**CSE202- Object Oriented Analysis & Design with Java**

**Assignment Part A: Banking System Documentation**

**Student Name:** Ashley Thabang Motana  
**Student Registration Number:** cse24-121  
**Date of Submission:** 19/09/2025

**Appendice: Interview Record**

**Interviewer:** Ashley Thabang Motana  
**Interviewee:** Mr. Themba Moeng  
**Date:** 18 September 2025  
**Subject:** Requirements Elicitation for the New Banking System

**Q1: Can customers update their personal information once they are registered?  
 A:** Yes, absolutely. Customers should also have the opportunity to update their contact information which includes their address and phone number using the online profile. **Q2: Would you like the system to be able to accommodate two or more user roles?   
A:** Yes. We should have three different positions Customer, Bank Staff, and Administrator. Customers control their accounts. Bank Staff has access to list of customers, assists in the creation of accounts and other transactions. Administrators deal with user roles and system settings. **Q3: What does the bank staff need to do with creating or approving accounts?  
 A:** Due to compliance, any new account applications may only be activated and a customer can use them after being reviewed and approved by a member of Bank Staff. They are not supposed to open immediately. **Q4: Do you anticipate customers to be notified of anything?  
 A:** Yes. Any transaction done (deposit or withdrawal) on the accounts of customers should be notified through email or SMS. Cheque account low balance warning messages would also come in handy. **Q5: Should the system manage joint accounts?  
 A:** In the case of the first version, we are going to concentrate on individual accounts only to ensure that the scope is manageable. In the future an update of joint accounts can be a feature. **Q6: Does the system permit fund transfers amongst accounts of a customer?   
A**: Yes, this is essential. A customer should be in a position to move money between his savings, investments and the cheques accounts without a problem. **Q7: Do you want to facilitate the transfers between the accounts of different customers?   
A:** Not at this stage. We shall begin with within bank transfers (between accounts of the same bank which is owned by the user). The complexity in Inter-account transfers across customers can be realized, and it shall be looked into in Phase 2. **Q8: Is it necessary to enable scheduled payments in the system?  
 A:** Yes, standing orders to make regular and fixed amount payments have to be supported. This is one of the major characteristics to our customers. **Q9: Do they have any transaction limits?   
A:** Yes. To be on the safe side we will limit cheques accounts to BWP 5,000.00 withdrawal per day. No limit will be placed on deposits. **Q10: A question of whether cheque accounts should be allowed to have overdrafts.**

**A:** No. The withdrawals should only be restricted to the amount. The system should not allow the transaction whereby any of the types of accounts will be in a negative balance.

**Q11: Would you like interest calculated on a real-time or on a monthly basis?**

**A:** Calculation and application of interest should be done at the end of every calendar month.

**Q12: Would interest be shown in the transaction history?**

**A:** Yes. When interest is charged, it should be recorded in the history of the account as a credit transaction and the description should be as an interest such as: monthly interest.

**Q13: When a customer is closing an account before the month ends should they be provided with interest partially?**

**A:** No. We only pay interest on those accounts that are in existence as at the end of the month. There will be no biased interest.

**Q14: Do the staff members need to access differently?**

**A:** Definitely. A customer has no option of accessing and carrying out transactions on his or her own accounts. A Bank Staff member will have a possibility to look at all the customers and all accounts but will not be able to do transactions with them without customer authorization. Administrators are allowed access to the system in entirety.

**Q15: Do you anticipate the system to record all actions?**

**A:** Security and auditing. The system shall keep a record of every user login (success and failure), and all financial transactions (deposit, withdrawal and transfers).

**Q16: Is it necessary that the system automatically logs out idle users?**

**A:** Yes. To keep the security, an inactive session must have an automatic end that takes place after 5 minutes.

**Q17: Is the interface supposed to be multi-lingual?**

**A:** The first release can be done in English-only.

**Q18: Would you like plain dashboard?**

**A:** Yes. The primary screen of the customer needs to be a dashboard with a summary of all their account balances and the 5 most recent transactions. **Q19: Should the system be designed for mobile devices?**  
**A:** The design should be "responsive," meaning it should work well on both desktop computers and mobile devices like phones and tablets.

**Q20: How should error messages be handled?**  
**A:** All error messages must be clear and helpful for the user. Technical details should be logged internally for developers but never shown to the user.

**1. Requirements Elicitation**

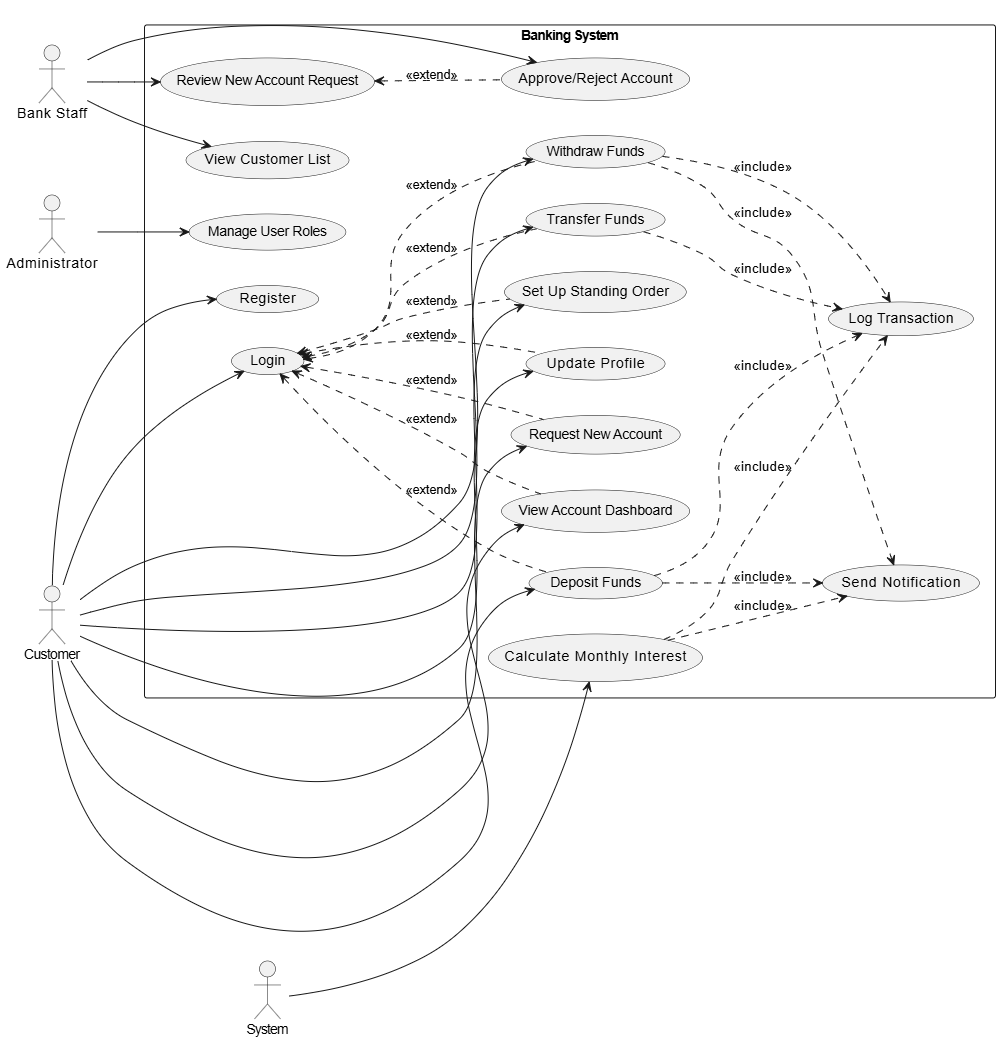
**1.1. Functional Requirements**

Based on the interview with Mr. Moeng, the banking system must perform the following functions:

* **User Management:**
  + The system shall support three user roles: Customer, Bank Staff, and Administrator.
  + Customers shall be able to register and update their personal information (e.g., address, phone number).
* **Account Management:**
  + Customers shall be able to request to open a Savings, Investment, or Cheque account.
  + Bank Staff shall be required to review and approve all new account requests before they become active.
  + A customer can have multiple accounts of different types.
  + An Investment account requires a minimum initial deposit of BWP 500.00 to be opened.
  + A Cheque account requires employment information (company name and address) to be opened.
  + Savings accounts do not permit withdrawals.
* **Transaction Processing:**
  + Customers shall be able to deposit funds into any of their accounts.
  + Customers shall be able to withdraw funds from Investment and Cheque accounts.
  + Customers shall be able to transfer funds between their own accounts.
  + The system shall support the creation of standing orders for scheduled payments.
  + A daily withdrawal limit of BWP 5,000.00 shall be enforced for Cheque accounts.
  + The system shall prevent any transaction that would result in a negative balance (no overdrafts).
* **Interest Calculation:**
  + The system shall calculate and apply interest at the end of each calendar month.
  + Savings accounts shall receive 0.05% monthly interest.
  + Investment accounts shall receive 5% monthly interest.
  + Interest payments shall be recorded as a transaction in the account history.
  + No partial interest is paid for accounts closed mid-month.
* **Security & Logging:**
  + The system shall log all login attempts (successful and failed) and all financial transactions.
  + User sessions shall timeout after 5 minutes of inactivity.
* **Notifications:**
  + The system shall notify customers via email/SMS for transactions and low balance alerts.
* **Dashboard & UI:**
  + Customers shall have a dashboard view showing all account balances and the five most recent transactions.
  + The system shall have a responsive user interface that works on desktop and mobile devices.
  + All error messages shall be user-friendly and non-technical.

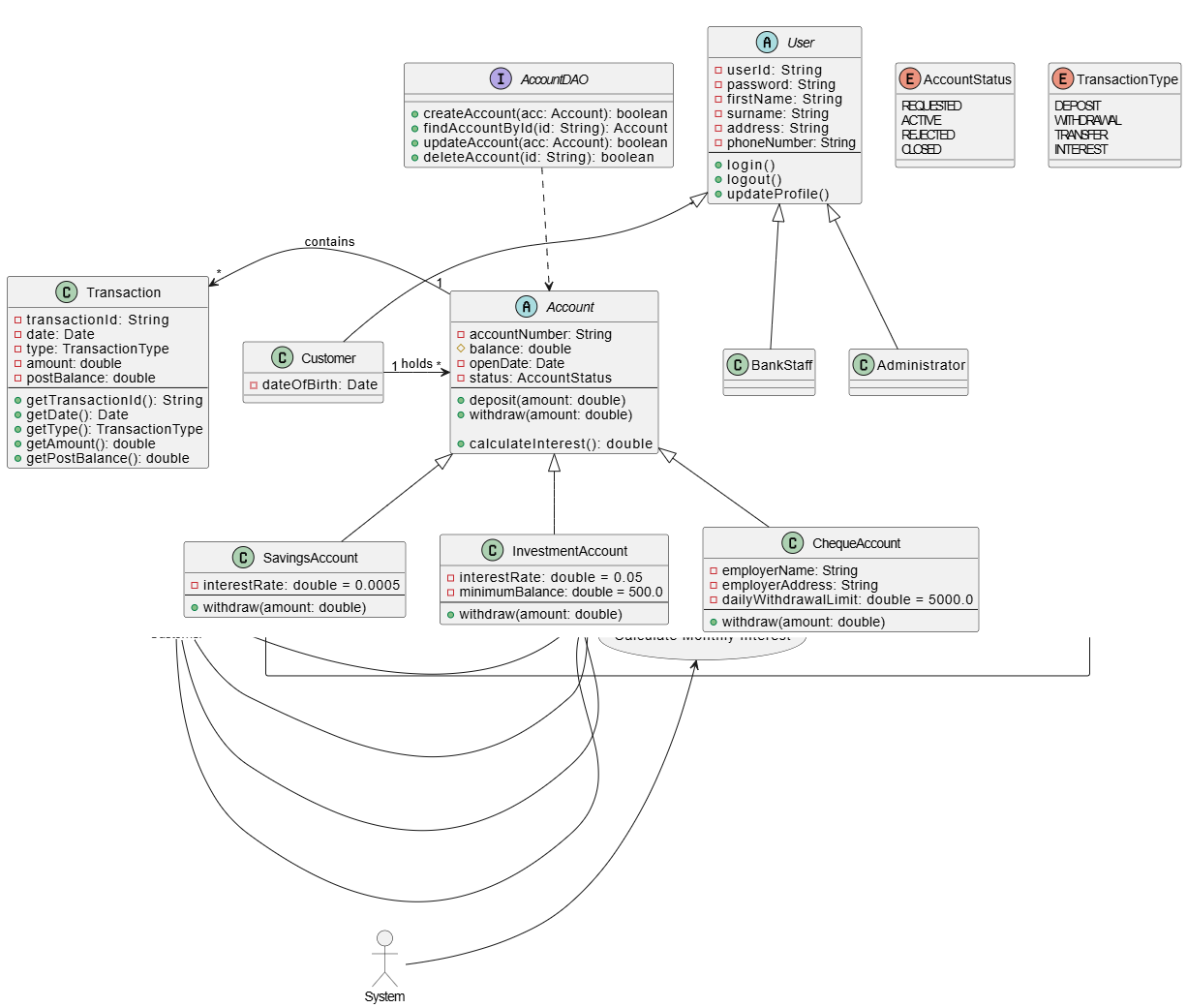
**1.2. Non-Functional Requirements**

* **Security:** The system shall be secure. Access to data and functions shall be strictly controlled based on user roles (principle of least privilege). All passwords shall be stored using strong, salted hashing encryption (e.g., BCrypt). All data transmissions shall be encrypted (HTTPS/SSL).
* **Usability:** The user interface shall be intuitive, consistent, and easy to use for customers with basic computer skills, following common design principles.
* **Performance:** The system should respond to user inputs (e.g., clicking a button, loading a page) in less than 2 seconds under normal load to ensure a smooth user experience.
* **Reliability:** The system must be highly available, targeting 99.5% uptime. Transaction data must be accurate, consistent, and persistent, with no data loss under normal operating conditions.
* **Maintainability:** The code shall be well-structured, modular, and commented following Java coding standards to allow for easy future modifications, feature additions, and debugging by other developers.  
    
  **2. Structural UML Modelling**
* **2.1. System Use Case Diagram**
* **Use Case Diagram for the Banking System**



* **Description:**  
  The Use Case Diagram defines the interactions between the system's actors and its core functionalities.
* **Actors:**
* **Customer:** The primary user who owns and manages bank accounts.
* **Bank Staff:** An employee who manages account applications and assists customers.
* **Administrator:** A super-user who manages system users and settings.
* **System:** An external actor that performs automated tasks like interest calculation.
* **Use Cases & Relationships:** The <<include>> relationship is used for Log Transaction, which is included in Deposit Funds, Withdraw Funds, and Transfer Funds. The <<extend>> relationship is used for Update Profile, which extends Login."
* **Customer Use Cases:** Register, Login, Update Profile, Request New Account, View Account Dashboard, Deposit Funds, Withdraw Funds, Transfer Funds, Set Up Standing Order.
* **Bank Staff Use Cases:** Review New Account Request, Approve/Reject Account, View Customer List.
* **Administrator Use Cases:** Manage User Roles.
* **System Use Cases:** Calculate Monthly Interest, Log Transaction, Send Notification.
* Relationships: <<include>> is used for common sub-functions like logging transactions. <<extend>> is used for optional behaviors like updating a profile, which requires a user to be logged in first.

**Class Diagram for the Banking System**



**Description:**  
The Class Diagram defines the static structure of the system's classes, their attributes, operations, and relationships. "The Account class is abstract and has attributes: accountNumber, balance, openDate. It has methods deposit(), withdraw(), and calculateInterest(). SavingsAccount, InvestmentAccount, and ChequeAccount inherit from Account and override withdraw(). The Transaction class is linked to Account with a one-to-many relationship."

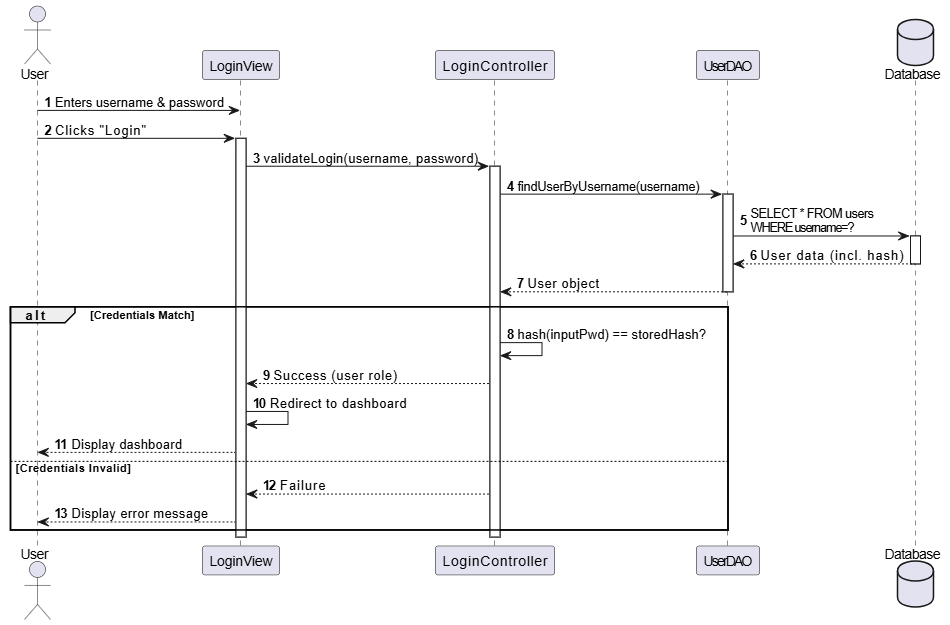
**Core Classes and OOP Principles:**

1. User**(Abstract Class):** Demonstrates **Abstraction**. Base class for all users.
2. Customer**,**BankStaff**,**Admin**:** **Inherit** from User.
3. Account**(Abstract Class):** Demonstrates **Abstraction**. Base class for all accounts.
4. SavingsAccount**,**InvestmentAccount**,**ChequeAccount**:** **Inherit** from Account and **override** the withdraw() method, demonstrating **Polymorphism** and **Method Overriding**.
5. Transaction**:** Demonstrates **Encapsulation** with private attributes and public getters.
6. AccountDAO**(Interface):** Defines a contract for persistence operations. A concrete AccountDAOImpl class would **implement** this interface, promoting loose coupling and adherence to the Dependency Inversion Principle.

**3. Behavioural UML Modelling**

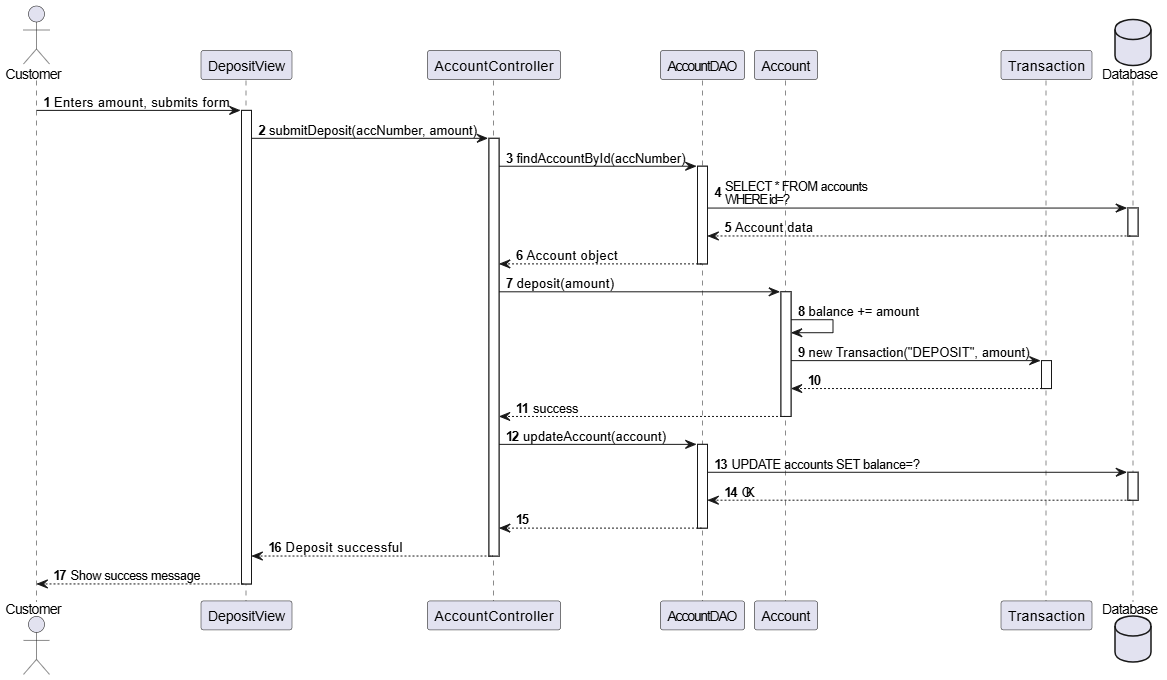
**3.1. Sequence Diagram: Login**

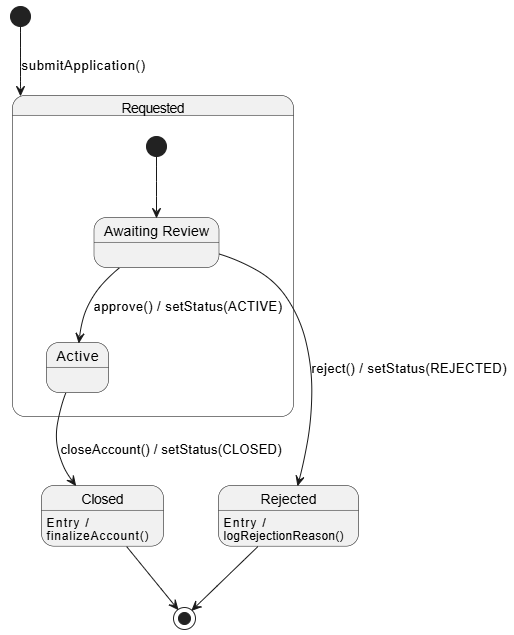
**Figure 3: Sequence Diagram for the Login Use Case**



**Description:** This diagram shows the flow of messages when a user attempts to log in. The LoginController validates the credentials by retrieving the user's data from the database via the UserDAO and comparing password hashes. The autonumber keyword helps trace the chronological order of events.  
  
**3.2. Sequence Diagram: Deposit Funds**

**Figure 4: Sequence Diagram for the Deposit Funds Use Case**

  
  
**Description:** This diagram illustrates the process of a customer depositing money. The AccountController retrieves the account, calls the deposit method on the Account object (which updates its balance and creates a Transaction), and then persists the changes back to the database via the AccountDAO.  
  
**3.3. State Diagram: Account**

**Figure 5: State Diagram for an Account Object**

**Description:** This diagram shows the lifecycle of an Account object. It begins in the **Requested** state. A Bank Staff member can trigger a transition to either **Active** (approved) or **Rejected**. An active account can later be **Closed**. **Rejected** and **Closed** are final states. An account begins in the Requested state. A Bank Staff member can approve() it (transition to Active) or reject() it (transition to Rejected). An active account can be closeAccount()-ed to move to the Closed state. Rejected and Closed are final states."