**ASSIGNMENT SUBMISSION COVER SHEET**   
**Student Id:** cse24-116  
**Student names:** NEELO MIMI PILANE  
**Student email:** cse24-116@thuto.bac.ac.bw  
**Cohort:** 2025  
**Assignment title:** CSE202- OBJECT ORIENTED ANALYSIS & DESIGN WITH JAVA  
**Date of submission:**   
**Programme of Study:** COMPUTER SYSTEMS ENGINEERING  
**Year of Study:** Year 2  
  
  
Intellectual property statement. By checking the box below, I certify that this assignment is my own work and is free from plagiarism. I understand that the assignment may be checked for plagiarism by electronic or other means and may be transferred and stored in a database for the purposes of data-matching to help detect plagiarism. The assignment has not previously been submitted for assessment in any other unit or to any other institution. I have read and understood the Botswana Accountancy College plagiarism guidelines policy.   
  
**☐ Agree**   
  
**Signature…………………………………….**   
**Date………19/09/2025…………………………………..**  
  
  
  
  
**Part A: System Documentation Instructions**  
  
**Functional Requirements**  
**User Verification (Login)**-To access their accounts, customers must be able to log in safely. Authentication should only be available to registered customers.

**Management of Accounts** - Consumers are able to open savings, investment, and check accounts. It is possible for a customer to have more than one account.

Rules must be enforced by the system:

A savings account pays 0.05% interest each month and permits deposits but not withdrawals.

Investment accounts pay 5% monthly interest, accept deposits and withdrawals, and require a minimum initial deposit of BWP 500.

Usually used for salary payments, a cheque account requires proof of employment and permits deposits and withdrawals.

Usually used for salary payments, a check account requires proof of employment and permits deposits and withdrawals.

**Deposit money -** Funds can be deposited into any customer's account by both customers and bank tellers.  
 **Withdraw money -** Only investment and check accounts—not savings accounts—allow customers and bank tellers to take money out.  
  
  
  
  
  
  
  
  
**Non-Functional Requirements**  
**Safety** - The system must use authorization and authentication procedures to safeguard user data. The system should only be accessible by authorized users (customers, tellers).

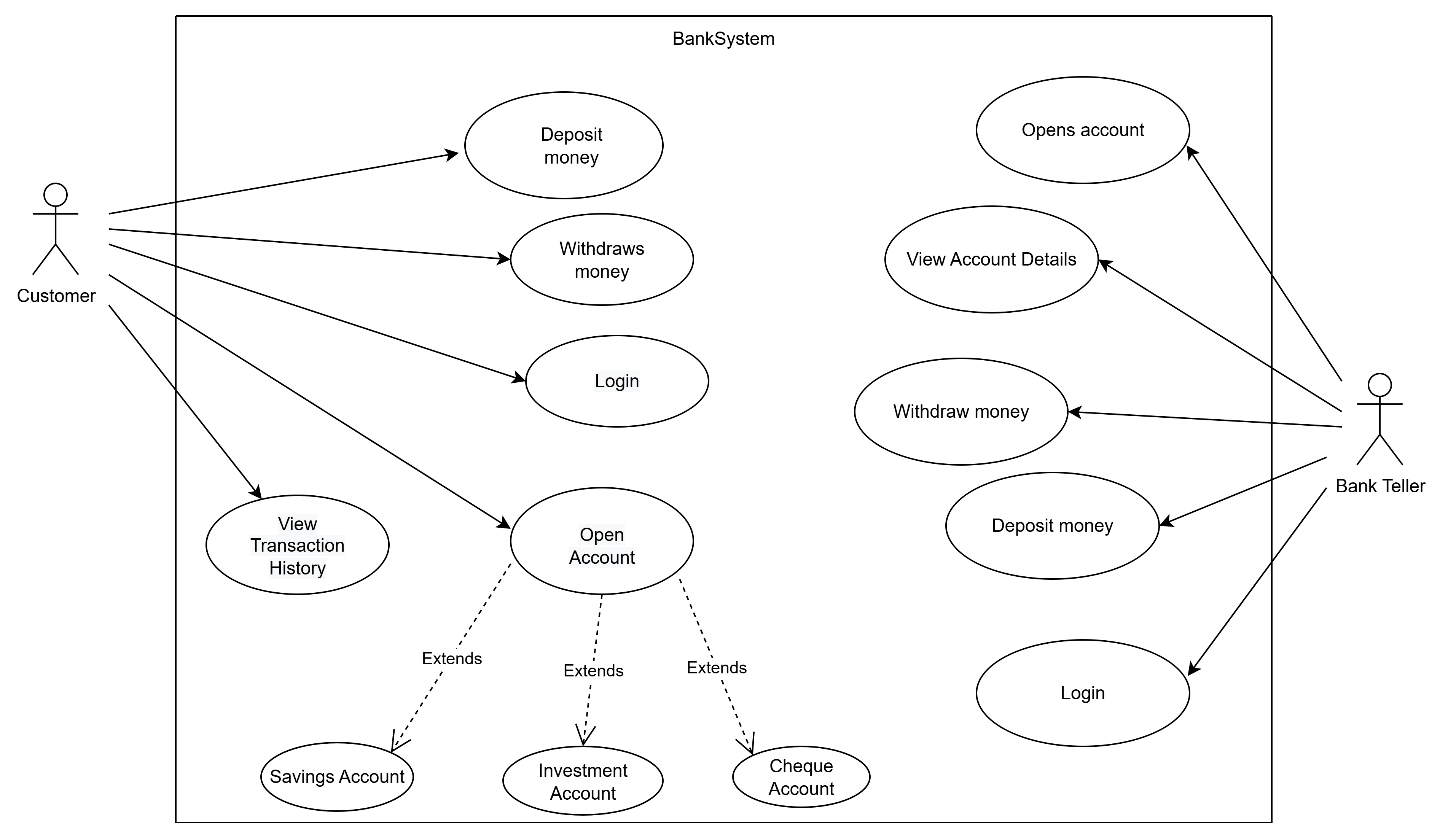
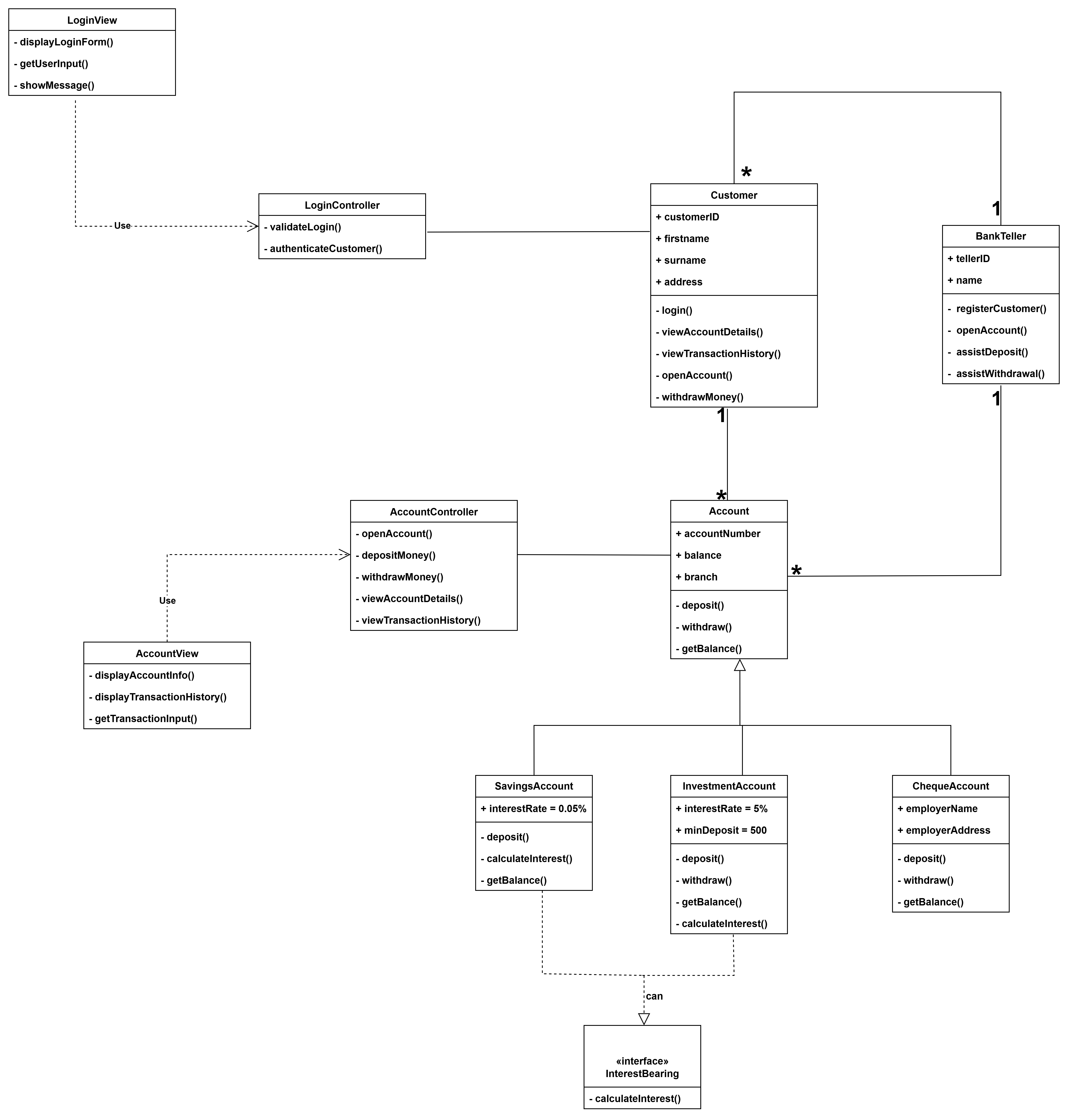
**Performance** - Deposits and withdrawals should be processed by the system instantly. Interest must be calculated by the end of each month, or within an acceptable processing time.

**Usability -** The interface needs to be easy to use, accessible, and simple for both bank tellers and customers. The colors used should represent the current trends of technology e.g it is necessary to provide unambiguous error messages and confirmations.

**Scalability -** Multiple clients and accounts should be supported by the system without experiencing any performance problems.

**Availability & Dependability** - Customers should be able to access the system around-the-clock. All updates must be persistent, and transaction data should never be lost.

**Sustainability** - To facilitate updates, the system should be modular, separating the controller, GUI, and core logic.

**Structural UML Modelling**  
 **1. USE CASE DIAGRAM**  
  
  
The customer and the bank teller are the two primary actors whose use of the bank system is depicted in the use case diagram  
The system allows the customer to interact directly with their accounts. To access their banking services, a customer can safely log in. After entering, they can open a new account and select from a variety of options, including a check account, savings account, or investment account. In order to keep track of previous activities, they can also view their transaction history and make deposits and withdrawals. This guarantees that clients are in charge of their money and that their banking activities are transparent. A system assistant is what the bank teller does. Tellers can open accounts for customers to help them get started with the bank. They can also view detailed account information when needed, which makes it easier to respond to client inquiries. Tellers are also responsible for overseeing the proper and effective handling of deposits and withdrawals. The diagram generally shows a clear division of labor: customers use the system to manage their personal accounts, while bank tellers help customers and supervise transactions on the system's backend.  
**CLASS DIAGRAM**  


Banking System Class Diagram – Brief Description

The Banking System Class Diagram models the structure and relationships of the system using Object-Oriented principles (OOAD). It is designed with three main layers: Boundary (UI), Controller (Application Logic), and Core (Model) classes, ensuring modularity and maintainability.

Model Layer Core Classes:

Customer: Stands in for a bank customer and has access to transaction history and account information.

All account types are based on the Account (abstract) class, which defines common methods (deposit, withdraw) and attributes (accountNumber, balance). SavingsAccount and InvestmentAccount are subclasses of Account that use their own calculateInterest() methods (polymorphism) to implement the InterestBearing interface. ChequeAccount: An account subclass that doesn't generate interest. BankTeller: Acts as a teller's representative, handling account management and customer registration.

**2. User interface**

InterestBearing: Shows abstraction and polymorphism by defining calculateInterest(), which is used by interest-earning accounts (investment and savings).

**3. UI Layer Boundary Classes**

LoginView: Manages user login information.

AccountView: Shows transaction history and account details.

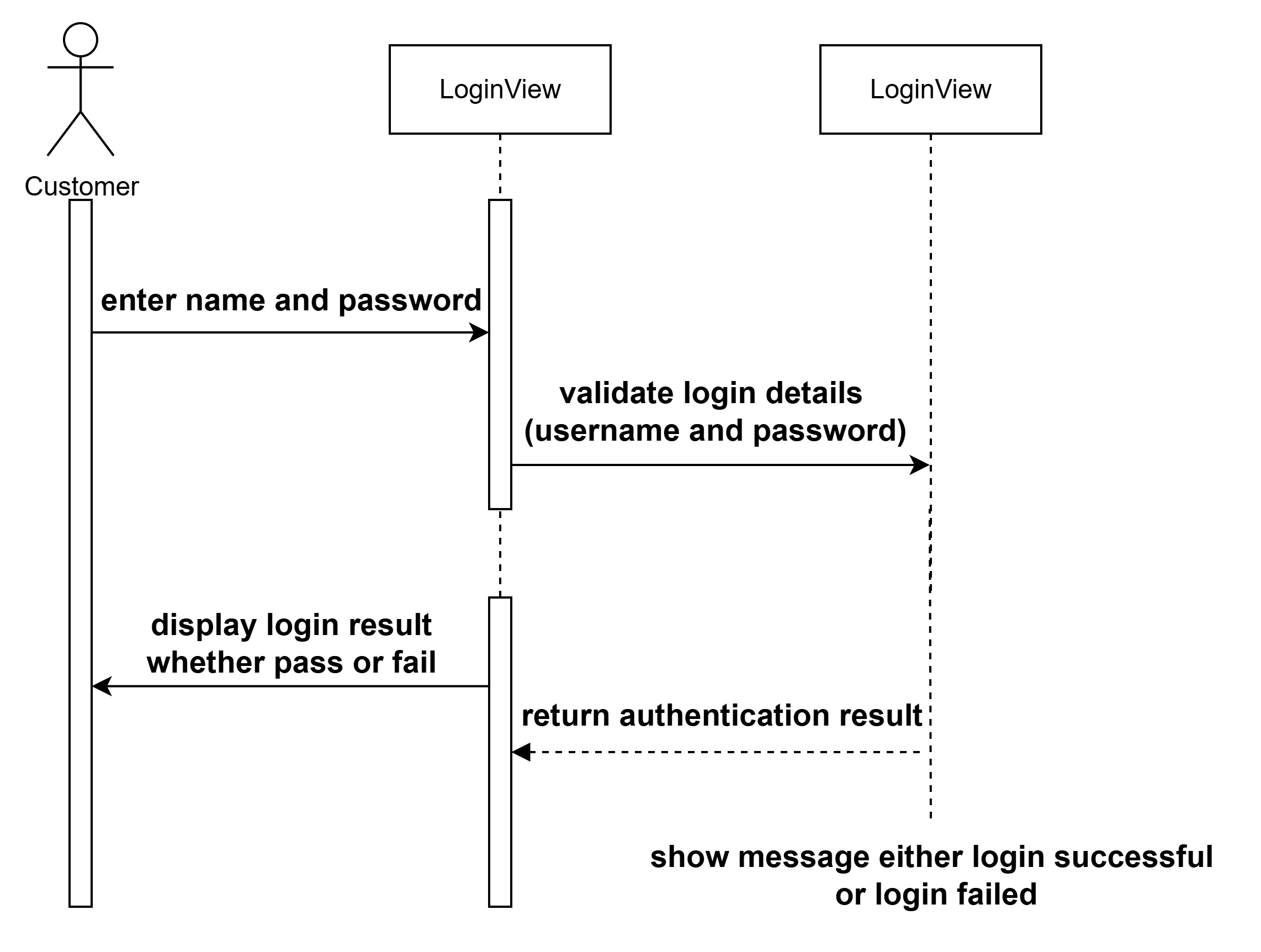
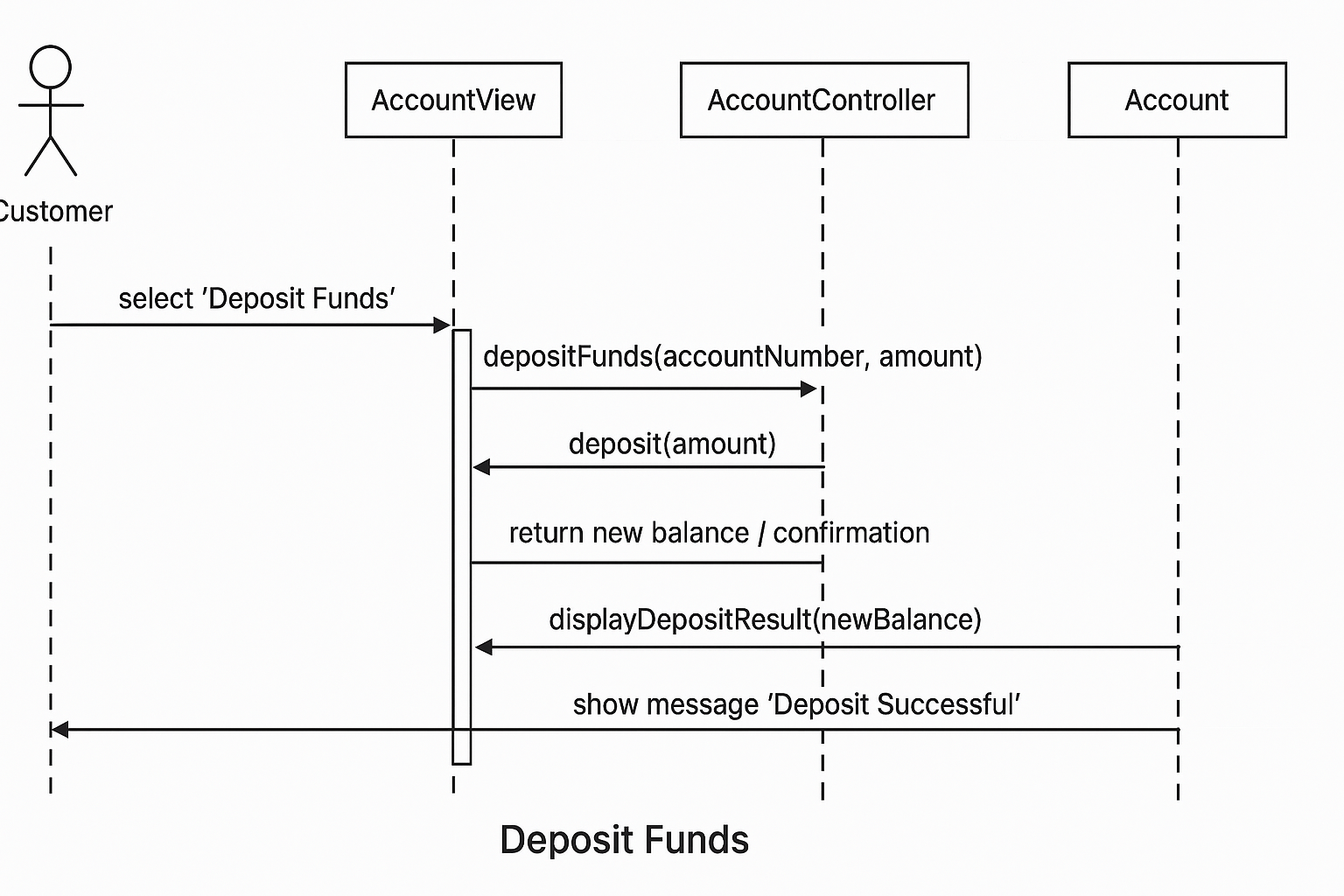
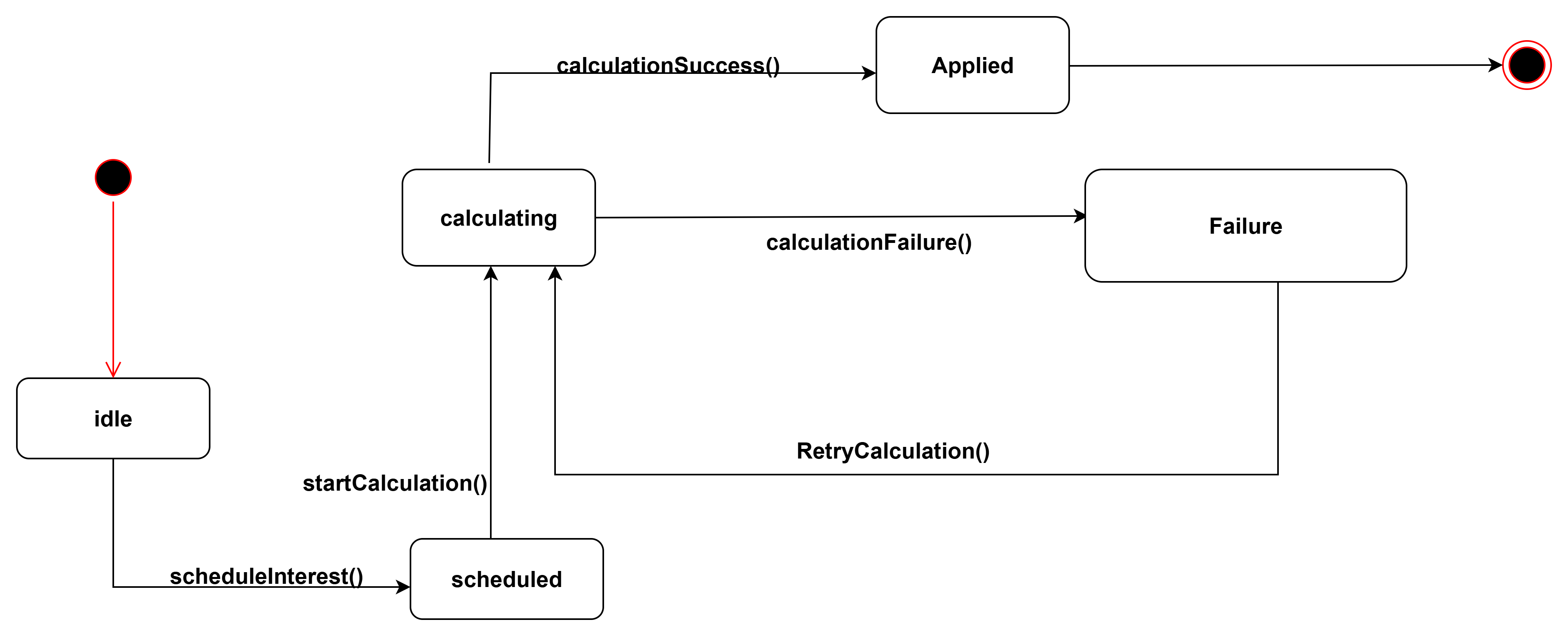
TransactionView / RegisterCustomerView: For customer registration and deposits/withdrawals.

**4. Controller Classes (Application Logic Layer)**

* LoginController: Validates and authenticates Customer credentials.
* AccountController: Handles account operations (open, deposit, withdraw, view details) linking Boundary to Core classes.

**5. Relationships and Symbols**

* Association (solid line): Links BankTeller → Customer/Account; Controllers → Core classes.
* Dependency (dashed line): Boundary → Controller classes.
* Inheritance (solid line + hollow triangle): Account subclasses → Account.
* Interface realization (dashed line + hollow triangle): InterestBearing → Savings/InvestmentAccount.
* Multiplicity: Customer (1) → (1..*) Account; BankTeller (1) → (0..*) Customer/Account.

**Behavioural UML Modelling**  
  
**SEQUENCE DIAGRAM (LOGIN PROCESS)**  
  
  
  
  
  
  
  
  
  
  
  
**SEQUENCE DIAGRAM (DEPOSIT PROCESS)**  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**STATE DIAGRAM**  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Interview Record**

**Project:** Banking System OOAD Assignment  
**Student:** Neelo Mimi Pilane  
**Date of Interview:** 18/09/2025  
**Time:** 0815/ 10:30  
**Mode:** Microsoft Teams Breakout Room 1  
**Interviewee(s) :** Mr Themba Moeng and Mr Kentsenao Baseki  
**Interviewer:** Neelo Mimi Pilane

**Purpose of Interview**

The purpose of this interview was to gather detailed requirements and understand the processes of the banking system to support the creation of Use Case, Sequence, and State Diagrams.

**Interview Questions & Responses**

|  |  |  |
| --- | --- | --- |
|  | **Question** | **Response / Notes** |
| 1 | Can you describe how customers log in to the banking system? | Customers enter their username and password. The system verifies credentials and grants access. |
| 2 | What operations can a customer perform once logged in? | Deposit funds, withdraw funds, view account balance, view transaction history. |
| 3 | How does the deposit process work? | Customer enters amount then system updates account balance then confirmation displayed. |
| 4 | How does the withdrawal process work? | Customer requests amount then system checks balance then deducts amount then confirmation displayed. |
| 5 | Can customers open new accounts? | Yes, through the system; staff assists with verification. |
| 6 | Are there different types of accounts? | Yes. Savings, investment and cheque accounts. |
| 7 | Is a customer allowed to have many accounts? | Yes. A customer can have many different types of accounts but not many accounts that are of the same type. |

**Observations & Notes**

* Customers interact with the system primarily through the user interface or GUI.
* Controllers handle the logic between views and core data models.
* The interview provided clarity on edge cases and user roles.

**Signature of Interviewer**

**Name:** Neelo Mimi Pilane  
**Date:** 18/09/2025