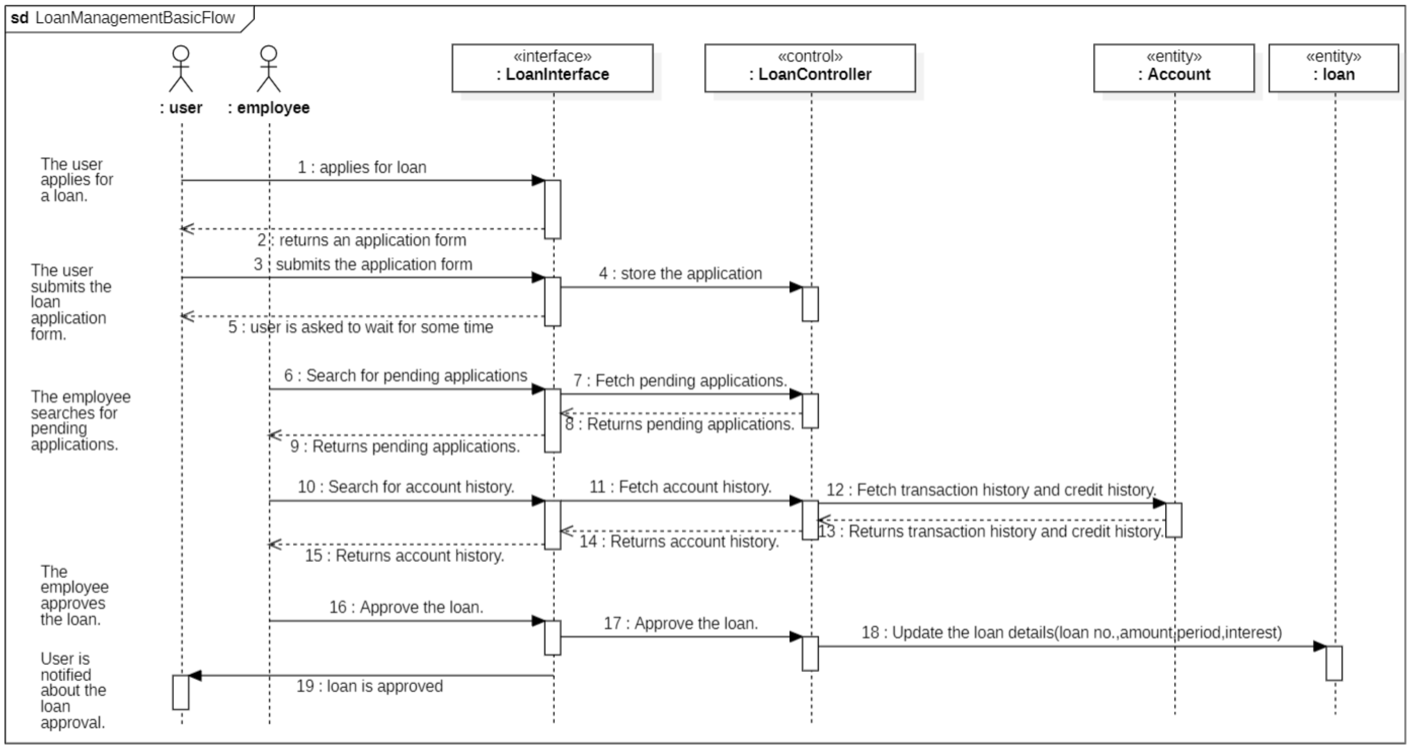

sd ManageAccount4BasicFlow
: user
«interface» : TransactionInterface
«control» : TransactionController
«entity» : Account
user requests to view transaction history for the account
1 : view transaction history
2 : asks account password for security
3 : account password
4 : view transaction history
5 : view transaction history
6 : details of previous transactions
7 : details of previous transactions
8 : details of previous transactions
details of previous transactions are returned to the user

ALTERNATIVE FLOW 1: No transaction found

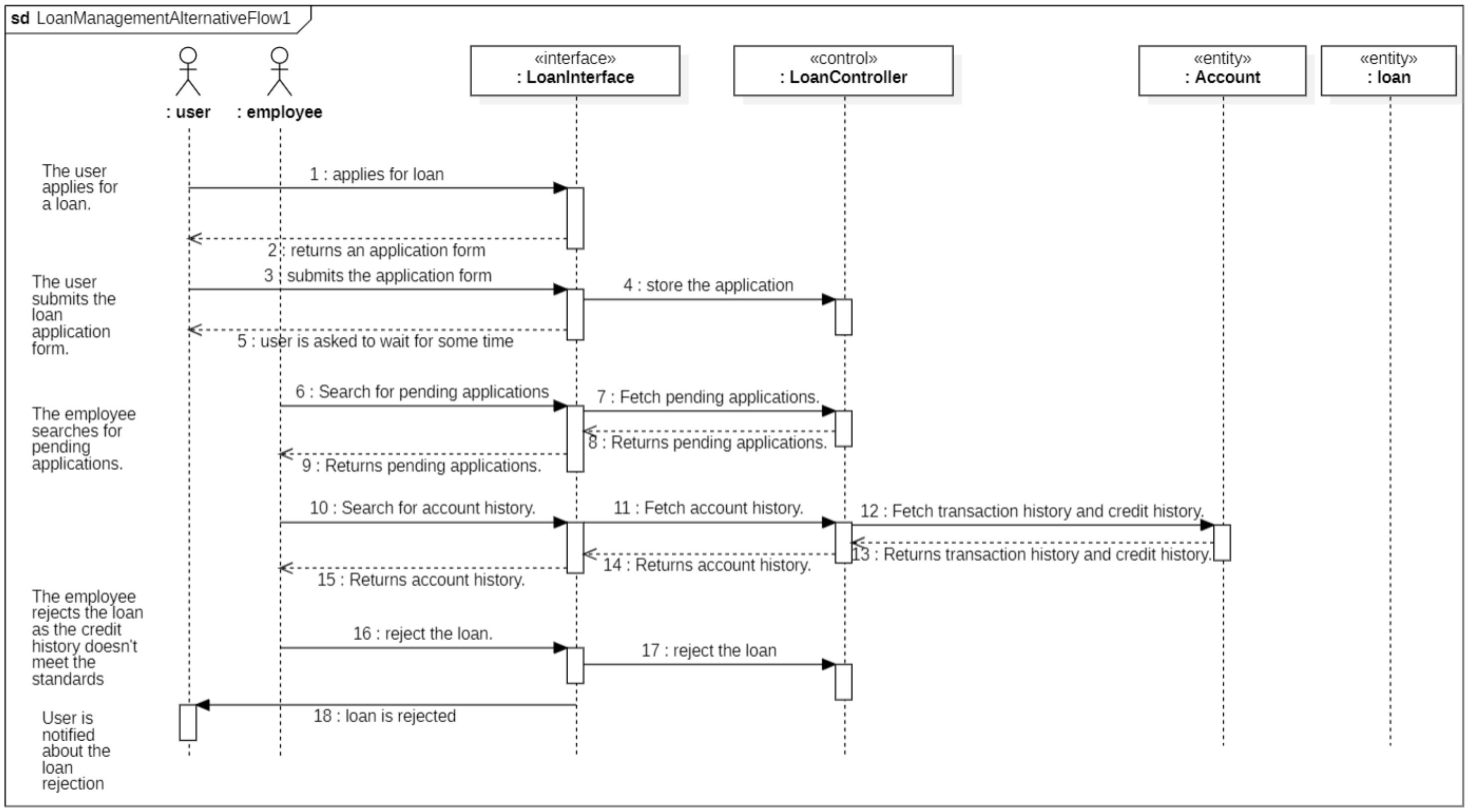

sd ManageAccount4AlternativeFlow1
: user
«interface» : TransactionInterface
«control» : TransactionController
«entity» : Account
user requests to view transaction history for the account
1 : view transaction history
2 : asks account password for security
3 : account password
4 : view transaction history
5 : view transaction history
6 : no transaction found
7 : no transaction found
8 : no previous transactions are found
user is informed that there are no previous transactions

5) Loan Management

BASIC FLOW

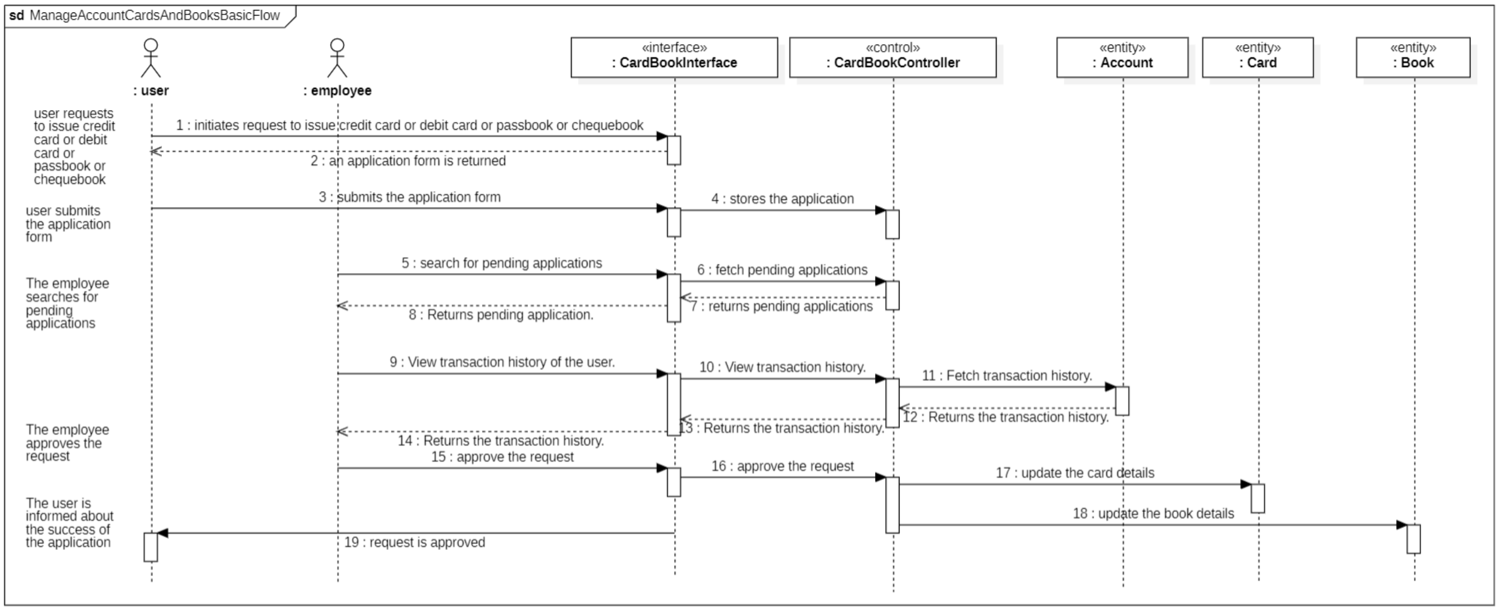

sd LoanManagementBasicFlow
: user
: employee
«interface» : LoanInterface
«control» : LoanController
«entity» : Account
«entity» : loan
The user applies for a loan.
1 : applies for loan
2 : returns an application form
The user submits the loan application form.
3 : submits the application form
4 : store the application
5 : user is asked to wait for some time
The employee searches for pending applications.
6 : Search for pending applications
7 : Fetch pending applications.
8 : Returns pending applications.
9 : Returns pending applications.
10 : Search for account history.
11 : Fetch account history.
12 : Fetch transaction history and credit history.
13 : Returns transaction history and credit history.
14 : Returns account history.
15 : Returns account history.
The employee approves the loan.
16 : Approve the loan.
17 : Approve the loan.
18 : Update the loan details(loan no.,amount,period,interest)
User is notified about the loan approval.
19 : loan is approved

ALTERNATIVE FLOW 1: Credit history not meeting standards

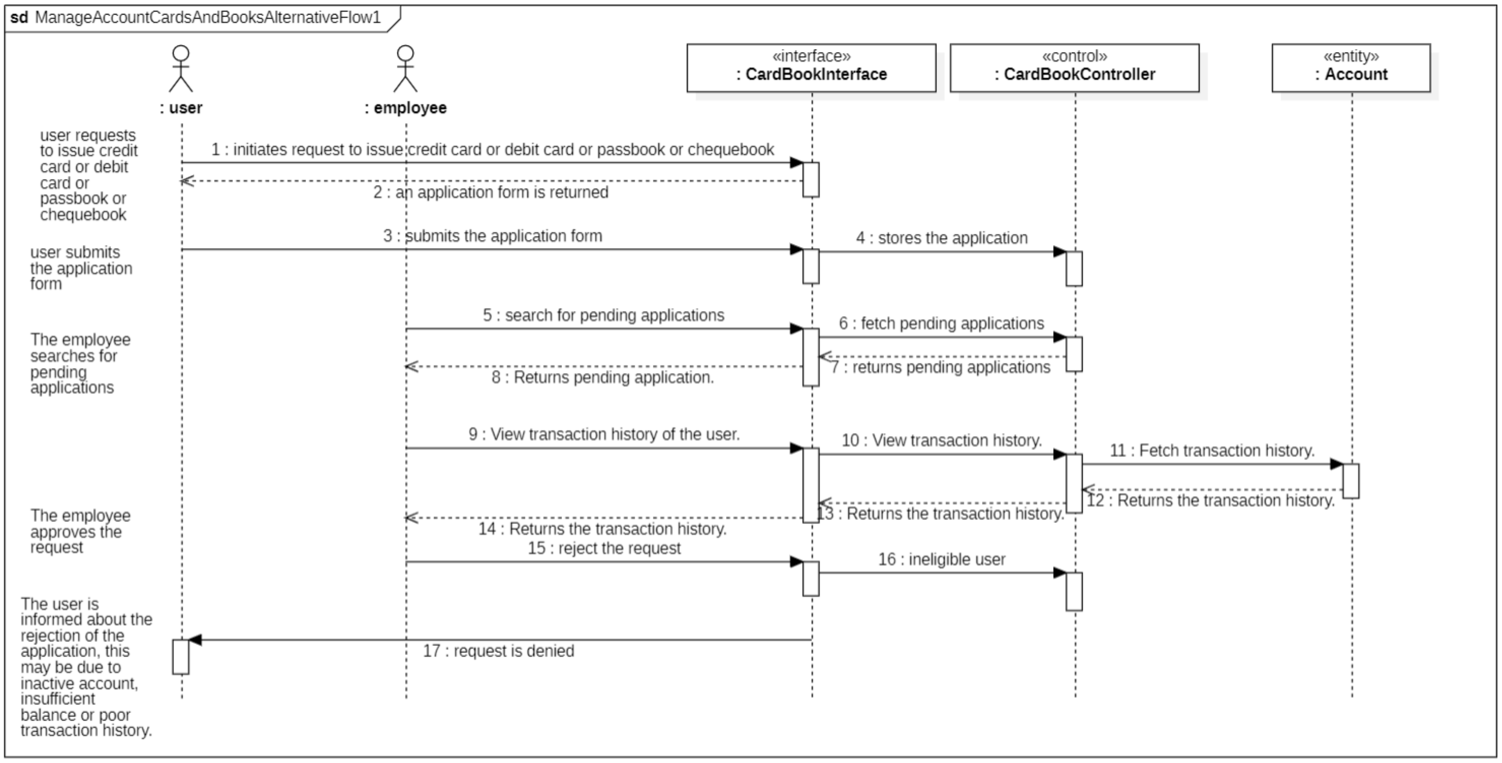

sd LoanManagementAlternativeFlow1
: user
: employee
«interface» : LoanInterface
«control» : LoanController
«entity» : Account
«entity» : loan
The user applies for a loan.
1 : applies for loan
2 : returns an application form
The user submits the loan application form.
3 : submits the application form
4 : store the application
5 : user is asked to wait for some time
The employee searches for pending applications.
6 : Search for pending applications
7 : Fetch pending applications.
8 : Returns pending applications.
9 : Returns pending applications.
10 : Search for account history.
11 : Fetch account history.
12 : Fetch transaction history and credit history.
13 : Returns transaction history and credit history.
14 : Returns account history.
15 : Returns account history.
The employee rejects the loan as the credit history doesn't meet the standards
16 : reject the loan.
17 : reject the loan
User is notified about the loan rejection
18 : loan is rejected

6) Manage account cards and books

BASIC FLOW

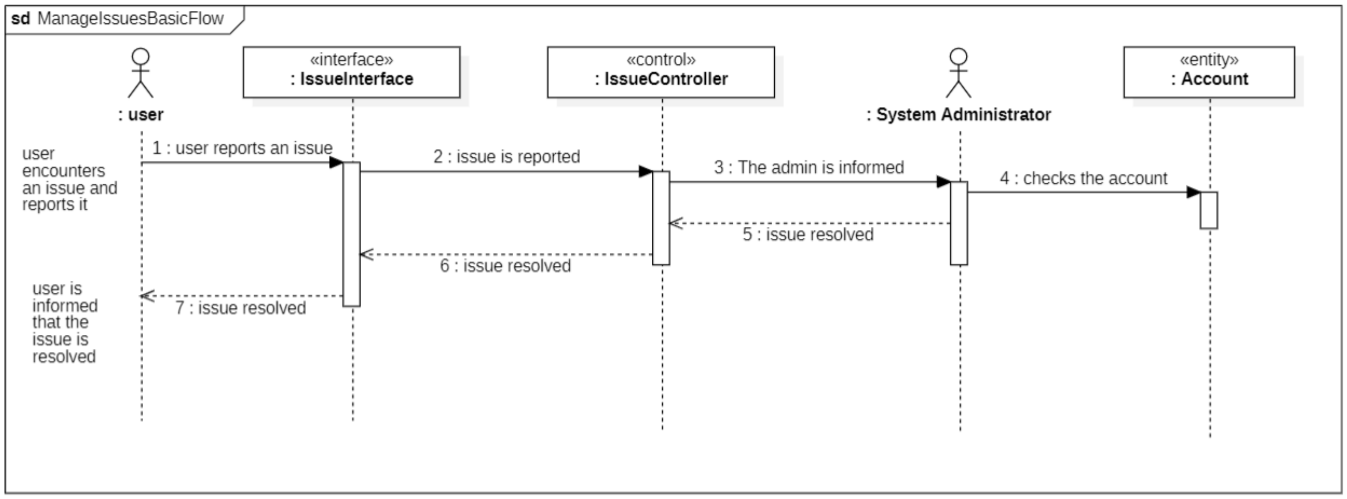

sd ManageAccountCardsAndBooksBasicFlow
: user
: employee
«interface» : CardBookInterface
«control» : CardBookController
«entity» : Account
«entity» : Card
«entity» : Book
user requests to issue credit card or debit card or passbook or chequebook
1 : initiates request to issue credit card or debit card or passbook or chequebook
2 : an application form is returned
user submits the application form
3 : submits the application form
4 : stores the application
The employee searches for pending applications
5 : search for pending applications
6 : fetch pending applications
7 : returns pending applications
8 : Returns pending application.
9 : View transaction history of the user.
10 : View transaction history.
11 : Fetch transaction history.
12 : Returns the transaction history.
13 : Returns the transaction history.
The employee approves the request
14 : Returns the transaction history.
15 : approve the request
16 : approve the request
17 : update the card details
18 : update the book details
The user is informed about the success of the application
19 : request is approved

ALTERNATIVE FLOW 1: Ineligible user

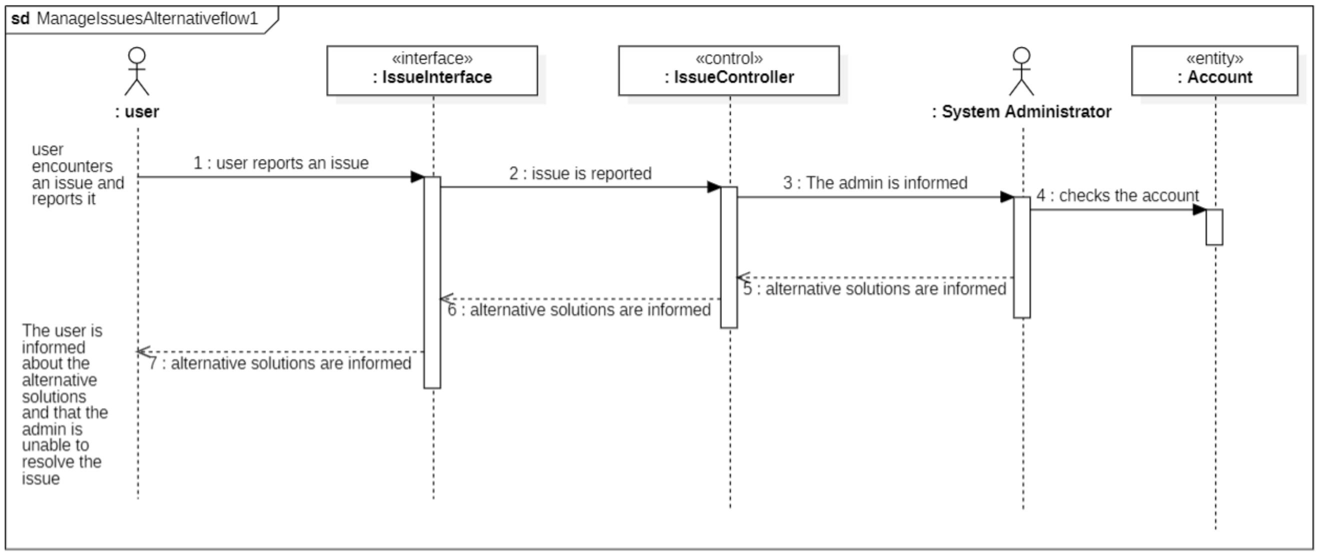

sd ManageAccountCardsAndBooksAlternativeFlow1
: user
: employee
«interface» : CardBookInterface
«control» : CardBookController
«entity» : Account
user requests to issue credit card or debit card or passbook or chequebook
1 : initiates request to issue credit card or debit card or passbook or chequebook
2 : an application form is returned
user submits the application form
3 : submits the application form
4 : stores the application
The employee searches for pending applications
5 : search for pending applications
6 : fetch pending applications
7 : returns pending applications
8 : Returns pending application.
9 : View transaction history of the user.
10 : View transaction history.
11 : Fetch transaction history.
12 : Returns the transaction history.
13 : Returns the transaction history.
The employee approves the request
14 : Returns the transaction history.
15 : reject the request
16 : ineligible user
The user is informed about the rejection of the application, this may be due to inactive account, insufficient balance or poor transaction history.
17 : request is denied

<u>7) Manage Issues</u>

BASIC FLOW

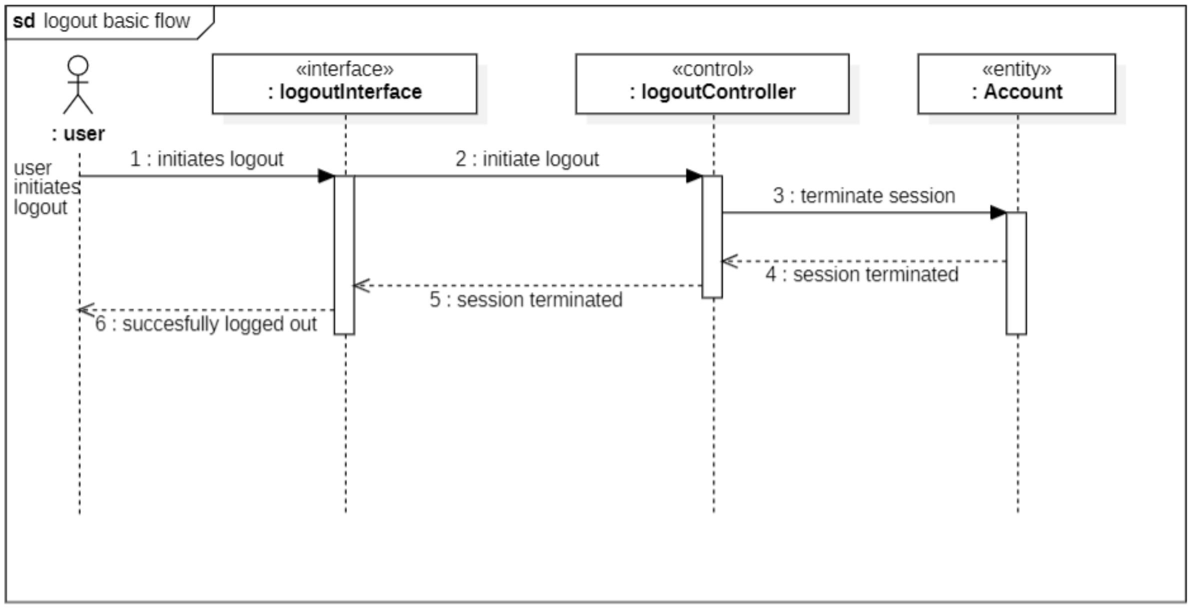

sd ManageIssuesBasicFlow
: user
«interface» : IssueInterface
«control» : IssueController
: System Administrator
«entity» : Account
user encounters an issue and reports it
1 : user reports an issue
2 : issue is reported
3 : The admin is informed
4 : checks the account
5 : issue resolved
6 : issue resolved
7 : issue resolved
user is informed that the issue is resolved

ALTERNATIVE FLOW 1: Administrator unable to solve the issue

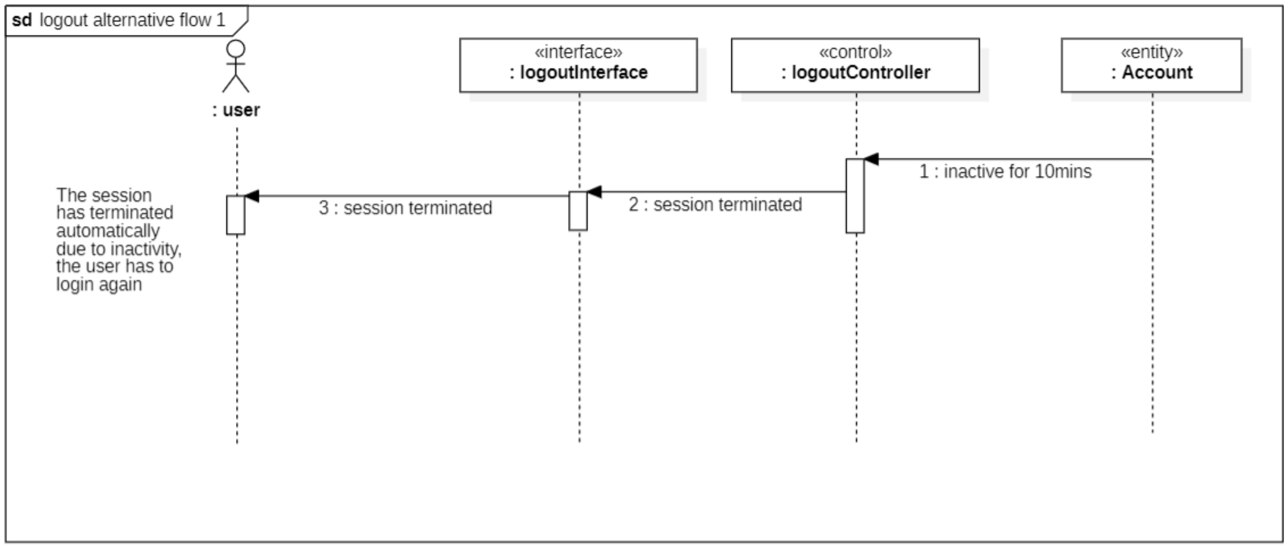

sd ManageIssuesAlternativeflow1
: user
«interface» : IssueInterface
«control» : IssueController
: System Administrator
«entity» : Account
user encounters an issue and reports it
1 : user reports an issue
2 : issue is reported
3 : The admin is informed
4 : checks the account
5 : alternative solutions are informed
6 : alternative solutions are informed
7 : alternative solutions are informed
The user is informed about the alternative solutions and that the admin is unable to resolve the issue

8) Logout

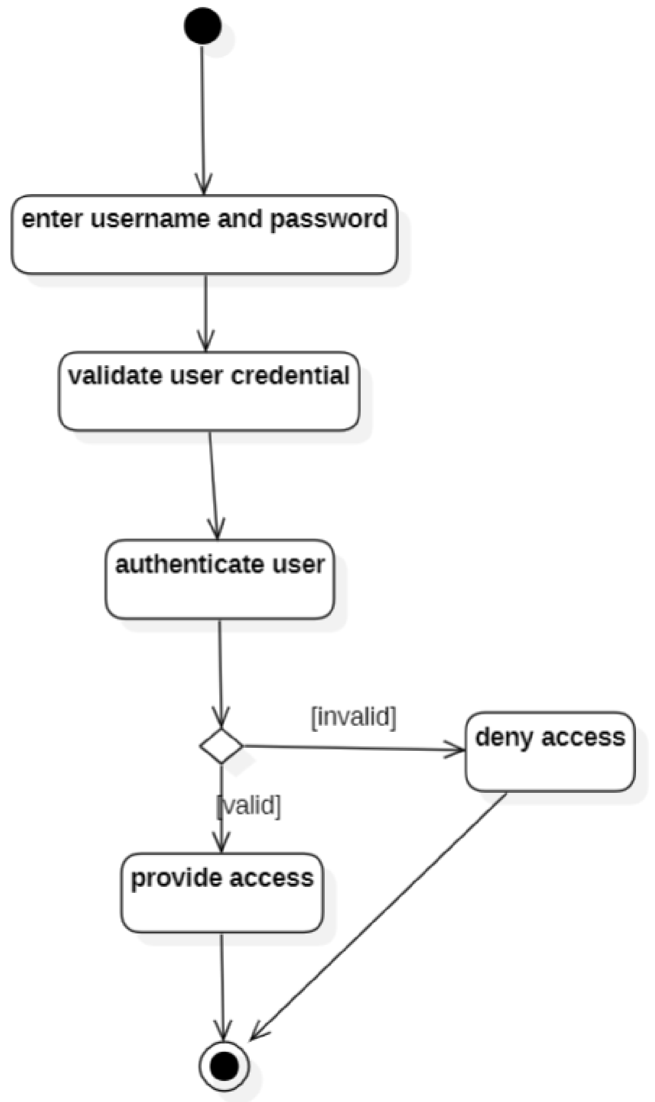
BASIC FLOW



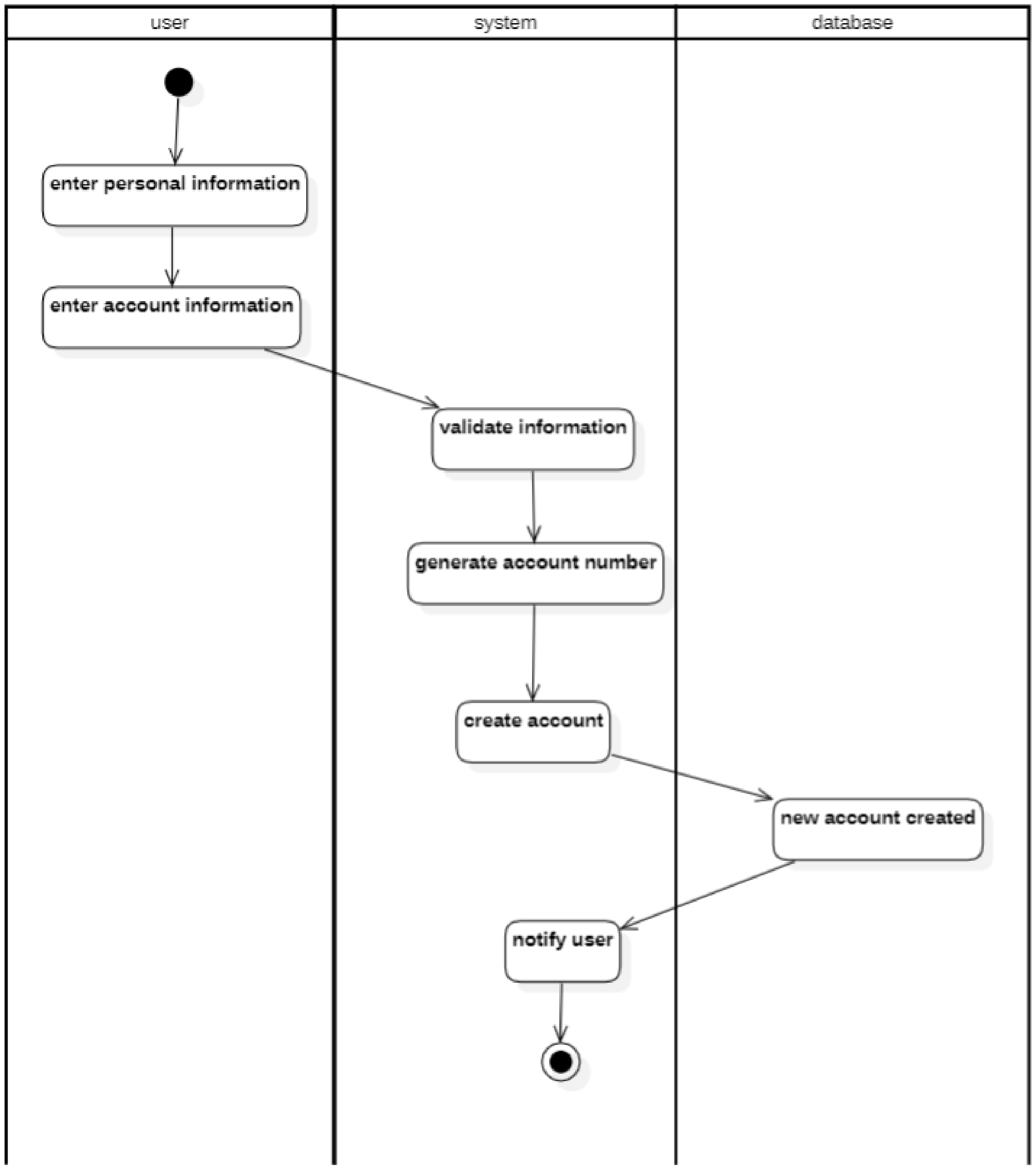
ALTERNATIVE FLOW 1: Automatic logout/session expired


sd logout alternative flow 1
: user
«interface» : logoutInterface
«control» : logoutController
«entity» : Account
1 : inactive for 10mins
2 : session terminated
3 : session terminated
The session has terminated automatically due to inactivity, the user has to login again

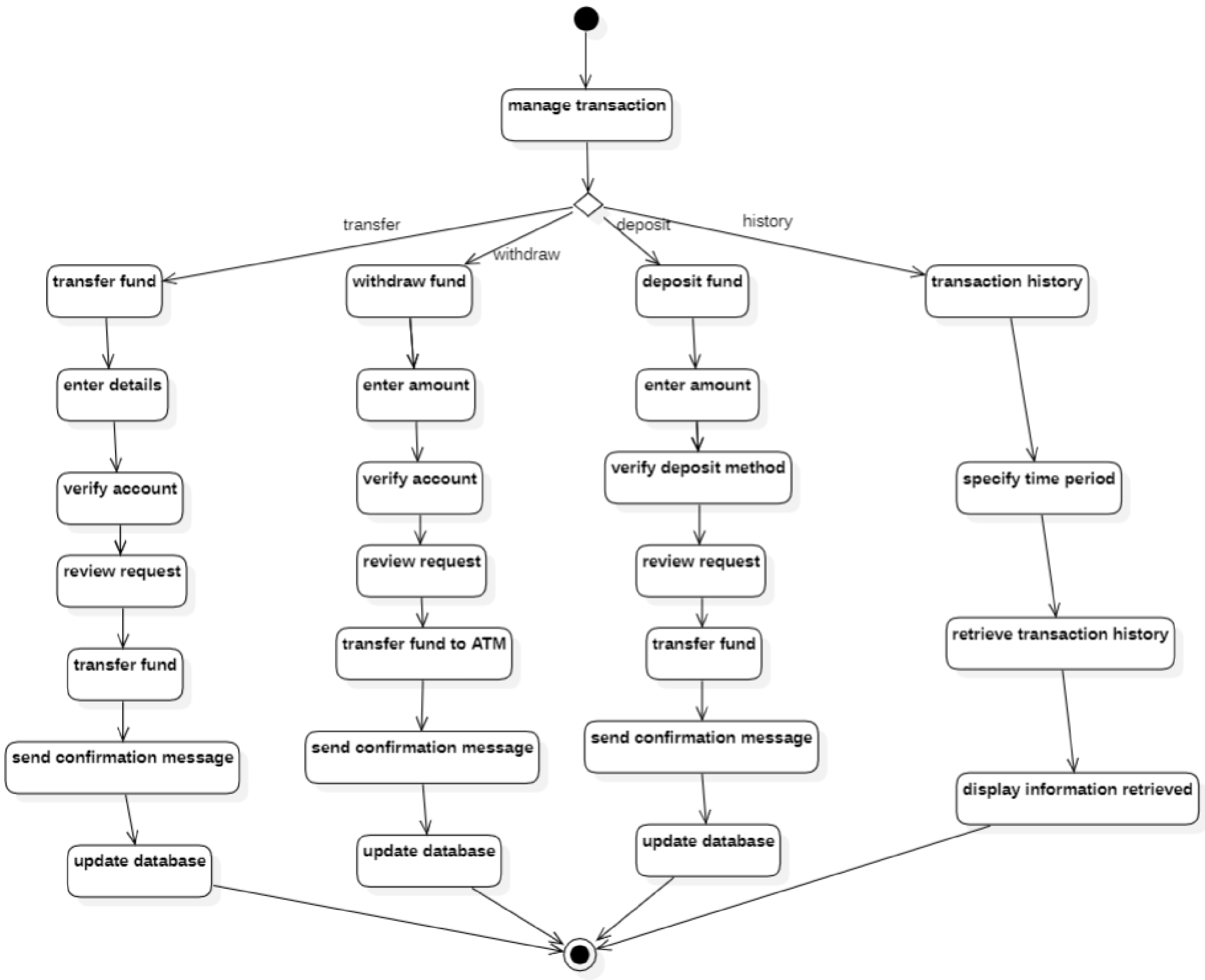
ACTIVITY DIAGRAM



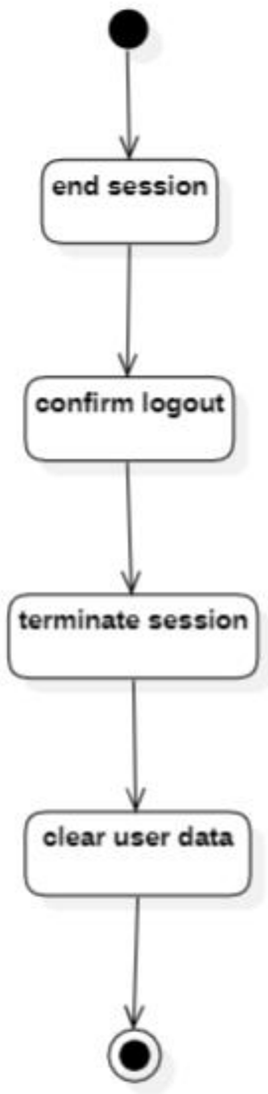
Activity Diagram of "Login" use case



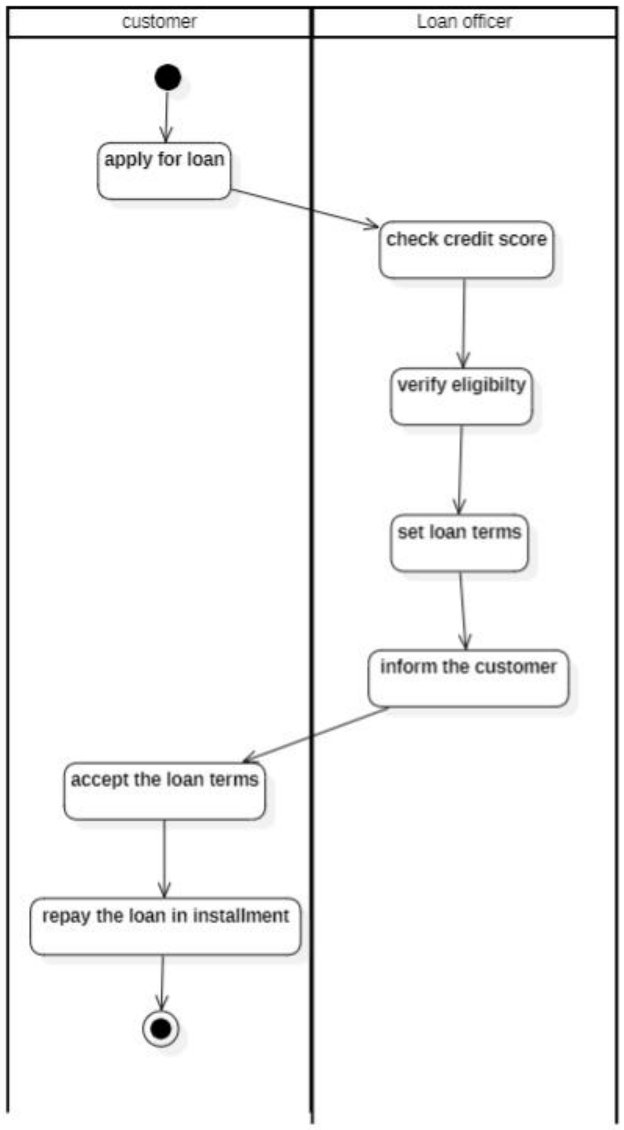
Activity diagram of "Register" use case



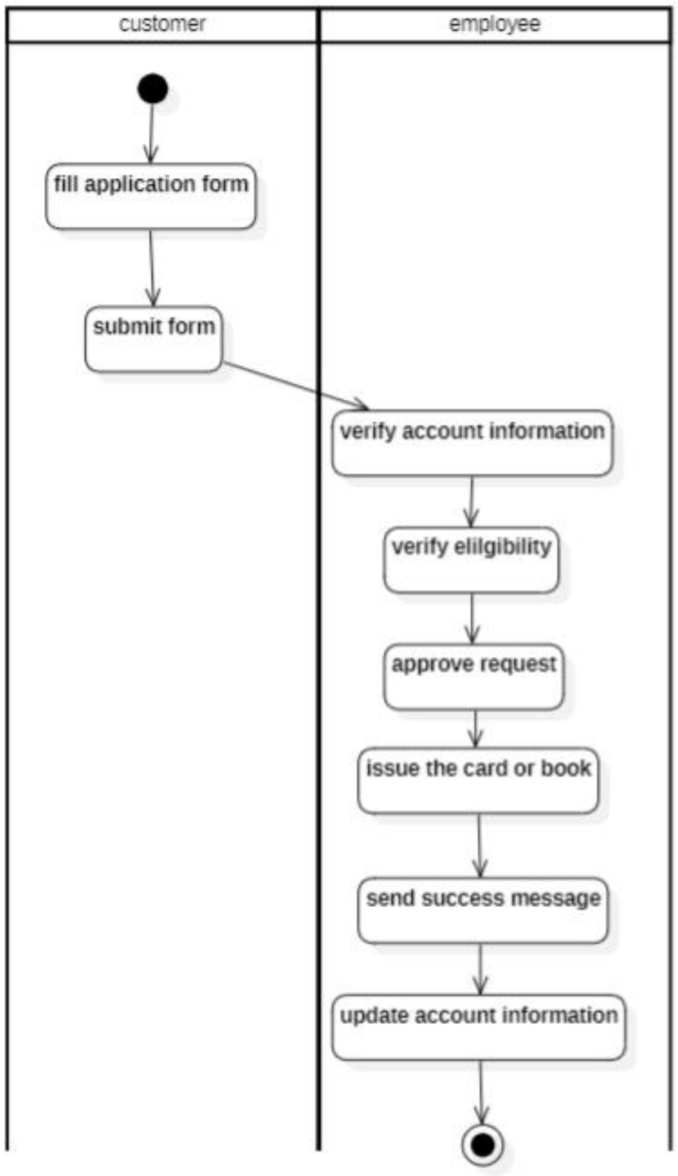
Activity diagram of "Manage Transactions" use case



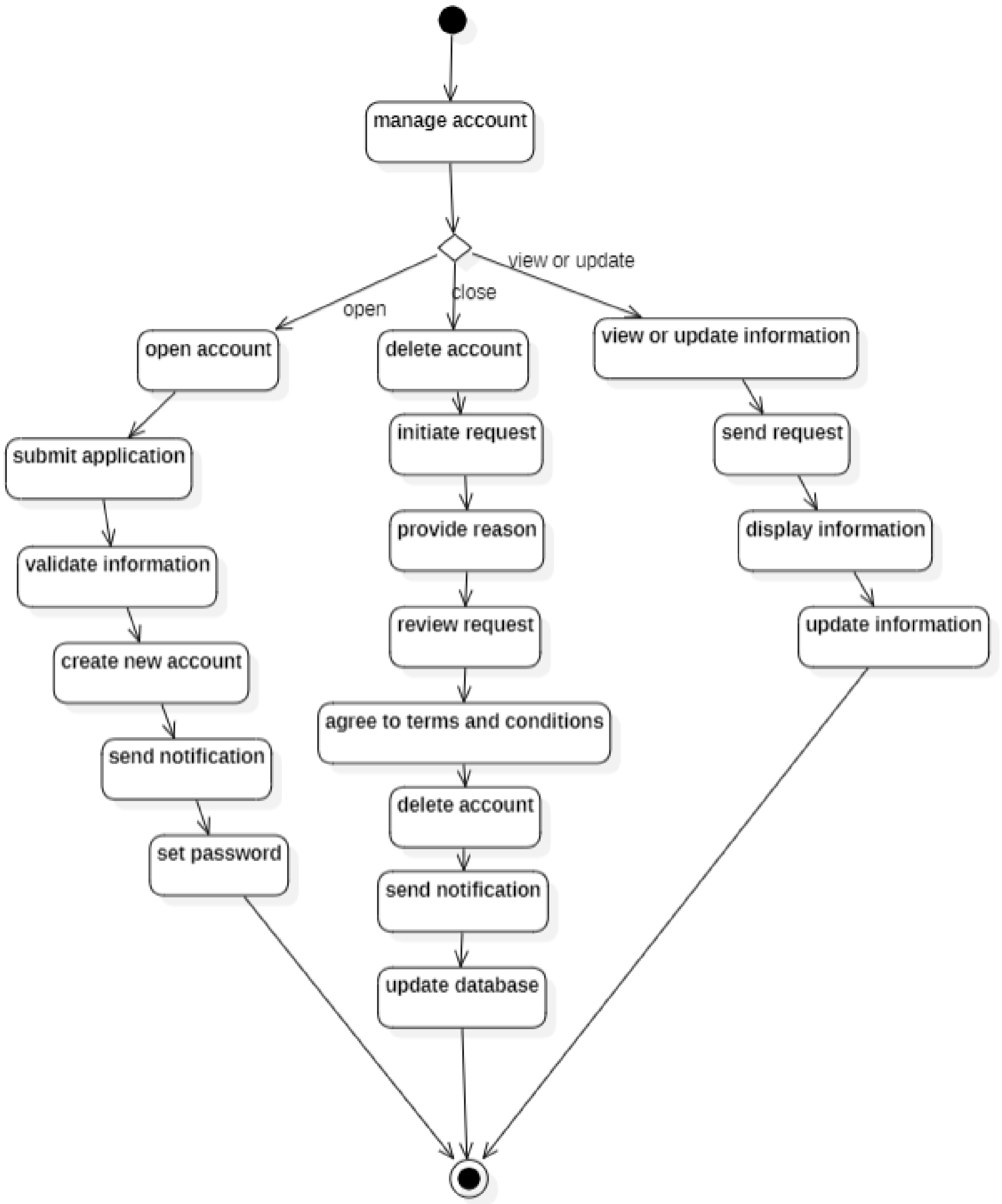
Activity diagram of "Logout" use case



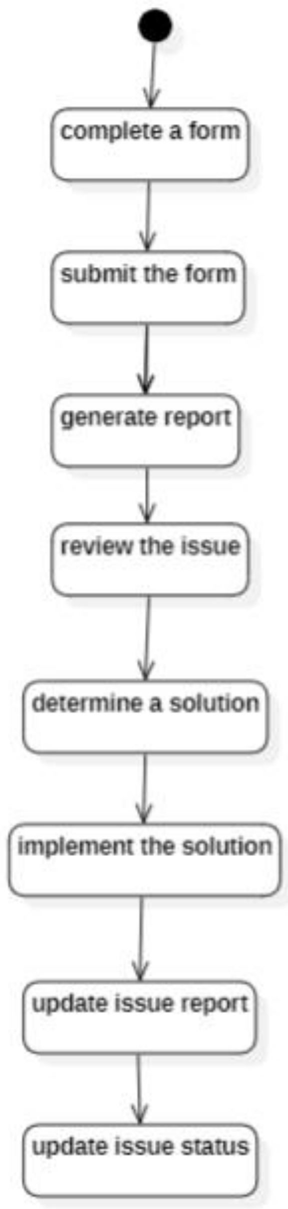
Activity diagram of "Loan Management" use case



Activity diagram of "Manage account books and cards" use case

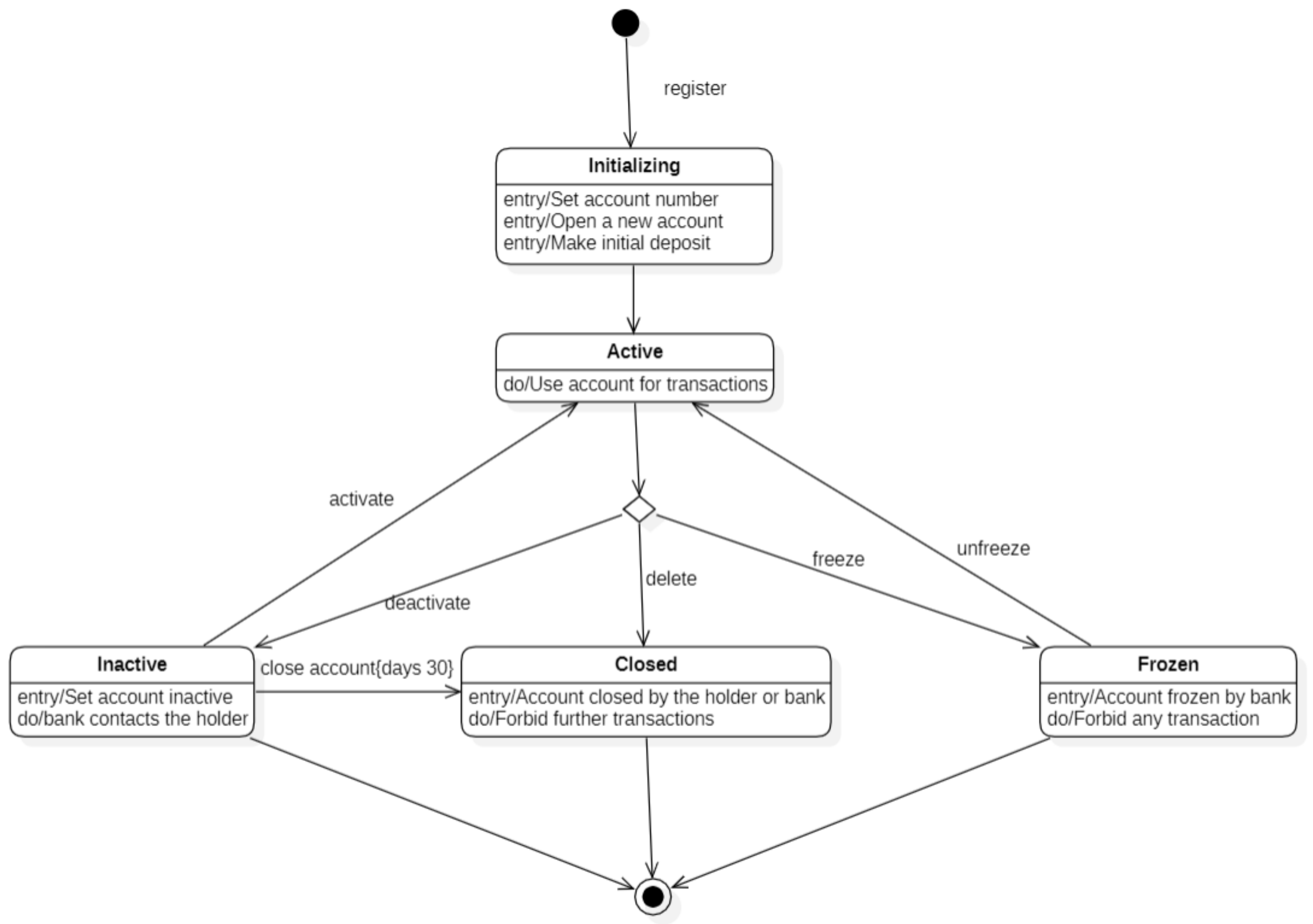


Activity diagram of "Manage Account" use case

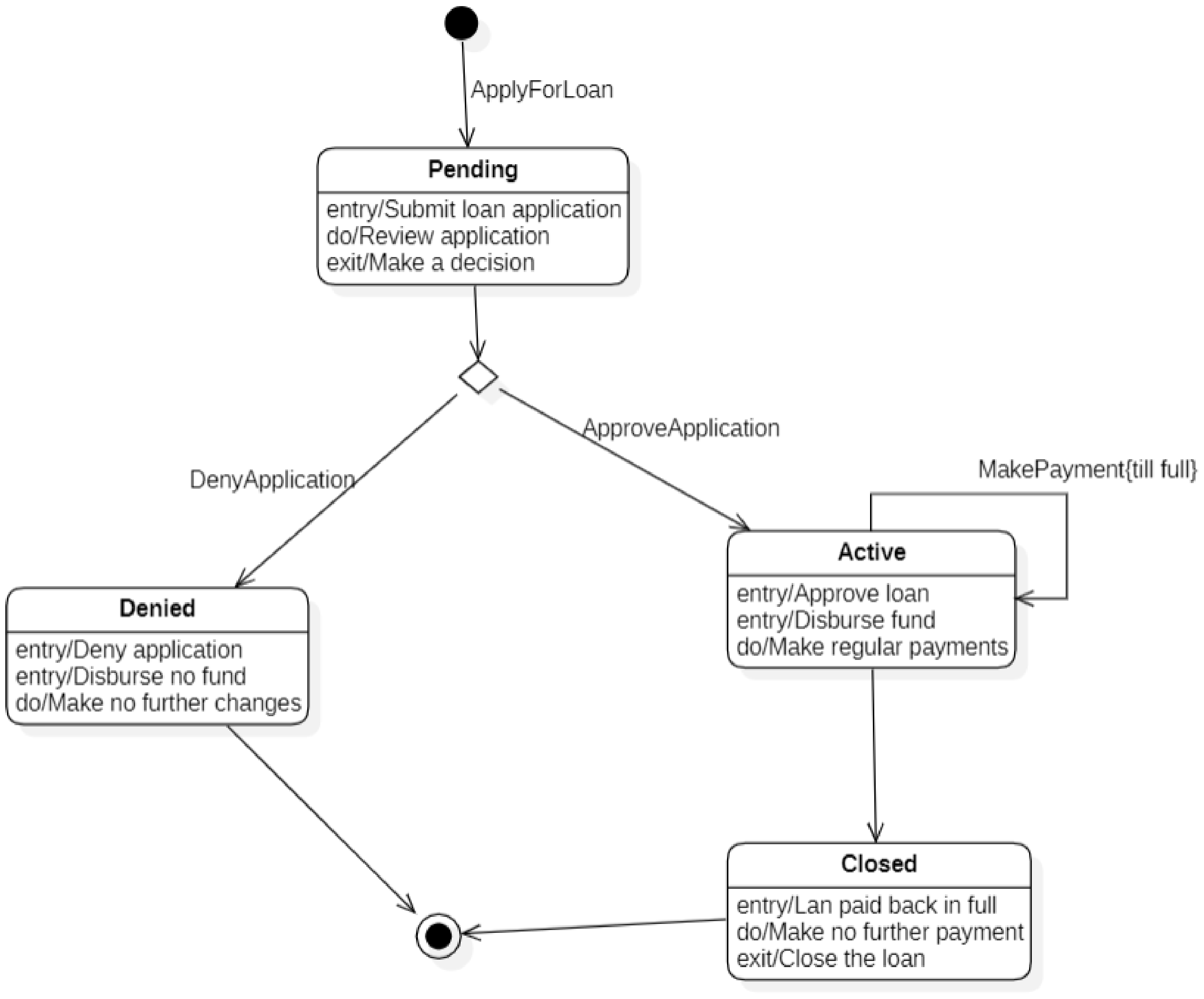


Activity diagram of "Manage Issues" use case

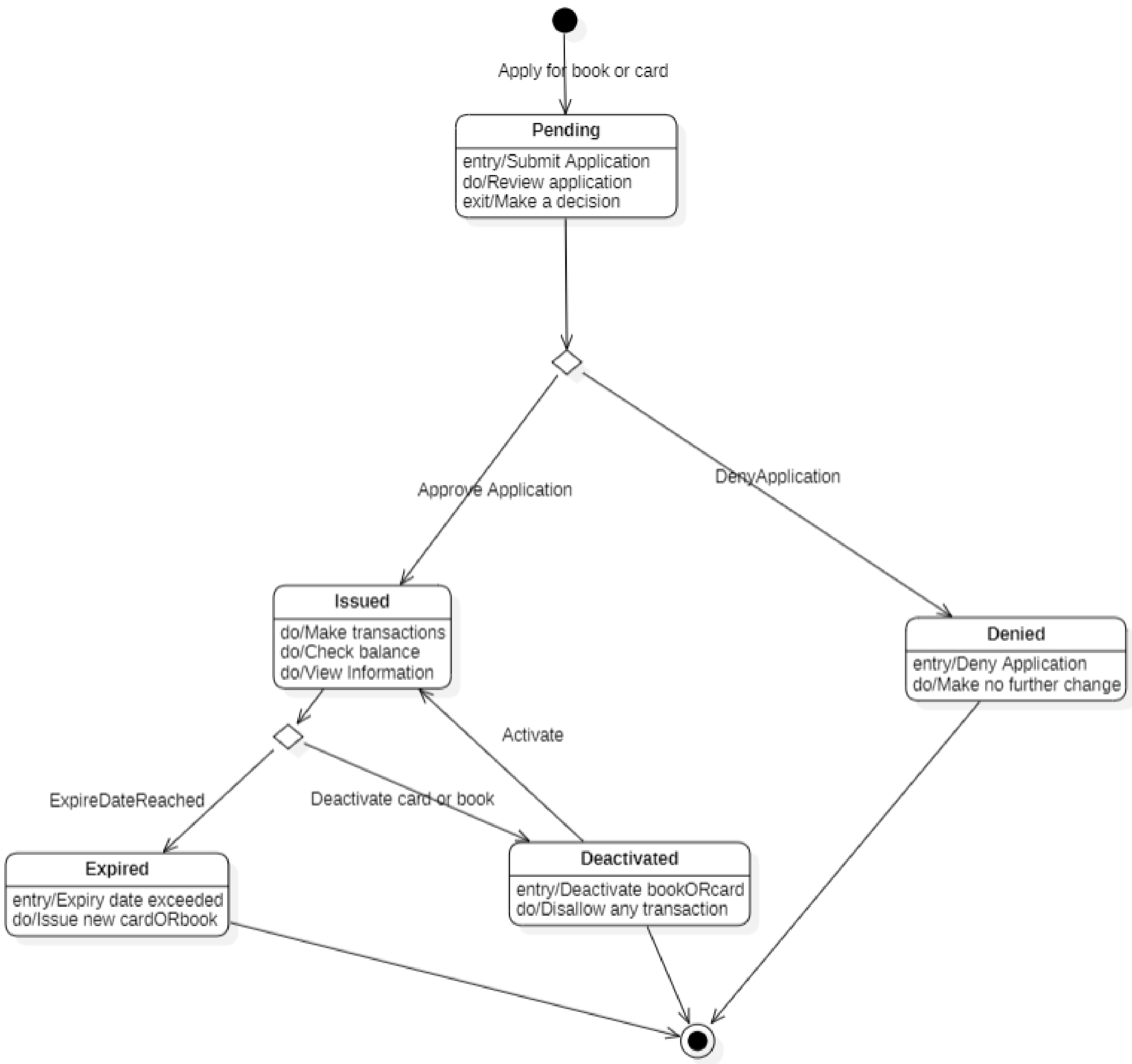
STATECHART DIAGRAM


register
Initializing
entry/Set account number
entry/Open a new account
entry/Make initial deposit
Active
do/Use account for transactions
activate
unfreeze
freeze
delete
deactivate
Inactive
entry/Set account inactive
do/bank contacts the holder
close account{days 30}
Closed
entry/Account closed by the holder or bank
do/Forbid further transactions
Frozen
entry/Account frozen by bank
do/Forbid any transaction

Statechart Diagram of Account and member classes



Statechart diagram of Loan class



Statechart Diagram of Book and Card classes

TEST CASE MATRIX

Test case matrix for Register use case

Test Case ID	Test Case Description	Input 1: Name (Valid/Invalid)	Input 2: Address (Valid/Invalid)	Input 3: Date of Birth (Valid/Invalid)	Expected Output(s)	Remark(s)
TC01	Register with all valid inputs	Valid	Valid	Valid	Registration successful with user ID generated	The test case passed successfully
TC02	Register with invalid name	Invalid	Valid	Valid	Error message displayed: Invalid name format	The test case passed successfully
TC03	Register with invalid address	Valid	Invalid	Valid	Error message displayed: Invalid address format	The test case passed successfully
TC04	Register with invalid date of birth	Valid	Valid	Invalid	Error message displayed: Invalid date of birth format	The test case passed successfully
TC05	Register with invalid name and address	Invalid	Invalid	Valid	Error message displayed: Invalid name and address format	The test case passed successfully
TC06	Register with invalid name and date of birth	Invalid	Valid	Invalid	Error message displayed: Invalid name and date of birth format	The test case passed successfully
TC07	Register	Valid	Invalid	Invalid	Error	The test

	with invalid address and date of birth				message displayed: Invalid address and date of birth format	case passed successfully
TC08	Register with invalid name, address and date of birth	Invalid	Invalid	Invalid	Error message displayed: Invalid name, address and date of birth format	The test case passed successfully
TC09	Register with existing name and address	Valid	Valid	Valid	Error message displayed: User with the same name and address already exists	The test case passed successfully
TC10	Register with existing name but different address	Valid	Valid	Valid	Registration successful with user ID generated	The test case passed successfully

Test case matrix with actual values for Register use case

Test Case ID	Test Case Description	Input 1: Name	Input 2: Address	Input 3: Date of Birth	Expected Output(s)	Remark(s)
TC01	Register with all valid inputs	John Smith	123 Main Street, Anytown, USA	01/01/1980	Registration successful with user ID generated	The test case passed successfully
TC02	Register with invalid name	John123	123 Main Street, Anytown, USA	01/01/1980	Error message displayed: Invalid name format	The test case passed successfully

TC03	Register with invalid address	John Smith	123 Main Street, #456, Anytown, USA	01/01/1980	Error message displayed: Invalid address format	The test case passed successfully
TC04	Register with invalid date of birth	John Smith	123 Main Street, Anytown, USA	01/01/20XX	Error message displayed: Invalid date of birth format	The test case passed successfully
TC05	Register with invalid name and address	John123	123 Main Street, #456, Anytown, USA	01/01/1980	Error message displayed: Invalid name and address format	The test case passed successfully
TC06	Register with invalid name and date of birth	John123	123 Main Street, Anytown, USA	01/01/20XX	Error message displayed: Invalid name and date of birth format	The test case passed successfully
TC07	Register with invalid address and date of birth	John Smith	123 Main Street, #456, Anytown, USA	01/01/20XX	Error message displayed: Invalid address and date of birth format	The test case passed successfully
TC08	Register with invalid name, address and date of birth	John123	123 Main Street, #456, Anytown, USA	01/01/20XX	Error message displayed: Invalid name, address and date of birth format	The test case passed successfully
TC09	Register with existing name and address	John Smith	123 Main Street, Anytown, USA	01/01/1980	Error message displayed: User with the same name and address already exists	The test case passed successfully
TC10	Register with existing name but different address	John Smith	456 Oak Street, Anytown, USA	01/01/1980	Registration successful with user ID generated	The test case passed successfully

Test case matrix for Manage account use case

Test Case ID	Test Case Description	Account Type (Valid/Invalid)	Customer Name (Valid/Invalid)	Initial Deposit (Valid/Invalid)	Account Number (Valid/Invalid)	Expected Output(s)	Remark(s)
TC01	Open Account with valid input	Valid	Valid	Valid	Generated by system	Account successfully created with Account Number generated and Initial Deposit credited	The test case passed successfully
TC02	Open Account with invalid input	Invalid	Valid	Invalid	N/A	Error message displayed: Invalid account type/initial deposit amount	The test case passed successfully
TC03	Delete Account with valid input	N/A	N/A	N/A	Valid	Account successfully deleted and remaining balance refunded to customer	The test case passed successfully
TC04	Delete Account with invalid input	N/A	N/A	N/A	Invalid	Error message displayed: Account does not exist	The test case passed successfully
TC05	View Account Detail with valid input	N/A	N/A	N/A	Valid	Account details displayed: Account	The test case passed successfully

						Type, Account Number, Customer Name, Account Balance	
TC06	View Account Detail with invalid input	N/A	N/A	N/A	Invalid	Error message displayed: Account does not exist	The test case passed successfully
TC07	Update Account Detail with valid input	N/A	Valid	N/A	Valid	Account details updated successfully	The test case passed successfully
TC08	Update Account Detail with invalid input	N/A	Invalid	N/A	Invalid	Error message displayed: Invalid customer name	The test case passed successfully
TC09	Open Account with existing customer name	Valid	Valid	Valid	Generated by system	Account successfully created with Account Number generated and Initial Deposit credited	The test case passed successfully
TC10	Update Account Detail with non-existing account number	N/A	Valid	N/A	Invalid	Error message displayed: Account does not exist	The test case passed successfully

Test case matrix with actual values for Manage account use case

Test Case ID	Test Case Description	Input 1 Account Type	Input 2 Customer Name	Input 3 Initial Deposit	Input 4 Account Number	Expected Output(s)	Remark(s)
TC01	Open Account with valid input	Savings	John Doe	$500	Generated by system	Account successfully created with Account Number generated and Initial Deposit credited	The test case passed successfully
TC02	Open Account with invalid input	Current	John Doe	$0	N/A	Error message displayed: Invalid account type/initial deposit amount	The test case passed successfully
TC03	Delete Account with valid input	N/A	N/A	N/A	12345	Account successfully deleted and remaining balance refunded to customer	The test case passed successfully
TC04	Delete Account with invalid input	N/A	N/A	N/A	99999	Error message displayed: Account does not exist	The test case passed successfully
TC05	View Account Detail with valid input	N/A	N/A	N/A	12345	Account details displayed: Account Type, Account Number, Customer Name, Account Balance	The test case passed successfully

TC06	View Account Detail with invalid input	N/A	N/A	N/A	88888	Error message displayed: Account does not exist	The test case passed successfully
TC07	Update Account Detail with valid input	N/A	Jane Doe	N/A	12345	Account details updated successfully	The test case passed successfully
TC08	Update Account Detail with invalid input	N/A	""	N/A	54321	Error message displayed: Invalid customer name	The test case passed successfully
TC09	Open Account with existing customer name	Current	John Doe	$1000	Generated by system	Account successfully created with Account Number generated and Initial Deposit credited	The test case passed successfully
TC10	Update Account Detail with non-existing account number	N/A	Jane Doe	N/A	55555	Error message displayed: Account does not exist	The test case passed successfully

Test case matrix for Login use case

Test Case ID	**Test Case Description**	**Input 1: Username (valid/invalid)**	**Input 2: Password (valid/invalid)**	**Expected Output(s)**	**Remark(s)**
TC01	Login with valid username and password	valid	valid	Login successful	The test case passed successfully

TC02	Login with valid username and invalid password	valid	invalid	Error message displayed: Incorrect password	The test case passed successfully
TC03	Login with invalid username and valid password	invalid	valid	Error message displayed: Username not found	The test case passed successfully
TC04	Login with empty username and password fields	invalid	invalid	Error message displayed: Username and password required	The test case passed successfully
TC05	Login with valid username and empty password field	valid	invalid	Error message displayed: Password required	The test case passed successfully
TC06	Login with empty username field and valid password	invalid	valid	Error message displayed: Username required	The test case passed successfully
TC07	Login with invalid username and invalid password	invalid	invalid	Error message displayed: Username not found and incorrect password	The test case passed successfully
TC08	Login with SQL injection in username field	invalid	valid	Error message displayed: Username not found	The test case passed successfully

TC09	Login with XSS attack in username field	invalid	valid	Error message displayed: Username not found	The test case passed successfully
TC10	Login with special characters in username and password fields	valid	valid	Login successful	The test case passed successfully

Test case matrix with actual values for Login use case

Test Case ID	Test Case Description	Input 1: Username	Input 2: Password	Expected Output(s)	Remark(s)
TC01	Login with valid username and password	johndoe	password123	Login successful	The test case passed successfully
TC02	Login with valid username and invalid password	johndoe	pass123	Error message displayed: Incorrect password	The test case passed successfully
TC03	Login with invalid username and valid password	janedoe	password123	Error message displayed: Username not found	The test case passed successfully
TC04	Login with empty username and password fields			Error message displayed: Username and password required	The test case passed successfully

TC05	Login with valid username and empty password field	johndoe		Error message displayed: Password required	The test case passed successfully
TC06	Login with empty username field and valid password		password123	Error message displayed: Username required	The test case passed successfully
TC07	Login with invalid username and invalid password	janedoe	pass123	Error message displayed: Username not found and incorrect password	The test case passed successfully
TC08	Login with SQL injection in username field	' OR 1=1;--	password123	Error message displayed: Username not found	The test case passed successfully
TC09	Login with XSS attack in username field	<script>alert('XSS')</script>	password123	Error message displayed: Username not found	The test case passed successfully
TC10	Login with special characters in username and password fields	john#doe	p@ssw0rd!	Login successful	The test case passed successfully

Test case matrix for Manage transactions use case

Test Case ID	Test Case Description	Input 1 - Name	Input 2 - Account Number	Input 3 - IFSC Code	Input 4 - Amount	Expected Result	Remarks
TC01	Verify user can deposit money into their account	N/A	N/A	N/A	Valid	User's account balance increases.	The deposit function worked as expected, and the user's account balance increased by the correct amount.
TC02	Verify user can withdraw money from their account	N/A	N/A	N/A	Valid	User's account balance decreases.	The withdrawal function worked as expected, and the user's account balance decreased by the correct amount.
TC03	Verify user can transfer money to another account	Valid	Valid	Valid	Valid	Sender's account balance decreases, and recipient's account balance increases.	The transfer function worked as expected, and both the sender's and recipient's account balances were updated correctly.
TC04	Verify user can view their transaction history	N/A	N/A	N/A	N/A	Transaction history is displayed correctly.	The transaction history was displayed correctly, and all transactions were listed in the correct order.
TC05	Verify user cannot deposit a negative amount	N/A	N/A	N/A	Invalid	System prevents user from depositing a negative amount and displays appropriate error message.	The system correctly prevented the user from depositing a negative amount and displayed the appropriate error message.

TC06	Verify user cannot withdraw more than their available balance	N/A	N/A	N/A	Invalid	System prevents user from withdrawing more than their available balance and displays appropriate error message.	The system correctly prevented the user from withdrawing more than their available balance and displayed the appropriate error message.
TC07	Verify user cannot transfer more than their available balance	Valid	Valid	Valid	Invalid	System prevents user from transferring more than their available balance and displays appropriate error message.	The system correctly prevented the user from transferring more than their available balance and displayed the appropriate error message.
TC08	Verify user cannot transfer to an invalid account number	Valid	Invalid	Valid	Valid	System prevents user from transferring to an invalid account number and displays appropriate error message.	The system correctly prevented the user from transferring to an invalid account number and displayed the appropriate error message.
TC09	Verify user cannot transfer with an invalid IFSC code	Valid	Valid	Invalid	Valid	System prevents user from transferring with an invalid IFSC code and displays appropriate error message.	The system correctly prevented the user from transferring with an invalid IFSC code and displayed the appropriate error message.

Test case matrix with actual values
for Manage transactions use case

Test Case ID	Test Case Description	Input 1 - Name	Input 2 - Account Number	Input 3 - IFSC Code	Input 4- Amount	Expected Result	Remarks
TC01	Verify user can deposit money into their account	N/A	N/A	N/A	Amount: $100	User's account balance increases by $100.	The deposit function worked as expected, and the user's account balance increased by the correct amount.
TC02	Verify user can withdraw money from their account	N/A	N/A	N/A	Amount: $50	User's account balance decreases by $50.	The withdrawal function worked as expected, and the user's account balance decreased by the correct amount.
TC03	Verify user can transfer money to another account	Recipient Name: Jane Doe	Recipient Account Number: 123456789	IFSC Code: ABCD123456	Amount: $75	Sender's account balance decreases by $75, and recipient's account balance increases by $75.	The transfer function worked as expected, and both the sender's and recipient's account balances were updated

							correctly.
TC04	Verify user can view their transaction history	N/A	N/A	N/A	N/A	The transaction history is displayed correctly with all transactions listed in the correct order.	The transaction history was displayed correctly, and all transactions were listed in the correct order.
TC05	Verify user cannot deposit a negative amount	N/A	N/A	N/A	Amount: -$50	System prevents user from depositing a negative amount and displays appropriate error message.	The system correctly prevented the user from depositing a negative amount and displayed the appropriate error message.
TC06	Verify user cannot withdraw more than their available balance	N/A	N/A	N/A	Amount: $1000	System prevents user from withdrawing more than their available balance and displays appropriate error message.	The system correctly prevented the user from withdrawing more than their available balance and displayed the appropriate error message.
TC07	Verify user cannot	Recipient	Recipient Account	IFSC Code: ABCD123456	Amount: $500	System prevents	The system correctly

	transfer more than their available balance	Name: Jane Doe	Number: 123456789			user from transferring more than their available balance and displays appropriate error message.	prevented the user from transferring more than their available balance and displayed the appropriate error message.
TC08	Verify user cannot transfer to an invalid account number	Recipient Name: Jane Doe	Recipient Account Number: 999999999	IFSC Code: ABCD123456	Amount: $50	System prevents user from transferring to an invalid account number and displays appropriate error message.	The system correctly prevented the user from transferring to an invalid account number and displayed the appropriate error message.
TC09	Verify user cannot transfer with an invalid IFSC code	Recipient Name: Jane Doe	Recipient Account Number: 123456789	IFSC Code: INVALIDCODE	Amount: $50	System prevents user from transferring with an invalid IFSC code and displays appropriate error message.	The system correctly prevented the user from transferring with an invalid IFSC code and displayed the appropriate error message.

Test case matrix for Loan Management use case

Test Case ID	Test Case Description	Input 1: Loan Amount	Input 2: Interest Rate	Input 3: Loan Tenure	Expected Output(s)	Remark(s)
TC01	Loan calculation with valid inputs	valid	valid	valid	Monthly EMI calculated and displayed	The test case passed successfully
TC02	Loan calculation with loan amount greater than maximum limit	invalid	valid	valid	Error message displayed: Loan amount exceeds maximum limit	The test case passed successfully
TC03	Loan calculation with loan amount less than minimum limit	invalid	valid	valid	Error message displayed: Loan amount is below minimum limit	The test case passed successfully
TC04	Loan calculation with interest rate greater than maximum limit	valid	invalid	valid	Error message displayed: Interest rate exceeds maximum limit	The test case passed successfully
TC05	Loan calculation with interest rate less than minimum limit	valid	invalid	valid	Error message displayed: Interest rate is below minimum limit	The test case passed successfully

TC06	Loan calculation with loan tenure greater than maximum limit	valid	valid	invalid	Error message displayed: Loan tenure exceeds maximum limit	The test case passed successfully
TC07	Loan calculation with loan tenure less than minimum limit	valid	valid	invalid	Error message displayed: Loan tenure is below minimum limit	The test case passed successfully
TC08	Loan calculation with invalid inputs	invalid	invalid	valid	Error message displayed: Invalid loan amount	The test case passed successfully
TC09	Loan calculation with invalid inputs	valid	invalid	invalid	Error message displayed: Invalid interest rate	The test case passed successfully
TC10	Loan calculation with invalid inputs	valid	valid	invalid	Error message displayed: Invalid loan tenure	The test case passed successfully

Test case matrix with actual values for Loan Management use Case

Test Case ID	Test Case Description	Input 1: Loan Amount	Input 2: Interest Rate	Input 3: Loan Tenure	Expected Output(s)	Remark(s)
TC01	Loan calculation with valid inputs	500000	9.5	24	Monthly EMI calculated and displayed	The test case passed successfully
TC02	Loan calculation with loan amount greater than maximum limit	1500000	8.5	36	Error message displayed: Loan amount exceeds maximum limit	The test case passed successfully
TC03	Loan calculation with loan amount less than minimum limit	5000	10	12	Error message displayed: Loan amount is below minimum limit	The test case passed successfully
TC04	Loan calculation with interest rate greater than maximum limit	750000	15.5	48	Error message displayed: Interest rate exceeds maximum limit	The test case passed successfully

TC05	Loan calculation with interest rate less than minimum limit	100000	0.5	6	Error message displayed: Interest rate is below minimum limit	The test case passed successfully
TC06	Loan calculation with loan tenure greater than maximum limit	1000000	12	120	Error message displayed: Loan tenure exceeds maximum limit	The test case passed successfully
TC07	Loan calculation with loan tenure less than minimum limit	150000	8.75	2	Error message displayed: Loan tenure is below minimum limit	The test case passed successfully
TC08	Loan calculation with invalid inputs	-1000	12	24	Error message displayed: Invalid loan amount	The test case passed successfully
TC09	Loan calculation with invalid inputs	500000	-1	12	Error message displayed: Invalid interest rate	The test case passed successfully
TC10	Loan calculation with invalid inputs	500000	8.5	0	Error message displayed: Invalid loan tenure	The test case passed successfully

Test case matrix for Manage Account Book/Card use case

Test Case ID	Personal Information	Account Information	Expected Output	Remarks
TC01	Valid	Valid	Card/Book issued successfully	The test case passed successfully
TC 02	Invalid	Valid	Error: Invalid personal information	The test case failed as expected
TC03	Valid	Invalid	Error: Invalid account information	The test case failed as expected
TC04	Invalid	Invalid	Error: Invalid personal and account information	The test case failed as expected
TC05	Missing information	Valid	Error: Missing personal information	The test case failed as expected
TC06	Valid	Missing information	Error: Missing account information	The test case failed as expected
TC07	Existing card/book	Valid	Error: Card/Book already issued for this account	The test case failed as expected
TC08	Valid	Account closed	Error: Account is closed	The test case failed as expected
TC09	Underage	Valid	Error: Applicant must be of legal age	The test case failed as expected

TC10	Valid	Low credit score	Error: Applicant does not meet credit score requirements	The test case failed as expected

Test case matrix with actual values for Manage Account Book/Card use Case

Test Case ID	Personal Information	Account Information	Expected Output	Remarks
TC01	John Smith, 35, Male	Account: 1234567890, Savings, Active	Card issued successfully	The test case passed successfully
TC02	Invalid Personal Info	Account: 1234567890, Savings, Active	Error: Invalid personal information	The test case failed as expected
TC03	John Smith, 35, Male	Invalid Account Info	Error: Invalid account information	The test case failed as expected
TC04	Invalid Personal Info	Invalid Account Info	Error: Invalid personal and account information	The test case failed as expected
TC05	Missing personal information	Account: 1234567890, Savings, Active	Error: Missing personal information	The test case failed as expected
TC06	John Smith, 35, Male	Missing account information	Error: Missing account information	The test case failed as expected
TC07	Jane Doe, 30, Female	Account: 1234567890, Savings, Active, Existing Card	Error: Card already issued for this account	The test case failed as expected

TC08	John Smith, 35, Male	Account: 0987654321, Savings, Closed	Error: Account is closed	The test case failed as expected
TC09	Joe Bloggs, 17, Male	Account: 1234567890, Savings, Active	Error: Applicant must be of legal age	The test case failed as expected
TC10	Mary Smith, 45, Female	Account: 1234567890, Savings, Active, Low Credit Score	Error: Applicant does not meet credit score requirements	The test case failed as expected

Test case matrix with actual values for Manage Issues use Case

Test Case ID	Test Case Description	Issue Description	Expected Output	Remarks
TC01	Report a new issue with complete information	Description of the issue	The new issue is added to the system with all the provided information	The test case executes as expected
TC02	Report a new issue with incomplete information	Description of the issue	The system prompts the user to provide all required information before adding the new issue	The system should indicates which information is missing as expected
TC03	Update an existing issue with a new description	Updated description of the issue	The issue's description is updated in the system	The test case executes as expected

TC04	Update an existing issue with an invalid ID	Updated description of the issue, invalid issue ID	The system informs the user that the provided issue ID is invalid and prompts them to correct it before updating the issue information	The system specifies the reason for the issue ID being invalid as expected
TC05	Mark an issue as resolved	Issue ID	The issue status is updated to "Resolved" in the system	The test case executes as expected
TC06	Search for an issue by keyword	Keyword to search for in the issue description	The system returns a list of all issues that contain the keyword in their description	The system returns all issues that contain the keyword, regardless of their status as expected
TC07	Search for an issue by ID	Issue ID	The system returns the details of the issue with the provided ID	The test case executes as expected
TC08	Search for an issue with an invalid ID	Invalid issue ID	The system informs the user that the provided issue ID is invalid and prompts them to correct it before searching for the issue	The test case executes as expected
TC09	Delete an issue	Issue ID	The issue is deleted from the system	The test case executes as expected
TC10	Delete an issue with an invalid ID	Invalid issue ID	The system informs the user that the provided issue ID is invalid and prompts them to correct it before deleting the issue	The test case executes as expected

Test case matrix with actual values for Logout use case

Test Case ID	Test Case Description	Input(s)	Expected Output(s)	Remark(s)
TC01	Request to logout successfully	N/A	User logged out successfully	The test case passed successfully
TC02	Request to logout with active session and no user input	N/A	User logged out successfully	The test case passed successfully
TC03	Request to logout with active session and user input "yes"	"yes"	User logged out successfully	The test case passed successfully
TC04	Request to logout with active session and user input "no"	"no"	User remains logged in	The test case passed successfully
TC05	Request to logout with active session and user input "cancel"	"cancel"	User remains logged in	The test case passed successfully
TC06	Request to logout with invalid session	N/A	Error message displayed: User session not found	The test case passed successfully
TC07	Request to logout with expired session	N/A	Error message displayed: User session has expired	The test case passed successfully
TC08	Request to logout with session timeout	N/A	User logged out due to session timeout	The test case passed successfully
TC09	Request to logout with multiple active sessions	N/A	User logged out successfully from all sessions	The test case passed successfully
TC10	Request to logout while already logged out	N/A	Error message displayed: User is already logged out	The test case passed successfully