**📘 Part A: System Documentation**

**1. Requirements Elicitation (10 marks)**

**1.1 Functional Requirements (5 marks)**

Based on the interview with the “client,” the Banking System must:

1. Allow customers to **register** and securely **log in**.
2. Enable customers to **open multiple types of accounts** (Savings, Investment, Cheque).
3. Allow **deposits** into all account types.
4. Allow **withdrawals** where permitted (Investment & Cheque only).
5. Automatically **calculate and pay interest**:
   * Savings Account: 0.05% monthly
   * Investment Account: 5% monthly
6. Provide **transaction history** for each account.
7. Link **multiple accounts** to one customer.

**An appendicle of the interview Record**

**The interview was held on teams by Tutor Themba Moeng**

**AI-generated content may be incorrect**

****

**Laressa Legakwa Motonto**

**started transcription**

**Themba Moeng**

**0 minutes 4 seconds0:04**

**Themba Moeng**

**If you mention a question that is just, I'll just say it's addressed and if necessary I may provide additional information, but otherwise you are free to continue.**

**Laressa Legakwa Motonto**

**The functional requirements what the system should do, must do. Are there conditions or are there conditions or restriction for obeying certain accounts?**

**Themba Moeng**

**Mm.**

**Themba Moeng**

**Yeah.**

**Themba Moeng**

**Well, there are no restrictions. It's just a matter of us having what provided the necessary details and where applicable, having the necessary deposit for certain types of accounts as per the scenario.**

**Laressa Legakwa Motonto**

**OK.**

**Themba Moeng**

**Speak.**

**Laressa Legakwa Motonto**

**Book.**

**Laressa Legakwa Motonto**

**Do you need to like support different branches or this one branch only only one branch?**

**Themba Moeng**

**Yeah, yeah. Since it's a desktop application, it will just be applicable to one branch, yes.**

**Laressa Legakwa Motonto**

**OK.**

**Laressa Legakwa Motonto**

**Uh, what? What are the key operation that the customers should be able to perform on the accounts?**

**Themba Moeng**

**OK, so once they have logged in or lastly they should be able to search for a specific account. Remember a customer can have multiple accounts, so they could be a drop down list of the accounts that they have or they could search by using an account number or account type.**

**Laressa Legakwa Motonto**

**I'm just.**

**Themba Moeng**

**And then having you found the necessary account, they should be able to withdraw to do deposit where applicable and then also to view the balance. And then also they should be able to see the transaction history after having specified the starting date.**

**Laressa Legakwa Motonto**

**Good.**

**Themba Moeng**

**And the ending date.**

**Laressa Legakwa Motonto**

**OK, another question, it's about the the non-functional, how the system has performed and I wanted to ask what are the expectation for the system's performance ?**

**Themba Moeng**

**Yes.**

**Themba Moeng**

**Yes.**

**Laressa Legakwa Motonto**

**For example, the speed of transactions.**

**Themba Moeng**

**Yes. So basically when you are interacting with the system, we expect a maximum response time of three milliseconds.**

**Themba Moeng**

**And then obviously in terms of throughput, we expect to be able to handle 200 concurrent transactions.**

**Laressa Legakwa Motonto**

**OK.**

**Laressa Legakwa Motonto**

**OK.**

**Laressa Legakwa Motonto**

**OK, another question. What level of security should the system provide, especially around that login and account access?**

**Themba Moeng**

**Yes. So basically all users of the system obviously must have credentials created that is customers as well as employees within the bank. So these.**

**Laressa Legakwa Motonto 3 minutes**

**But.**

**Themba Moeng**

**Must have a minimum password length of eight characters, which could be a combination of alphanumeric characters and then.**

**Themba Moeng**

**Yeah, so then also when storing the information in the database, it should be encrypted, meaning when you look physically into your database tables, you should not see plain text but scrambled text and then only the data is visible through the application after logging on.**

**Laressa Legakwa Motonto**

**OK.**

**Laressa Legakwa Motonto**

**Yes, the last question, how many customers should the system be able to support at once, like for example like scalability?**

**Themba Moeng**

**Yes, basically it should cater for however many customers as possible through through the database, yes.**

**Laressa Legakwa Motonto**

**Thank you.**

**Laressa Legakwa Motonto**

**OK.**

**Laressa Legakwa Motonto**

**Yes, Sir.**

**Laressa Legakwa Motonto**

**I think I don't have any further question.**

**Themba Moeng**

**OK. well done. Thank you very much. All the best.**

**Laressa Legakwa Motonto**

**OK.**

**Laressa Legakwa Motonto**

**Yes, thank you, Sir.**

**Themba Moeng**

**Bye, bye.**

**Laressa Legakwa Motonto**

**Yes, Sir.**

**1.2 Non-Functional Requirements (5 marks)**

The Banking System must also satisfy:

* **Security:** Customers must log in with valid credentials; sensitive data must be protected.
* **Performance:** Transactions (deposit, withdraw, login) must respond within 2 seconds.
* **Usability:** The interface must be simple and user-friendly.
* **Reliability:** Data must be consistent and protected against errors or crashes.
* **Scalability:** The system should support many customers and accounts as the bank grows.

**2. Structural UML Modelling (20 marks)**

**2.1 Use Case Diagram (10 marks)**

**Actors:** Customer, Bank System.  
**Use Cases:** Register/Login, Open Account, Deposit, Withdraw, View Balance, View Transactions, Receive Interest

A diagram of a customer

AI-generated content may be incorrect.

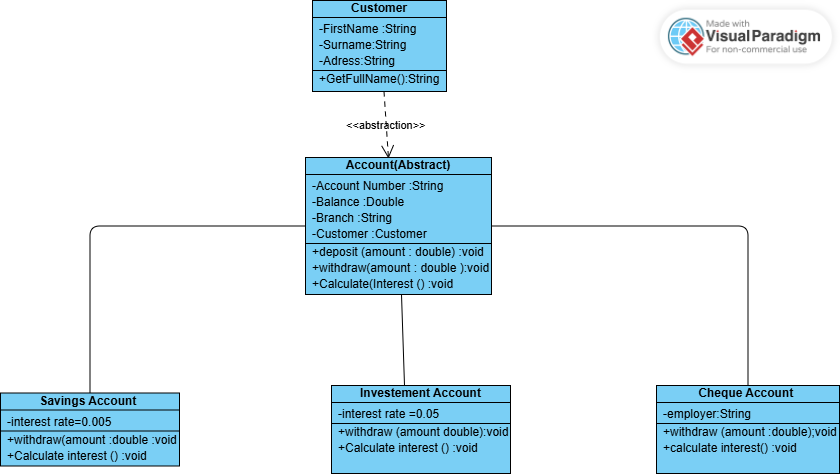
**2.2 Class Diagram (10 marks)**

**Main Classes:**

* Customer
* Account (abstract) → SavingsAccount, InvestmentAccount, ChequeAccount
* Transaction
* InterestBearing (interface)

OOP Principles:

* **Abstraction:** Account is abstract.
* **Inheritance:** Account subclasses (Savings, Investment, Cheque).
* **Encapsulation:** Private attributes with getters/setters.
* **Polymorphism:** Methods like withdraw() behave differently per account.
* **Interface:** InterestBearing defines calculateInterest().

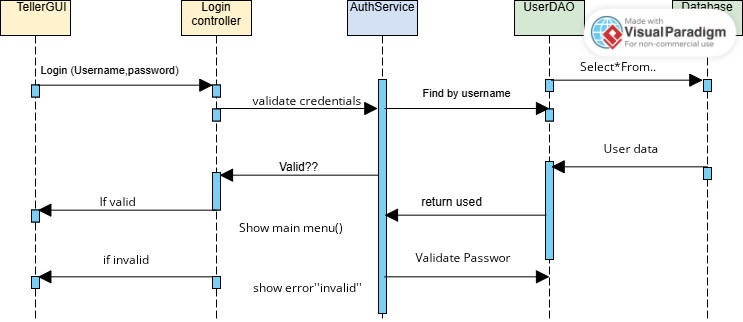


**3. Behavioural UML Modelling (10 marks)**

**3.1 Sequence Diagrams (6 marks)**

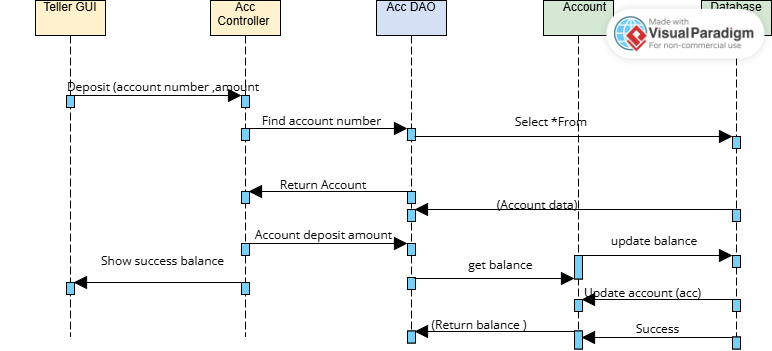
**(a) Login Use Case:**

* Customer enters login info → GUI (LoginView) → Controller (LoginController) → DAO → Database → back to Customer with result.



**(b) Deposit Funds Use Case:**

* Customer inputs amount → AccountView → AccountController → Account → Transaction recorded → Database updated → confirmation returned to Customer.



**3.2 State Diagram (4 marks)**

For **Pay Interest process**:

* **States:** Account Active → Interest Calculation → Update Balance → Interest Paid → back to Active.

