

19/ 09/ 2025



Year 2

Computer Systems Engineering

19th September 2025

OOAD (Java)

2025

Cse24-007@thuto.bac.ac.bw

Tlhongbotho Mercy Dichaba

Cse24-007

# Requirements Elicitation

## Functional Requirements

The Banking System shall provide the following functionalities:

1. **Customer Registration:** The system must allow a new customer (individual or organization) to register by providing personal details such as first name, surname, and address.
2. **Account Management:**
   * Open new accounts (Savings, Investment, Cheque) for an existing customer.
   * Ensure an Investment account requires a minimum initial deposit of BWP 500.00.
   * Ensure a Cheque account can only be opened for employed individuals, capturing their employer's details.
   * Allow a single customer to hold multiple accounts of different types.
3. **Transaction Processing:**
   * Allow deposits to be made into any account.
   * Allow withdrawals from Investment and Cheque accounts.
   * Prevent withdrawals from Savings accounts.
4. **Interest Calculation & Payment:** The system must automatically calculate and apply monthly interest to relevant accounts (5% for Investment, 0.05% for Savings).
5. **User Authentication:** A secure login mechanism for bank employees or customers to access the system.

## Non-Functional Requirements

The Banking System must adhere to the following quality constraints and attributes:

**1. Security:**

* **Data Confidentiality:** All sensitive customer data (e.g., names, addresses, balances) and user passwords must be encrypted both in transit and at rest.
* **Authentication:** The system shall require secure user authentication (login) to access any functionality. Passwords must be stored as hashed values, not plain text.
* **Authorization:** The system shall ensure that users can only perform actions on accounts they are authorized to access (e.g., a teller cannot approve their own loan).
* **Audit Trail:** A secure log of all user logins, transactions, and changes to critical data must be maintained for accountability.

**2. Reliability & Availability:**

* **Uptime:** The system must be highly available and operational during standard banking hours (e.g., 99.5% uptime between 8:00 AM and 5:00 PM).
* **Data Integrity:** The system must guarantee that all financial transactions are processed accurately and completely. Once a transaction is committed, it must not be lost. The system must prevent errors such as double-debiting or incorrect balance calculations.
* **Error Handling:** The system must gracefully handle errors (e.g., network failures, invalid input) without crashing and provide clear, helpful error messages to the user.

**3. Performance:**

* **Response Time:** The system shall respond to user interactions (e.g., loading an account list, performing a search, completing a deposit) in less than **3 seconds** under normal load.
* **Throughput:** The system should be able to support multiple concurrent users (e.g., 10+ bank tellers) performing transactions simultaneously without significant performance degradation.

**4. Usability:**

* **Learnability:** The user interface shall be intuitive enough for a new bank teller to learn core functions (e.g., deposit, withdrawal, viewing history) with minimal training (less than one hour).
* **Efficiency:** Common tasks (like processing a deposit) should be achievable in a minimal number of clicks. Navigation between different sections (account list -> specific account -> transaction history) should be clear and logical.
* **Clarity:** Information presented to the user (e.g., account balances, transaction history) must be clear, unambiguous, and well-formatted.

**5. Maintainability:**

* **Modularity:** The system shall be designed using Object-Oriented Principles (abstraction, encapsulation, inheritance, polymorphism) to ensure the code is well-structured and organized into logical components (e.g., separate classes for Customer, Account, Transaction).
* **Extensibility:** The design must make it easy to add new features in the future (e.g., adding a new "Loan Account" type, integrating with a new payment gateway) without requiring a major rewrite of the existing codebase.
* **Code Readability:** The source code must be well-documented with comments and follow standard Java coding conventions.

**6. Scalability:**

* The system's architecture should allow for potential future growth, such as handling an increasing number of customers, accounts, and transactions without a fundamental redesign.

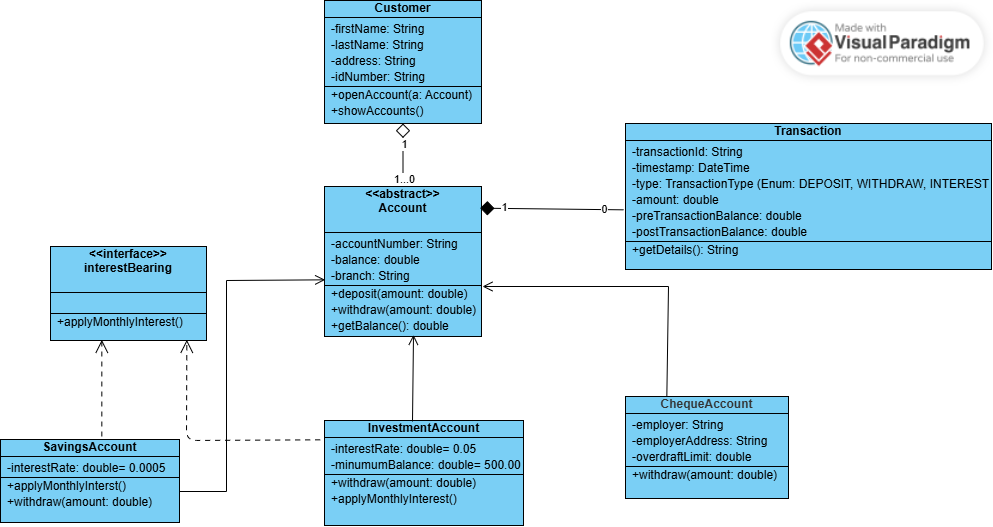
# 2. Structural UML Modelling

## 2.1. System Use Case Diagram

A diagram of a company

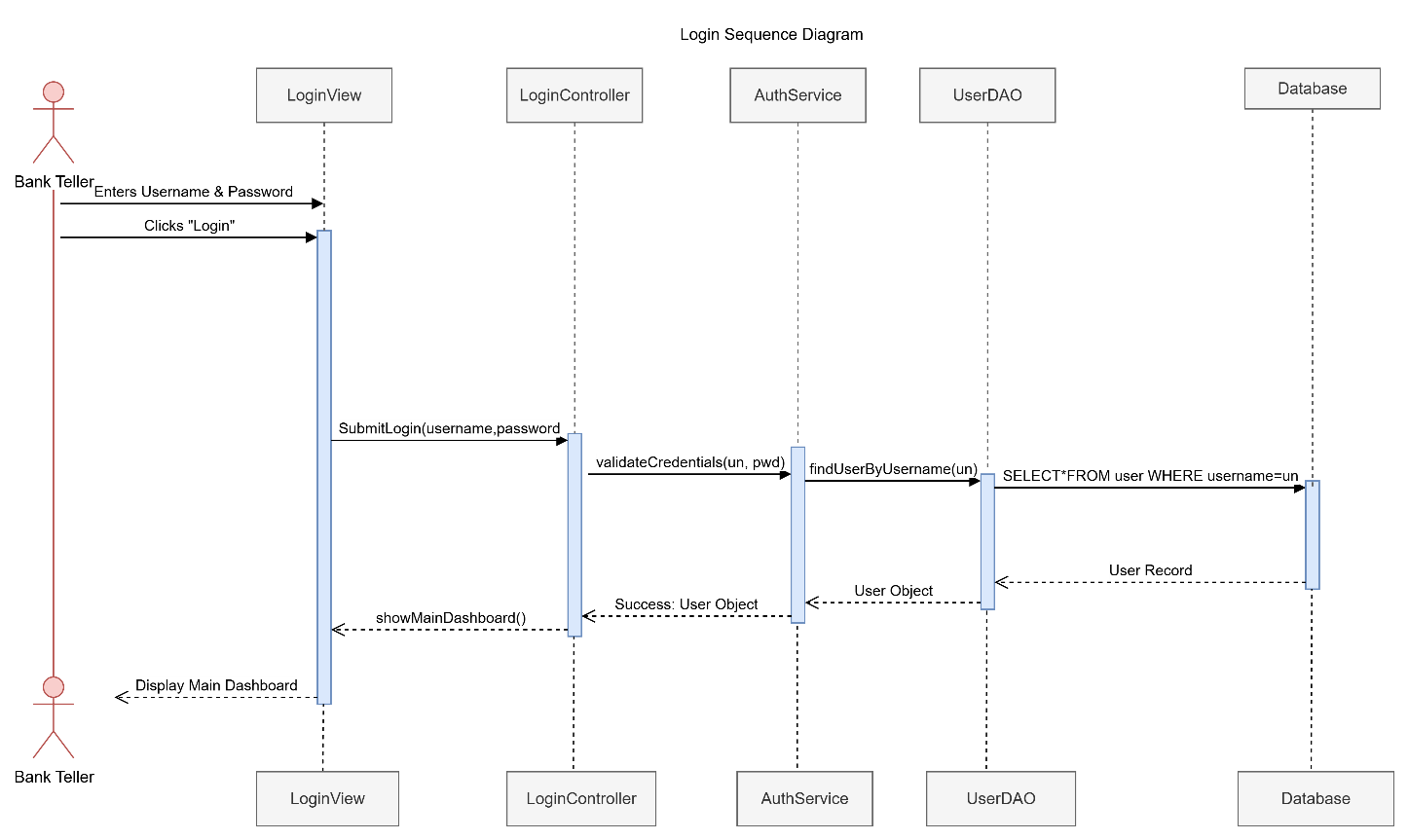
AI-generated content may be incorrect.

## 2.2. Class Diagram



# Behavioural UML Modelling

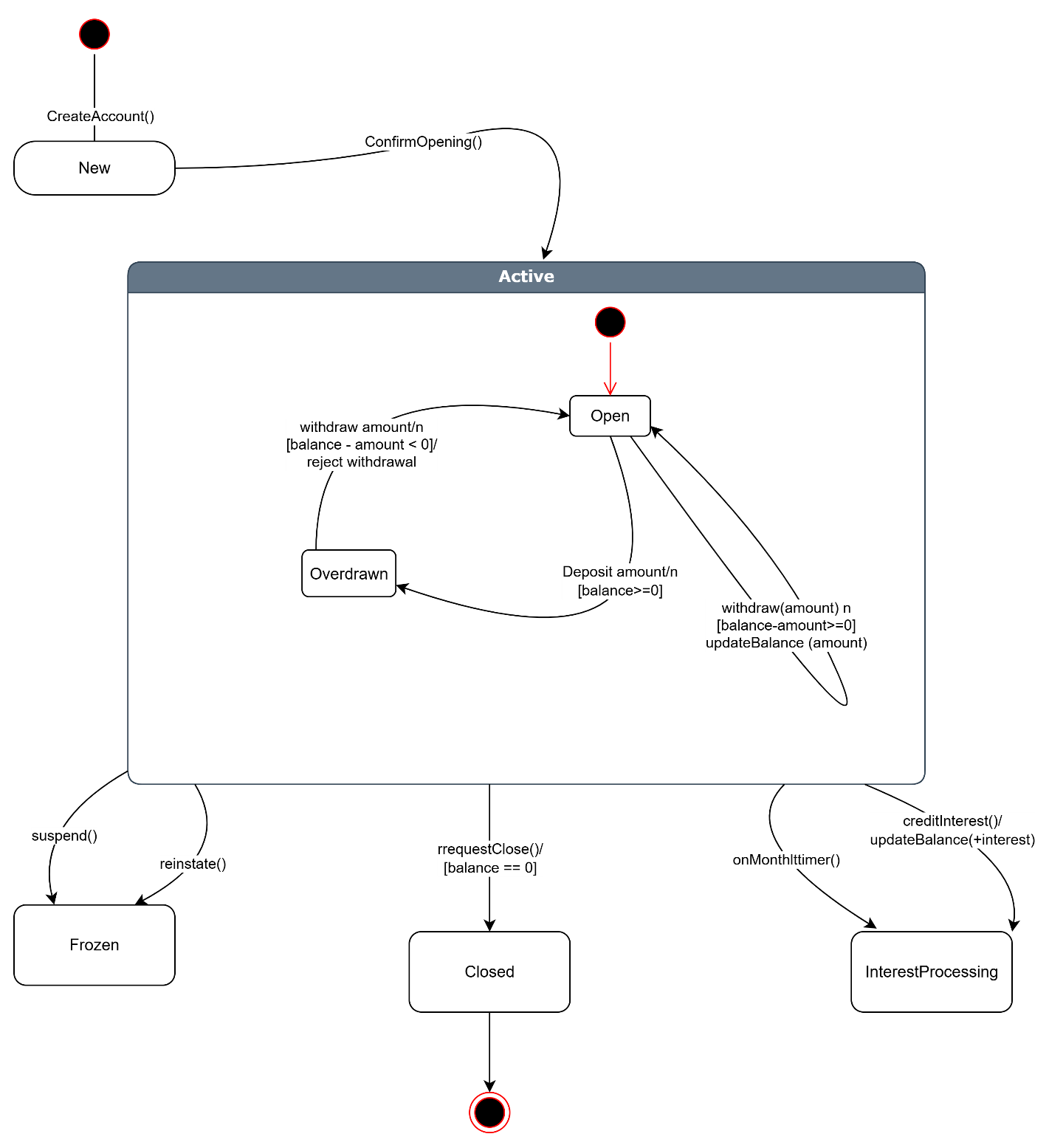
## 3.1. Login And Deposit Sequence diagrams



A computer screen shot of a black background

AI-generated content may be incorrect.

## 3.2. State Diagram



**Appendix: Formal Interview Record**

**Interview Topic:** Requirements Elicitation for Banking System  
**Interviewer:** Tlhongbotho Mercy Dichaba (Student)  
**Client:** Themba Moeng (Lecturer/Tutor acting as Client)  
**Date:** September 18th  
**Time:** 08:46

**Transcript:**

**Dichaba:** Good morning, Mr. Moeng. Thank you for taking the time to speak with me today. I’d like to ask you a few questions to better understand the requirements for the new banking system we are developing.

**Moeng:** Good morning. You’re welcome, I’m happy to help clarify the system's needs.

**Dichaba:** My first question is, what should the user be able to see immediately after logging in?

**Moeng:** Yes, they should be able to navigate by basically say OK, these are my account lists. I should be able to select an account or search for it. Once I find an account, then it should facilitate the transactions such that I'm in a position to say I'm withdrawing where applicable or depositing where applicable and also having the ability to view the balance and also being in a position to pick a starting and ending date to show the transaction history of the selected account.

**Dichaba:** Oh, OK Um, what kind of transactions can be performed?

**Moeng:** Primarily it will be the ability to view the balance to withdraw in the accounts where applicable, the ability to deposit, the ability to then OK, I've talked about viewing the balance, those are the primary activities and then obviously the secondary ones, the logging in, logging out. Changing the password, yes.

**Dichaba:** OK, should the system keep any record of the transactions?

**Moeng:** Yes, it's critical since one of the features is to view the transaction history, meaning as you deposit and withdraw a record will be kept in terms of when the transaction occurred, in terms of date and time, the amount involved, the account involved. Etcetera. The type of a transaction? Was it a deposit? Was it a withdrawal?

**Dichaba:** OK. Are there any specific reports of use needed in the system?

**Moeng:** Yeah, primarily we should be able to see the transaction history and on the bank like side, we can then view some reports that show in terms of each month, what are the total deposits, what were the total withdrawals and the difference.

**Dichaba:** OK, I think that covers all my questions for now. Thank you so much for your time and detailed explanations.

**Moeng:** OK. All right. You're welcome. All the best with the project.