# Object-Oriented Analysis & Design with Java

# Banking System – Part A Submission

# Name: *Kaelo Kealeboga Selome* Student ID: *cse24-086* Date: *18/09/2025*

# Requirements Elicitation

**Date:** 18 Sept 2025  
**Interviewer:** Kaelo Kealeboga Selome  
**Interviewee (Client):** Mr Themba Moeng (acting as bank manager)  
**Purpose:** To gather functional and non-functional requirements for the banking system.

**Q1:** What are the core services this bank system should provide?  
**A:** The system should allow customers to register with the bank, open accounts, deposit money, withdraw and deposit money (where applicable), and check their account balances.

**Q2:** What performance is expected from the system?  
**A:** The system should be smooth and responsive and only take a few seconds to perform transactions.

**Q3:** How is interest supposed to work?  
**A:** Savings Account earns 0.05% monthly interest, while Investment Account earns 5% monthly interest. The system must automatically calculate and apply interest monthly.

**Q4:** Should there be any restrictions when opening an account?  
**A:** Yes, Investment Account must have an initial deposit of at least BWP500. Cheque Account requires employer information.

**Q5:** What should the system do when a customer makes a transaction?  
**A:** It should validate the account, update the balance, and store the transaction details in a database for record keeping.

**Q6:** Are there any special quality requirements?  
**A:** The system should be secure, reliable and user-friendly.

**Q7:** Are there any extra features you would like?  
**A:** I advise you to stick to the scope to maintain the timeline but you are free to add one or two like transfer funds if you can maintain the timeline.

**Q6:** Should all account data be stored permanently?  
**A:** Yes, all account data should be stored permanently in a database so as to maintain data integrity.

This section identifies and specifies the requirements of the Banking System.

Functional Requirements

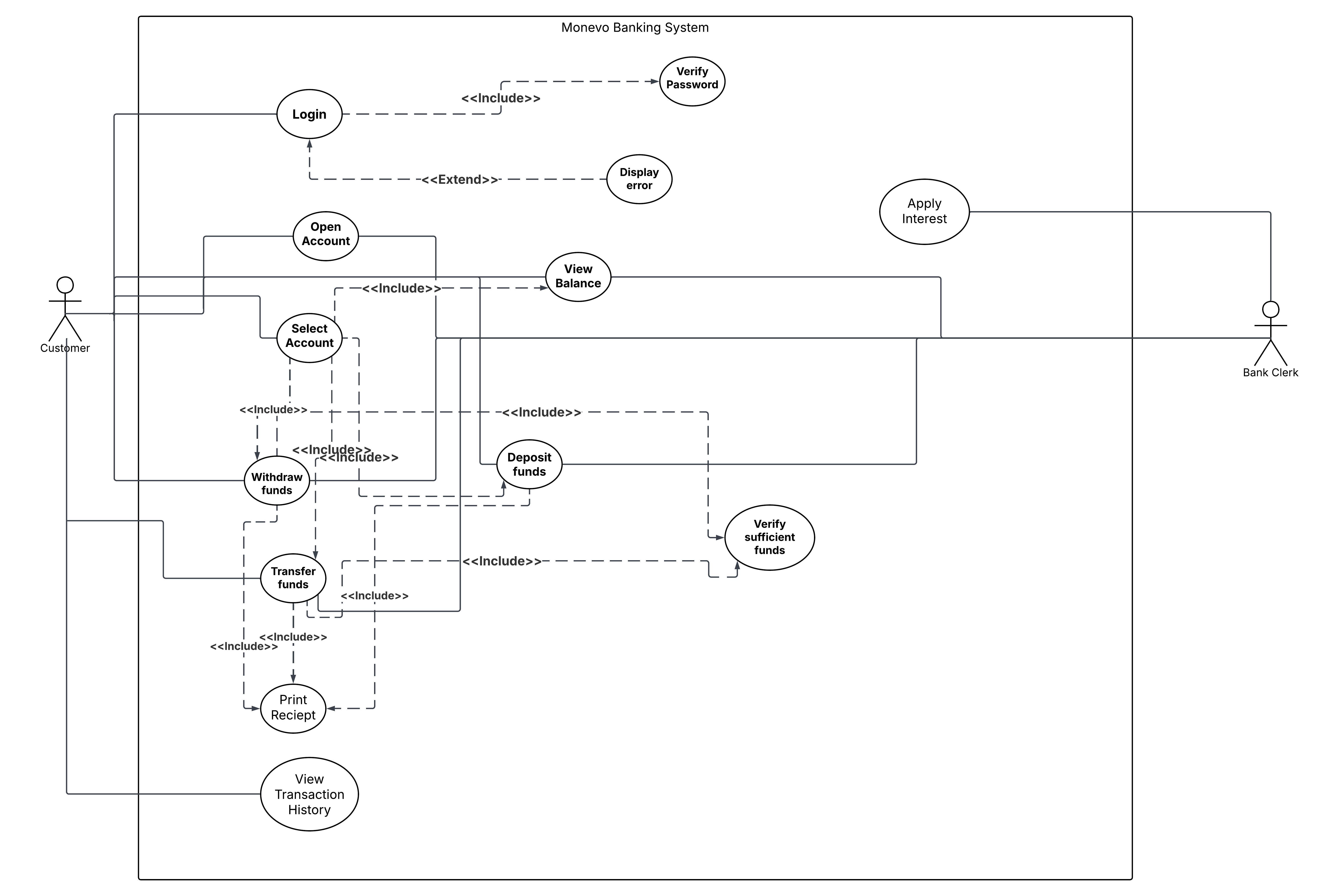
* Allow a customer to register and log in.
* Allow a customer to open one or multiple accounts (Savings, Investment, or Cheque).
* Allow deposits into any account.
* Allow withdrawals from respective accounts (Investment and Cheque, but not Savings).
* Calculate and pay monthly interest (0.05% for Savings, 5% for Investment).
* Display account details (balance, account number, recent transaction history).
* Maintain a record of all transactions.
* Authorize account opening rules (minimum BWP500 for Investment, employer details for Cheque).
* Print a receipt after each transaction.

## Non-Functional Requirements

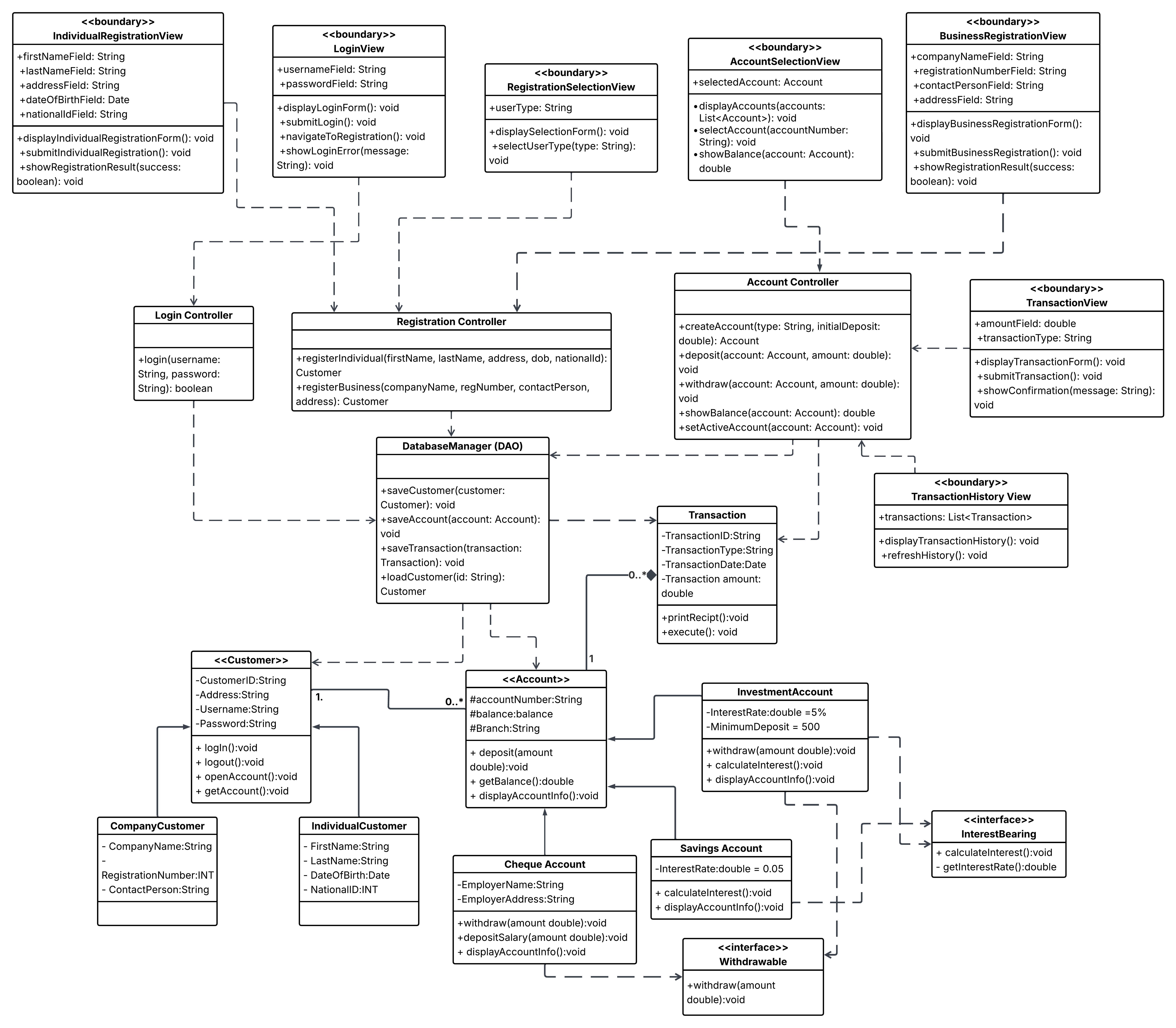
* Security: Customer data and transactions must be protected.
* Performance: Transactions should be processed within seconds.
* Reliability: All transactions must be accurately saved without data loss.
* Usability: The GUI must be simple for non-technical users.
* Portability: The system should run on any machine with Java installed.
* Maintainability: Separation of concerns between GUI, controller, and business logic layers.
* Scalability: The system should support many customers and accounts as the bank grows.
* Data Integrity: All account and customer data should be stored in the database and updated if altered with during transactions.

Structural UML Modelling

Use Case Diagram



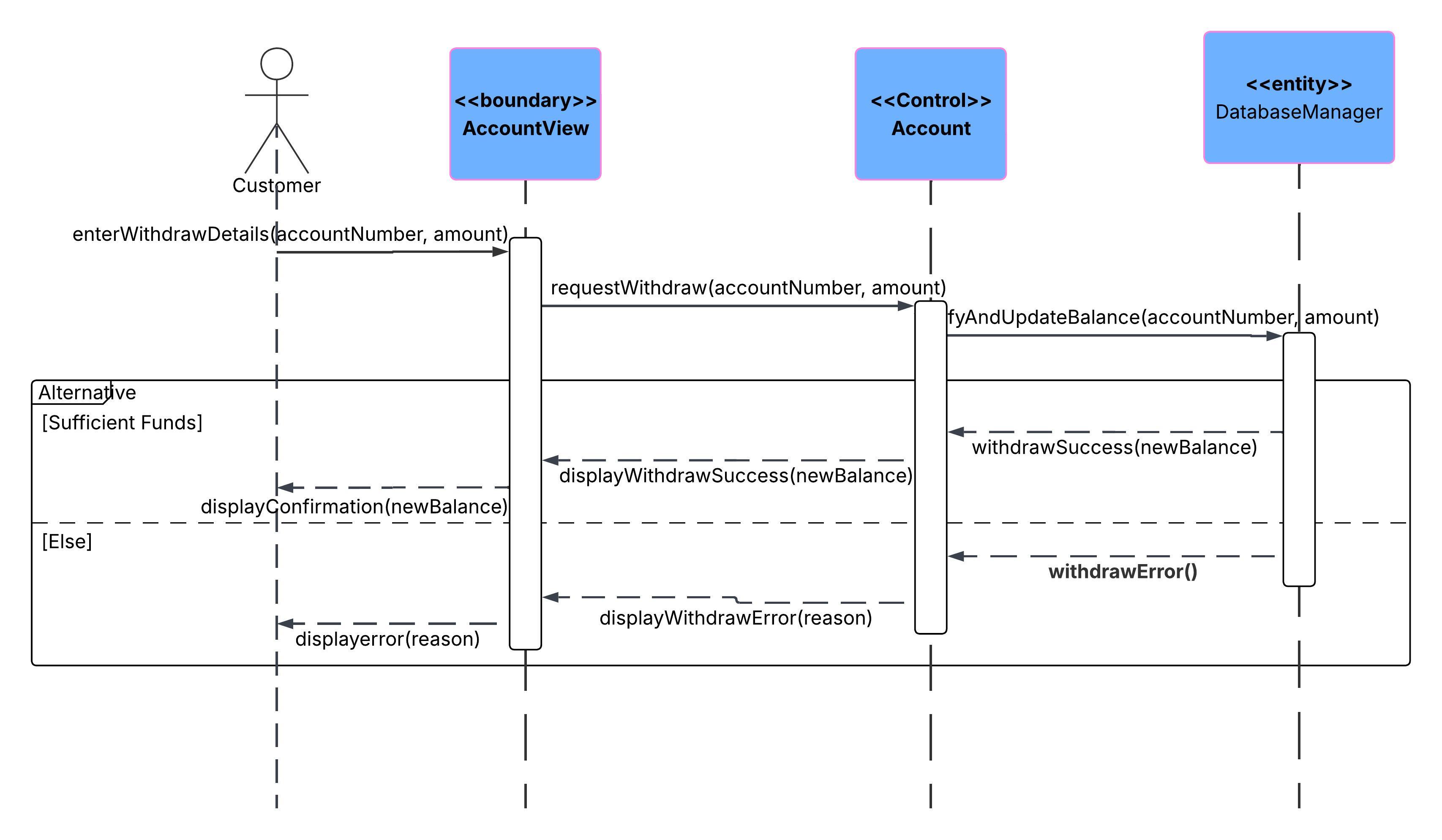
Class Diagram



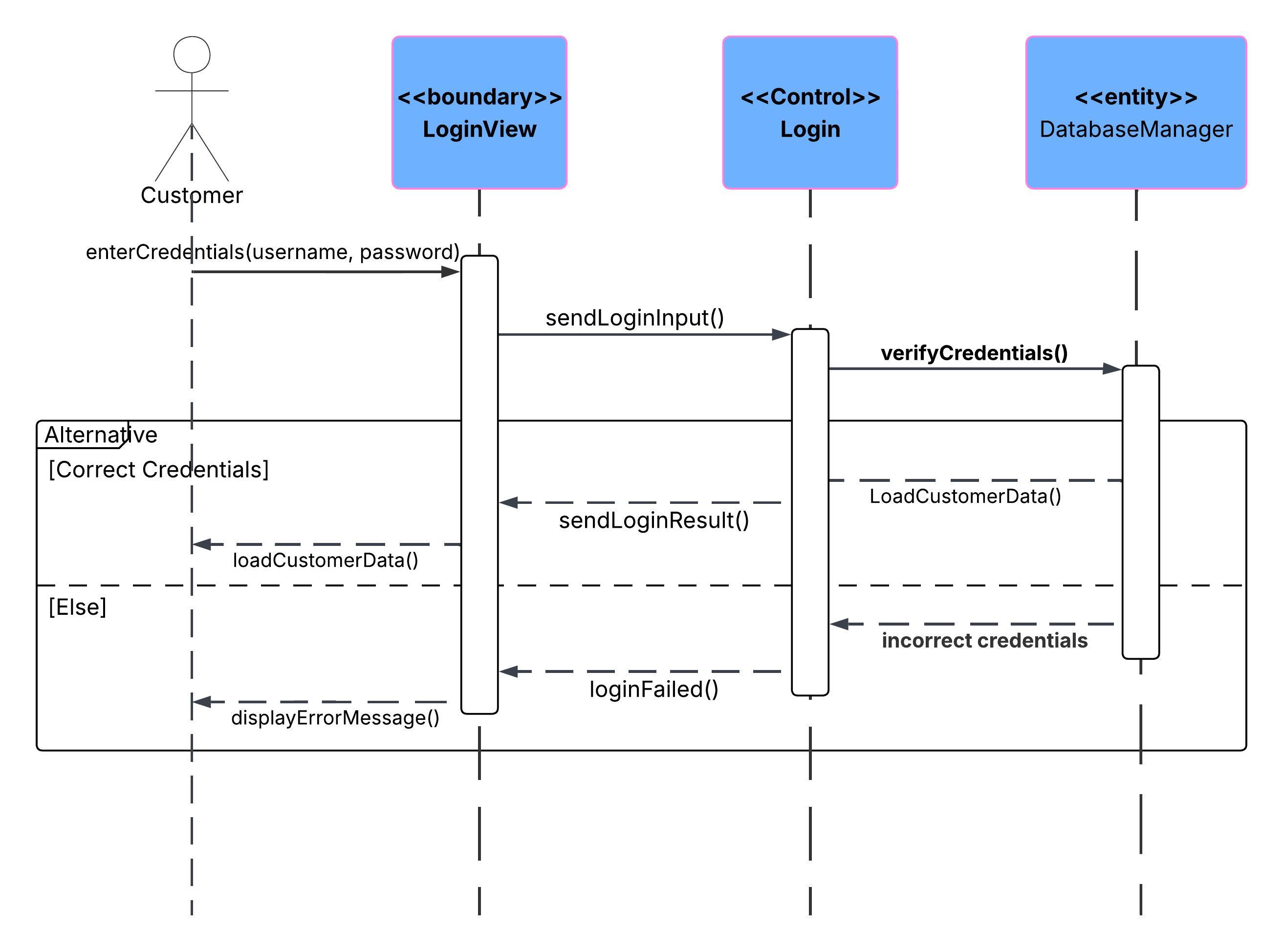
Behavioral UML Modelling

Sequence Diagrams

Deposit Funds



Login



State Diagram

