Student Id: cse24-102

Student names: Arabang Benville Mothofela

Student email: cse24-102@thuto.bac.ac.bw

Cohort: 2025

Assignment title: OBJECT ORIENTED ANALYSIS & DESIGN WITH JAVA

Date of submission: 19 Sep 2025

Programme of Study: COMPUTER SYSTEMS ENGINEERING

Year of Study: 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: A.Mothofela

Date: 19/09/2025

Part A: System Documentation

1. Requirements Elicitation

Functional Requirements

- **Q:** Should the system support different categories of customers (e.g., individuals vs organisations)?
  A: Yes, the system should support both individuals and organisations as customers.
- **Q:** How should the system handle customers who change their details (e.g., new address, job change)?
  A: The system must allow updates to customer profiles while keeping an audit log of changes.
- **Q:** Should the system notify customers of account activities (e.g., SMS/email alerts)?
  A: Yes, at least email alerts should be included for key activities like deposits, withdrawals, or failed transactions.
- **Q:** Should the system allow linking accounts together (e.g., cheque account linked to savings)?
  A: Yes, customers should be able to link accounts for convenience.
- **Q:** If a customer leaves the bank, what happens to their accounts in the system?
  A: Accounts should be marked as 'closed' but the data must remain in the system for reporting and auditing.
- **Q:** Should the system allow joint accounts (two or more owners)?
  A: Yes, joint accounts must be supported for both individuals and organisations.
- **Q:** Do you want the system to log failed transactions (e.g., insufficient funds)?
  A: Yes, failed transactions should be logged for auditing and customer notifications.
- **Q:** Should large transactions require special approval or checks?
  A: Yes, transactions above a configurable threshold should require manager approval.
- **Q:** Should the system generate monthly account statements automatically?
  A: Yes, the system must generate monthly account statements accessible from the customer dashboard.
- **Q:** Should the system allow the bank to change interest rates easily in the future?
  A: Yes, interest rates should be configurable by administrators.
- **Q:** Besides interest, should the system also handle bank charges (e.g., account maintenance fees)?
  A: Yes, maintenance fees and penalties should be supported.
- **Q:** Should customers be able to view how much interest/charges have been applied over time?

A: Yes, a detailed breakdown of interest earned and charges deducted must be visible on customer dashboards.

- **Q:** Should the system include a reporting module (e.g., total deposits, most active customers)?
  A: Yes, reporting is essential for managers and auditors.
- **Q:** Should the system be designed for possible online or mobile access in the future?
  A: Yes, the design should anticipate future online and mobile banking.

Non-Functional Requirements

- **Q:** Who should be allowed to access the system (bank staff only, or customers as well)?
  A: Both — customers should access through a customer dashboard, and employees through an employee dashboard.
- **Q:** Should staff have different access levels (e.g., teller vs manager vs admin)?
  A: Yes, access should be role-based with clear separation of duties.
- **Q:** Should there be audit trails (logs) to track who accessed/changed customer data?
  A: Yes, every transaction and data change must generate an audit trail.
- **Q:** How should sensitive data be secured in the system?
  A: All customer and transaction data must be encrypted in the database.
- **Q:** What is the expected system response time?
  A: The system should have a maximum response time of 3 milliseconds for any transaction.
- **Q:** Should the system prioritize usability (easy navigation, simple displays)?
  A: Yes, it should be easy to use, with dashboards showing all customer accounts at a glance.
- **Q:** Should the system support scalability to handle many customers and transactions?
  A: Yes, it must be scalable to support thousands of customers and transactions simultaneously.

Structural UML Modelling

2.1. System Use Case Diagram

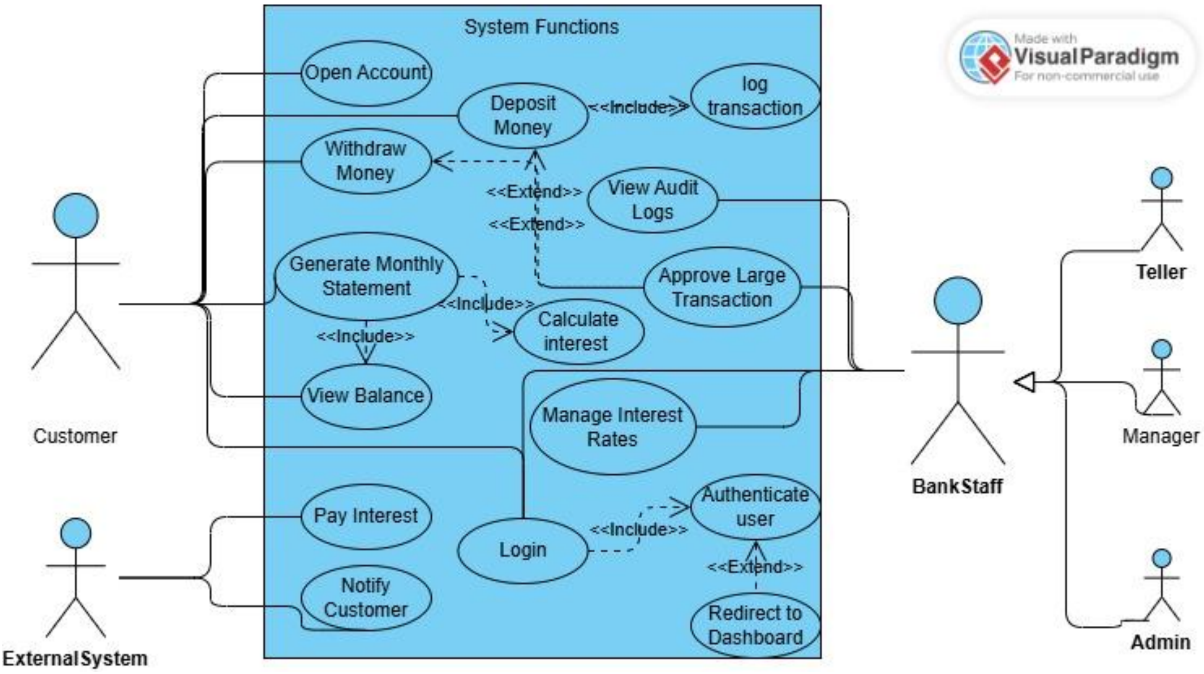

Figure 1: Use Case Diagram for the Banking System

2.2. Class Diagram

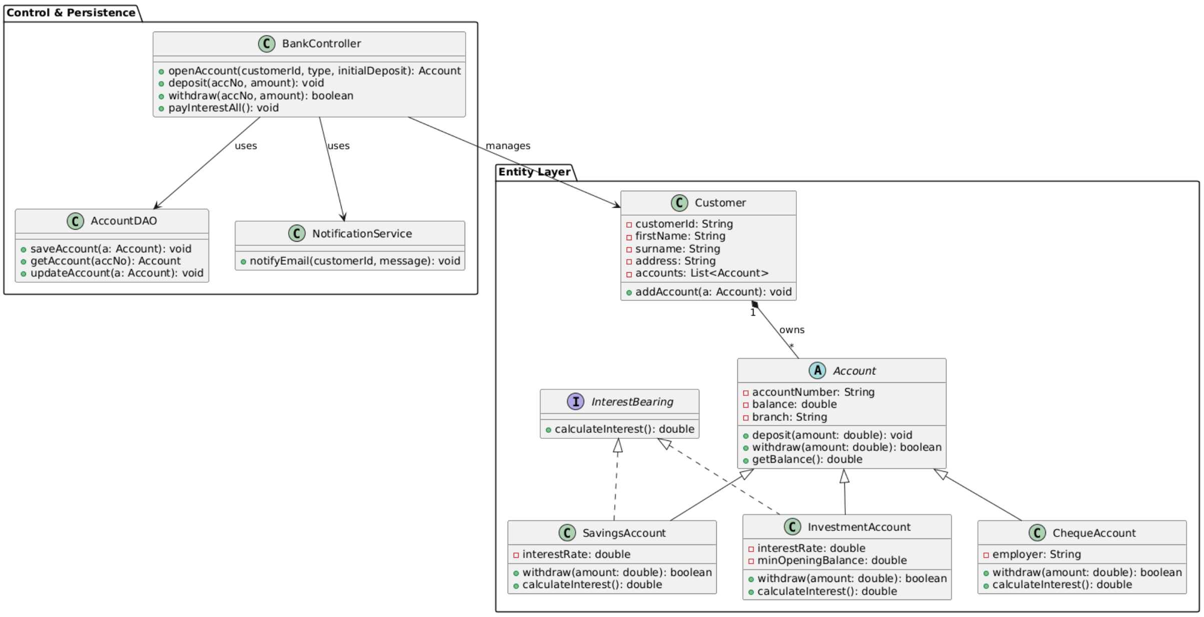
Control & Persistence
C BankController
openAccount(customerId, type, initialDeposit): Account
deposit(accNo, amount): void
withdraw(accNo, amount): boolean
payInterestAll(): void
uses
uses
manages
C AccountDAO
saveAccount(a: Account): void
getAccount(accNo): Account
updateAccount(a: Account): void
C NotificationService
notifyEmail(customerId, message): void
Entity Layer
C Customer
customerId: String
firstName: String
surname: String
address: String
accounts: List<Account>
addAccount(a: Account): void
1
owns
*
A Account
accountNumber: String
balance: double
branch: String
deposit(amount: double): void
withdraw(amount: double): boolean
getBalance(): double
I InterestBearing
calculateInterest(): double
C SavingsAccount
interestRate: double
withdraw(amount: double): boolean
calculateInterest(): double
C InvestmentAccount
interestRate: double
minOpeningBalance: double
withdraw(amount: double): boolean
calculateInterest(): double
C ChequeAccount
employer: String
withdraw(amount: double): boolean
calculateInterest(): double

Figure 2: Class Diagram for the Banking System

3. Behavioural UML Modelling

3.1. Login and Deposit Sequence Diagrams

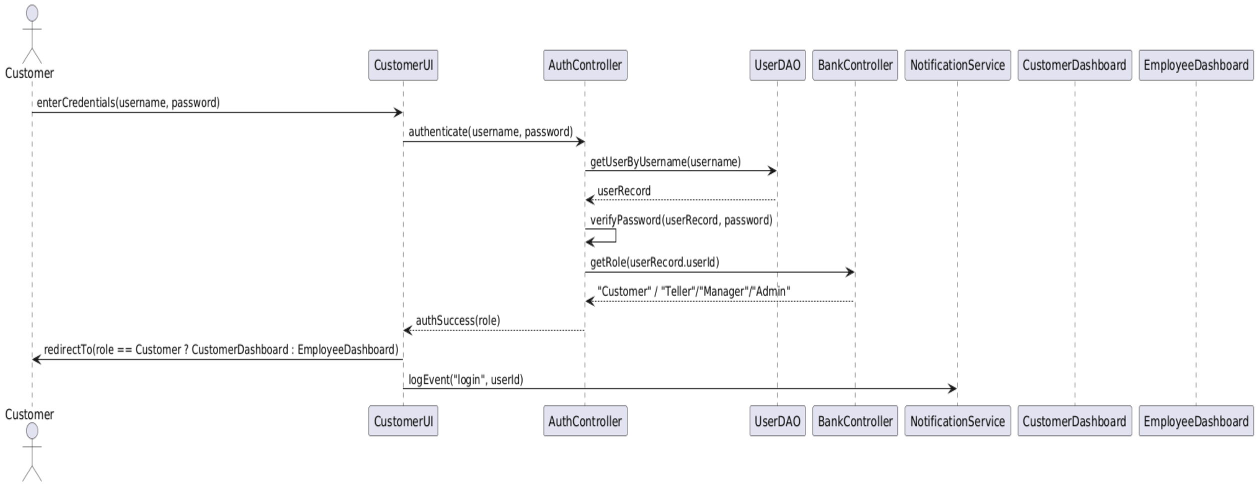
Customer
CustomerUI
AuthController
UserDAO
BankController
NotificationService
CustomerDashboard
EmployeeDashboard
enterCredentials(username, password)
authenticate(username, password)
getUserByUsername(username)
userRecord
verifyPassword(userRecord, password)
getRole(userRecord.userId)
"Customer" / "Teller"/"Manager"/"Admin"
authSuccess(role)
redirectTo(role == Customer ? CustomerDashboard : EmployeeDashboard)
logEvent("login", userId)
Customer
CustomerUI
AuthController
UserDAO
BankController
NotificationService
CustomerDashboard
EmployeeDashboard

Figure 3: Sequence Diagram for Login Use Case

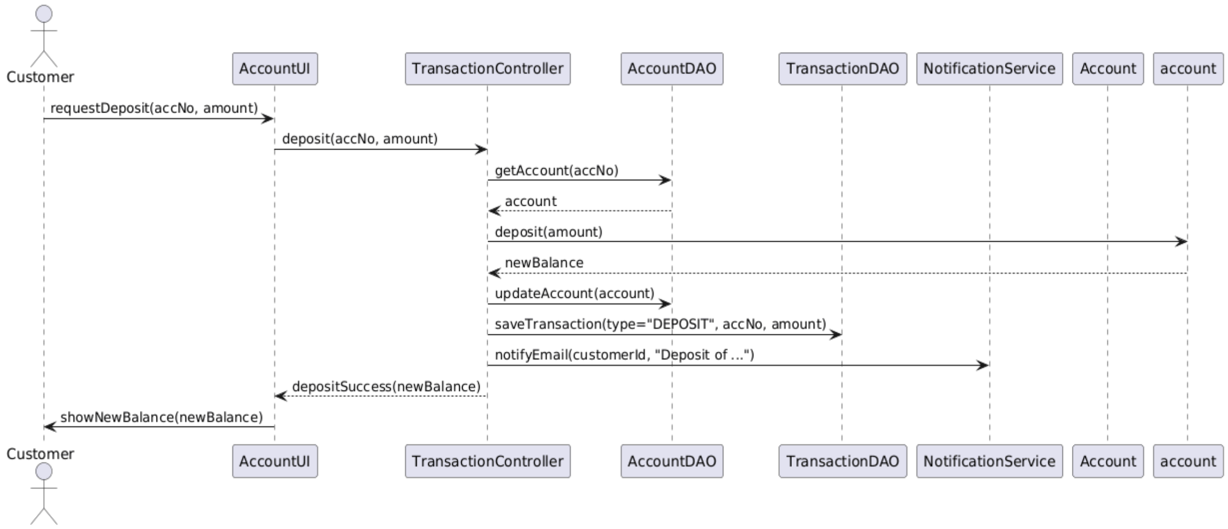

Figure 4: Sequence Diagram for Deposit Funds Use Case

3.2. State Diagram

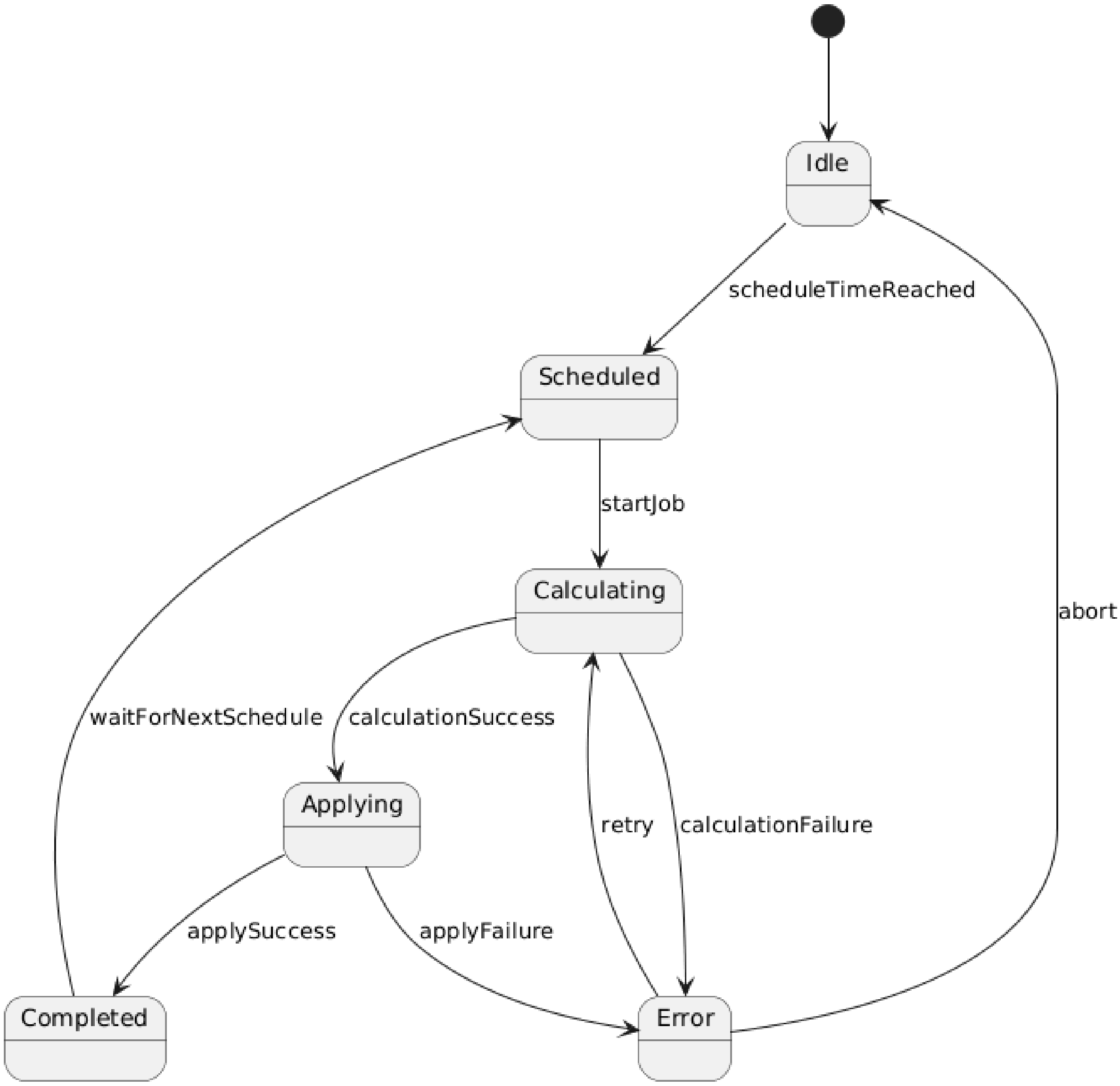

Figure 5: State Diagram for PayInterest Lifecycle

Meeting Record Appendix

Requirements Elicitation Interview Record

Interview Details

- Interviewer: Arabang B. Mothofela
- Interviewee (Client): Themba Moeng
- Date: 17 September 2025
- Time: 15:20–15:28
- Place: ONLINE

Transcript of Interview:

Functional Requirements

- **Q:** Should the system support different categories of customers (e.g., individuals vs organisations)?
  A: Yes, the system should support both individuals and organisations as customers.
- **Q:** How should the system handle customers who change their details (e.g., new address, job change)?
  A: The system must allow updates to customer profiles while keeping an audit log of changes.
- **Q:** Should the system notify customers of account activities (e.g., SMS/email alerts)?
  A: Yes, at least email alerts should be included for key activities like deposits, withdrawals, or failed transactions.
- **Q:** Should the system allow linking accounts together (e.g., cheque account linked to savings)?
  A: Yes, customers should be able to link accounts for convenience.
- **Q:** If a customer leaves the bank, what happens to their accounts in the system?
  A: Accounts should be marked as 'closed' but the data must remain in the system for reporting and auditing.
- **Q:** Should the system allow joint accounts (two or more owners)?
  A: Yes, joint accounts must be supported for both individuals and organisations.
- **Q:** Do you want the system to log failed transactions (e.g., insufficient funds)?
  A: Yes, failed transactions should be logged for auditing and customer notifications.
- **Q:** Should large transactions require special approval or checks?
  A: Yes, transactions above a configurable threshold should require manager approval.

- **Q:** Should the system generate monthly account statements automatically?
  A: Yes, the system must generate monthly account statements accessible from the customer dashboard.
- **Q:** Should the system allow the bank to change interest rates easily in the future?
  A: Yes, interest rates should be configurable by administrators.
- **Q:** Besides interest, should the system also handle bank charges (e.g., account maintenance fees)?
  A: Yes, maintenance fees and penalties should be supported.
- **Q:** Should customers be able to view how much interest/charges have been applied over time?
  A: Yes, a detailed breakdown of interest earned and charges deducted must be visible on customer dashboards.
- **Q:** Should the system include a reporting module (e.g., total deposits, most active customers)?
  A: Yes, reporting is essential for managers and auditors.
- **Q:** Should the system be designed for possible online or mobile access in the future?
  A: Yes, the design should anticipate future online and mobile banking.

Non-Functional Requirements

- **Q:** Who should be allowed to access the system (bank staff only, or customers as well)?
  A: Both — customers should access through a customer dashboard, and employees through an employee dashboard.
- **Q:** Should staff have different access levels (e.g., teller vs manager vs admin)?
  A: Yes, access should be role-based with clear separation of duties.
- **Q:** Should there be audit trails (logs) to track who accessed/changed customer data?
  A: Yes, every transaction and data change must generate an audit trail.
- **Q:** How should sensitive data be secured in the system?
  A: All customer and transaction data must be encrypted in the database.
- **Q:** What is the expected system response time?
  A: The system should have a maximum response time of 3 milliseconds for any transaction.
- **Q:** Should the system prioritize usability (easy navigation, simple displays)?
  A: Yes, it should be easy to use, with dashboards showing all customer accounts at a glance.

- **Q:** Should the system support scalability to handle many customers and transactions?
  A: Yes, it must be scalable to support thousands of customers and transactions simultaneously.