Cse24-152 Festus L.A Ntala

REQUIREMENTS ELICITATION

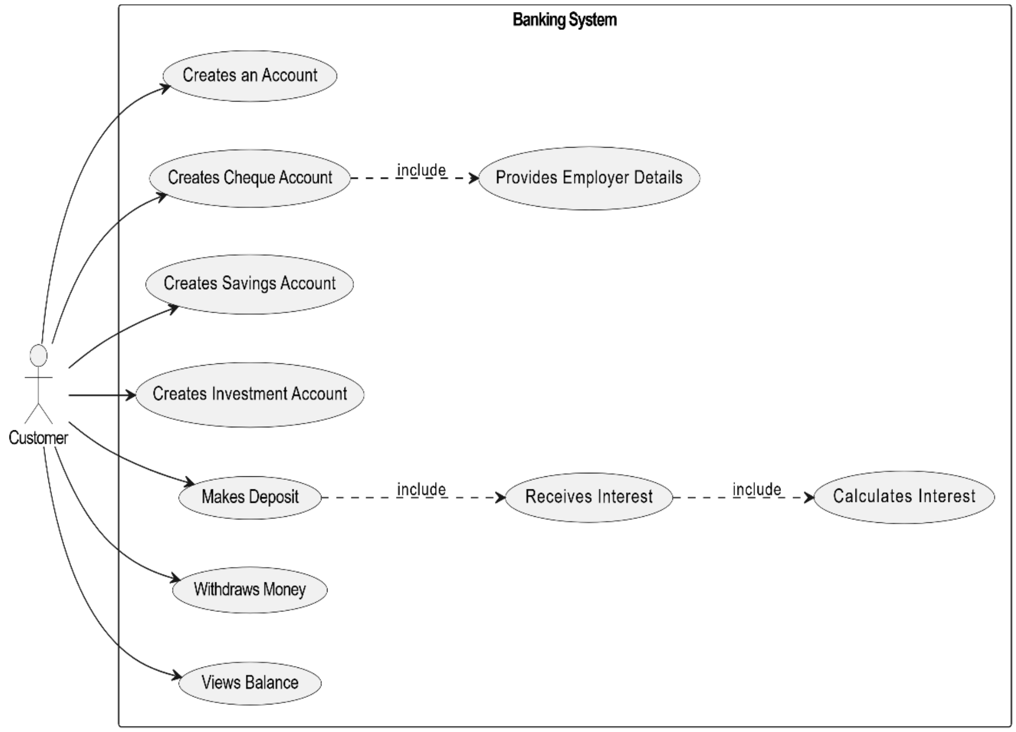
FUNCTIONAL REQUIREMENTS

1. **Customer Management:**
   - The system shall allow a new customer to register by providing their first name, surname, and address.
   - The system shall allow a customer to open a new account. The type of account must be specified. The account is can be a Savings, Investment or a Cheque account.
   - The system shall enforce account-specific opening rules:
     - **Investment Account:** Requires an initial deposit of at least BWP 500.00.
2. **Account Operations:**
   - The system shall allow a customer to deposit funds into any of their existing accounts.
   - The system shall allow a customer to withdraw funds from their **Investment** or **Cheque** accounts.
   - The system shall **not** allow withdrawals from a **Savings** account.
3. **Interest Calculation:**
   - The system shall automatically calculate and add monthly interest to applicable accounts.
     - **Investment Account:** 5% monthly interest on the balance.
     - **Savings Account:** 0.05% monthly interest on the balance.
     - **Cheque Account:** No interest.
4. **Viewing Information:**
   - The system shall allow a customer to view their profile details.
   - The system shall allow a customer to view the balance and details of all their accounts.

NON-FUNCTIONAL REQUIREMENT

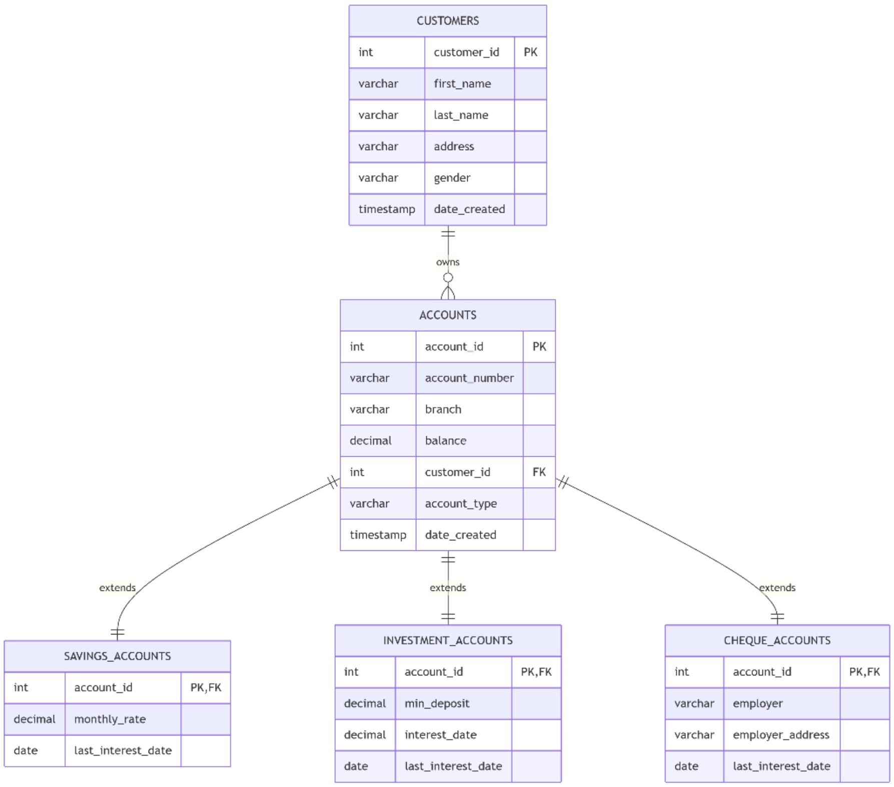
- **Usability:** The graphical user interface shall be intuitive and easy to navigate for users with basic computer skills. Error messages shall be clear and informative.
- **Performance:** The system should respond to user actions in less than 2 seconds.
- **Security:** The system must validate all user input to prevent invalid data from being processed.
- **Maintainability:** The code shall be well-structured using OOP principles to allow for easy future modifications.
- **Reliability:** The system must accurately calculate and apply interest and must not lose transaction data once committed.

System Use Case Diagram



- **Relationships:** The Customer actor is linked to all use cases.

Class Diagram



CUSTOMERS		
int	customer_id	PK
varchar	first_name	
varchar	last_name	
varchar	address	
varchar	gender	
timestamp	date_created	

ACCOUNTS		
int	account_id	PK
varchar	account_number	
varchar	branch	
decimal	balance	
int	customer_id	FK
varchar	account_type	
timestamp	date_created	

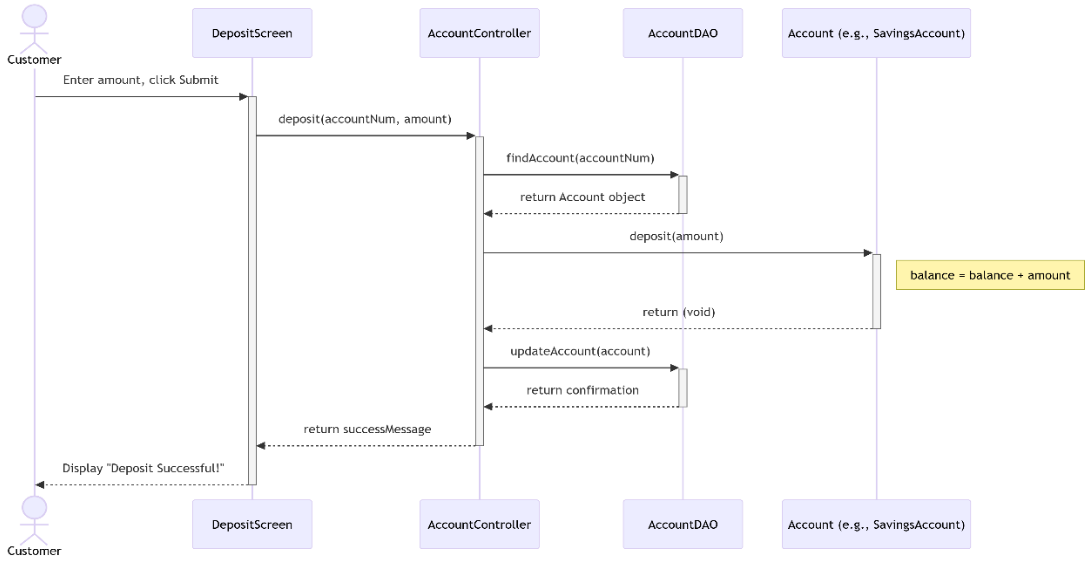
SAVINGS_ACCOUNTS		
int	account_id	PK,FK
decimal	monthly_rate	
date	last_interest_date	

INVESTMENT_ACCOUNTS		
int	account_id	PK,FK
decimal	min_deposit	
decimal	interest_date	
date	last_interest_date	

CHEQUE_ACCOUNTS		
int	account_id	PK,FK
varchar	employer	
varchar	employer_address	
date	last_interest_date	

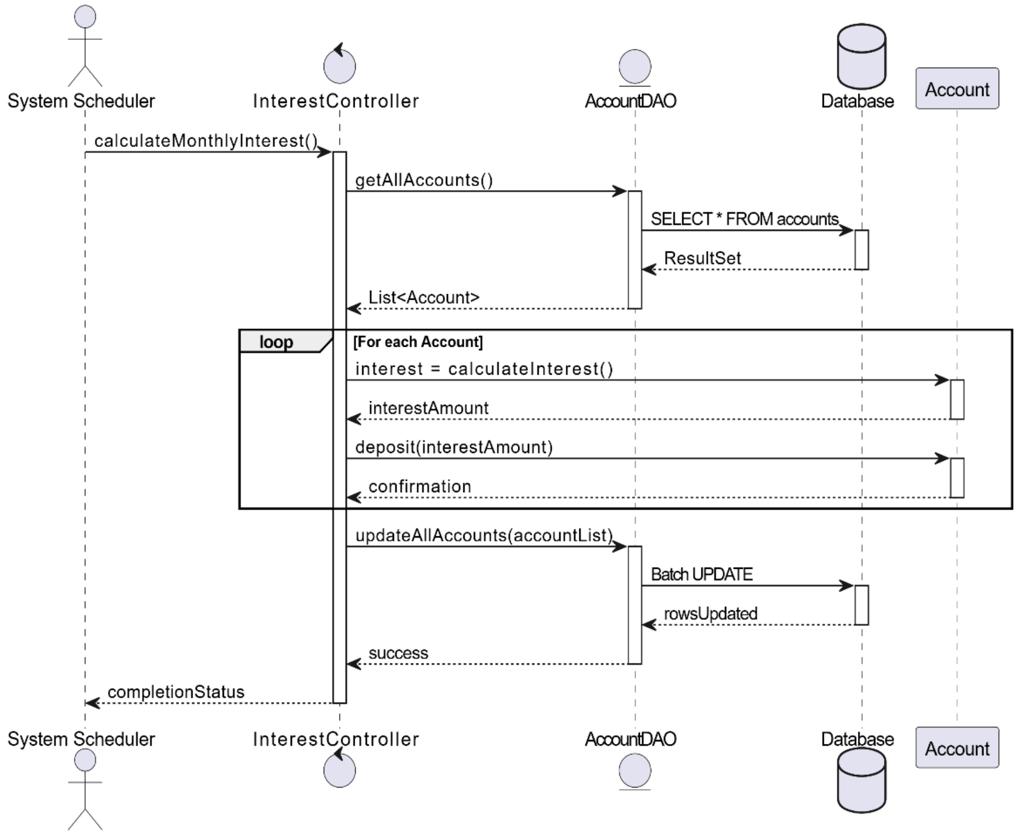
Behavioural UML Modelling

Sequence Diagram: "Deposit Funds"



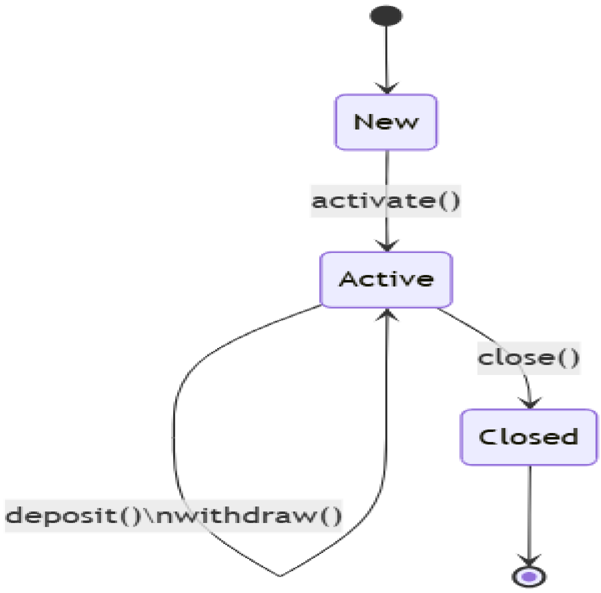
1. Customer enters amount and submits on the DepositView.
2. DepositView sends depositRequest(accountNumber, amount) to AccountController.
3. AccountController calls findAccount(accountNumber) on AccountDAO.
4. AccountDAO returns the Account object.
5. AccountController calls account.deposit(amount).
6. The Account object updates its internal balance.
7. AccountController calls updateAccount(account) on AccountDAO to persist the new balance.
8. AccountDAO confirms the update.
9. AccountController sends a success message to DepositView.
10. DepositView displays a confirmation message to the Customer.

Sequence Diagram: "Calculate Monthly Interest"



1. InterestScheduler triggers on the last day of the month.
2. It calls calculateMonthlyInterest() on AccountController.
3. AccountController calls getAllAccounts() on AccountDAO.
4. AccountDAO returns a List<Account>.
5. For each Account in the list, AccountController calls account.calculateInterest().
6. Each Account object calculates its specific interest amount.
7. The AccountController then calls account.deposit(interestAmount) for each account.
8. The Account object updates its balance.
9. AccountController calls updateAccount(account) on AccountDAO for each updated account to save the new balance.

10. **State Diagram: "Account"**



States:

- **New:** The account object has been created but not yet persisted. (Initial State)
- **Active:** The account is open and operational. All valid transactions are permitted.
- **Closed:** The account has been permanently closed. No further actions are possible.

Transitions:

- **New -> Active:** save() / activate()
- **Active -> Closed:** close()

MEETING RECORD APPENDIX

Interview Record: Banking System Requirements Elicitation

Date and Time:18th September 2025, 10:30am

Place:Virtual Meeting (Microsoft Teams)

Cse24-152 Festus L.A Ntala

Interviewee: Mr. Kentsenao Baseki, OOAD Lecture

Interviewer: Festus Ntala and other CSE Students, Systems Analysis Students

Purpose of Interview: To elicit and clarify functional and non-functional requirements for the development of a new Banking System.

Questions Asked & Responses

Q1: Could you briefly describe the main types of accounts your bank offers?

We currently offer three account types: Savings, Investment, and Cheque accounts. Each serves different customer needs and has specific rules regarding deposits, withdrawals, and interest.

Q2: What are the key features of a Savings Account?

A Savings Account allows customers to deposit funds for future use. It earns a monthly interest but does not allow any withdrawals. It's designed to encourage saving.

Q3: What about Investment Accounts?

Investment Accounts offer higher returns—5% monthly interest. They require an initial deposit of at least BWP 500. Customers can both deposit and withdraw from this account.

Q4: And Cheque Accounts?

Cheque Accounts are typically used for salary credits. They allow unlimited deposits and withdrawals. To open one, the customer must be employed and provide employer details—company name and address. No interest is paid on these accounts.

Q5: Can a single customer hold multiple accounts?

Absolutely. A customer can have one or more accounts of any type—Savings, Investment, or Cheque—simultaneously.

Q6: What core operations should the system support?

The system must allow customers to open accounts, deposit funds, and withdraw where applicable. It should also automatically calculate and credit interest monthly for Savings and Investment Accounts.

Q7: Are there any security or usability concerns we should consider?

Yes. The system must be user-friendly and secure. Input validation is crucial, for example, preventing negative deposits. In the future, we'd also want user authentication and encryption.

Q8: How should the system handle interest calculation?

Interest should be calculated and applied automatically at the end of each month: 5% for Investment Accounts and 0.05% for Savings Accounts. Cheque Accounts earn no interest.

Q9: Who are the primary users of this system?

Initially, the bank's customers. However, bank staff may also use it for administrative purposes like account management and oversight.

Q10: Is there anything else you'd like the system to do?

For now, those are the essentials. We want a clean, working system that handles accounts, transactions, and interest correctly. Additional features like loans or transfers can be added later.

Interview Conclusion:

Cse24-152 Festus L.A Ntala

The interview successfully clarified the core requirements and business rules for the Banking System. The responses will directly inform the functional and non-functional specifications, use cases, and class designs for the assignment.

Date: 18 September 2025

https://mybac.sharepoint.com/sites/20252026cse202objectorientedanalysisanddevelopmentwithjav/Shared%20Documents/General/Recordings/OOAD%20Elicitation%20Session-20250918_103316-Meeting%20Recording.mp4?web=1