# **Elicitation Session Report – Banking System (CSE24-096)**

**Date: September 18, 2025  
Time: 08:33 – 10:17 AM  
Location: Virtual (Microsoft Teams)  
Interviewer(s): Pitso Kefhitilwe & Peer Students  
Interviewee (SME): Kentsenao Baseki (Lecturer)  
Purpose:** To formally elicit and clarify functional and non-functional requirements for the Banking System assignment.

**Key Questions and Clarifications**

|  |  |  |
| --- | --- | --- |
| Topic | Question (Asked by) | Response / Clarification |
| Currency & Limits | Pitso: “Should we stick to USD or Pula? Limits on deposits/withdrawals?” | Any currency may be used (e.g., Pula). Focus is on data structure. Minimum balances: Savings (50), Investment (500), Cheque (0). |
| Customer Identification | Pitso: “What unique identifier should companies use?” | Companies use a registration number (e.g., BW001). Individuals use National ID. |
| Interest Calculation | Olerato: “Are interest calculations automated?” | Yes, fully automated. |
| Polymorphic Interest | Tadiwanashe: “Should interest be polymorphic?” | Yes, implement via interface (InterestBearing). |
| Monthly Statements | Amantle: “Are statements generated monthly?” | Yes, represented through transaction history. |
| Login Security | Felix: “Should we add login attempt limits?” | Not required; keep scope on core OO structure. |
| Extra Security | Alex: “Can additional security features be added?” | Not advised; maintain focus on assignment scope. |

**Agreed Outcomes**

1. Functional Requirements
   1. Bank staff can open new accounts (Savings, Investment, Cheque).
   2. Customers can deposit funds into any account.
   3. Customers can withdraw funds from Investment and Cheque accounts (Savings does not allow withdrawals).
   4. Customers can view account balances and transaction history.
   5. Automated monthly interest calculations for applicable accounts.
2. Non-Functional Requirements
   1. Usability: Intuitive, modern UI (JavaFX).
   2. Security: Basic login authentication for bank staff.
   3. Maintainability: Adherence to MVC architecture and OOP principles (inheritance, polymorphism, encapsulation).
   4. Reliability: Accurate transaction processing and consistent interest calculation.

**Reflection & Analysis**

The session:

* Reinforced that interest calculations are automated and recorded without manual intervention.
* Confirmed that monthly statements are represented as transaction history.
* Emphasized that the project scope prioritizes core OO principles over extended features.
* Provided clarity on minimum balances and currency flexibility, ensuring consistent design decisions.

Impact on System Design:

* The automated interest and clearly defined minimum balances simplify the Account class design.
* Restricting account creation to bank staff influences the Customer-Account relationship management and login access logic.
* Focus on OO principles ensures a clean, maintainable architecture suitable for future extensions.

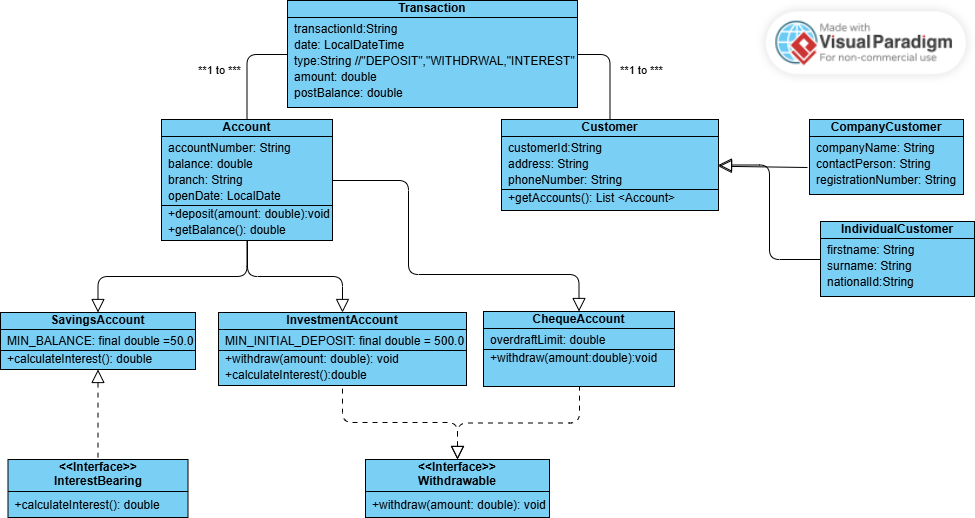
**Structural UML Modelling**

**Use Case Diagram**

A diagram of a customer

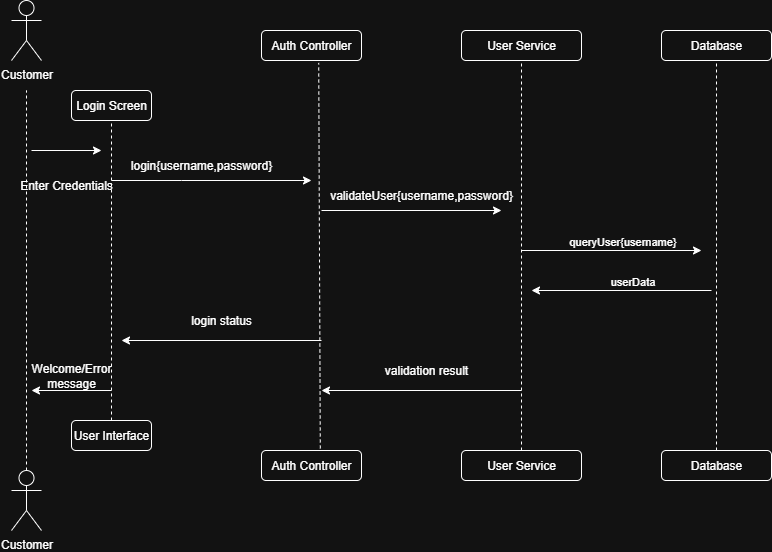
AI-generated content may be incorrect.

**Class Diagram**

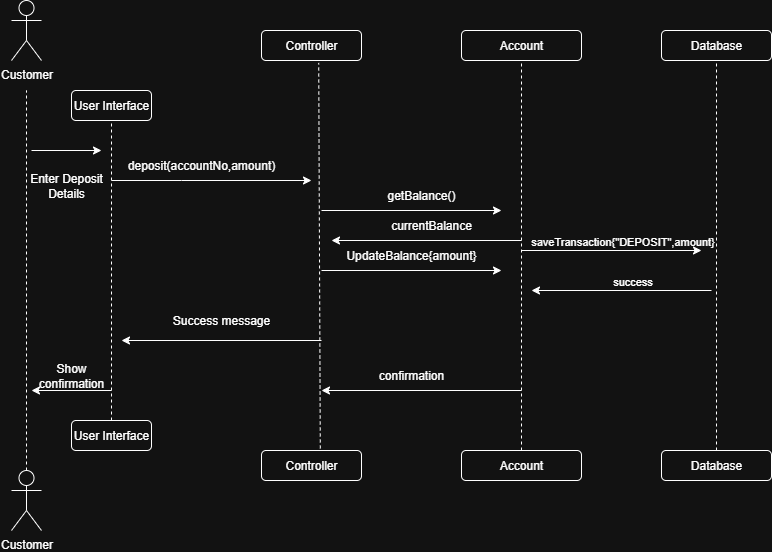


**Behavioural UML Modelling**

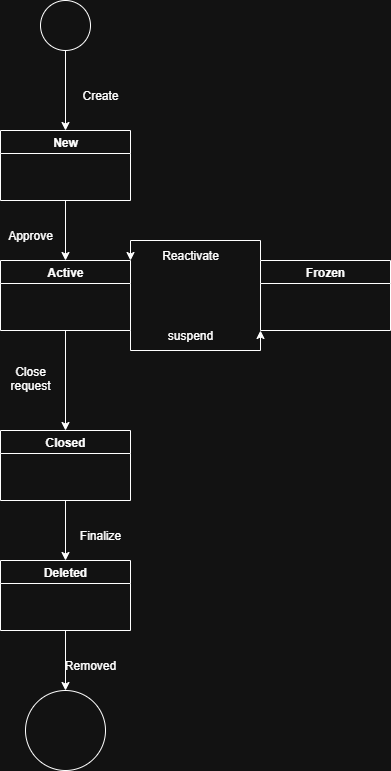
**Sequence Diagram Login**



**Sequence Diagram Deposit**

s

**State Diagram Account Creation**



**State Diagram Account Creation**

