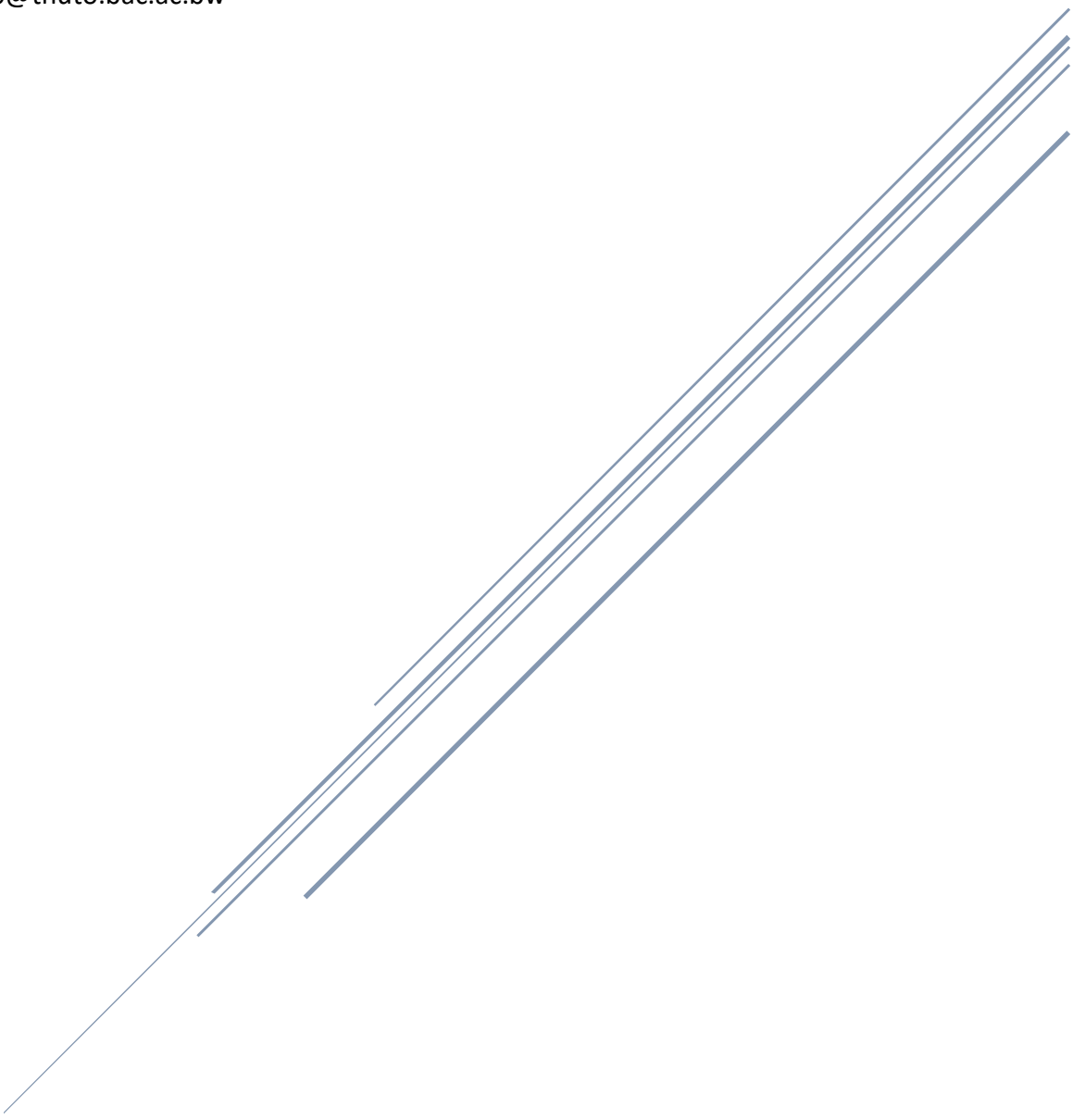
Requirements Elicitation

NAME: LUCKY KATLEGO CHINAYE

Cse24-150@thuto.bac.ac.bw



Introduction

Requirements elicitation is the starting point of system analysis and design. It ensures that both the **functional needs** (what the system must do) and the **non-functional qualities** (how well the system performs) are captured. For this banking system, requirements were gathered from stakeholder interviews, supported by use case diagrams, class diagrams, and sequence diagrams. The aim is to develop a reliable and secure system that manages customer accounts, supports transactions, and enforces business rules.

Functional Requirements (FR)

Authentication and User Management

FR1: The system shall allow new customers to register their details.

FR2: The system shall allow users (customers and bank staff) to log in with a username and password.

FR3: The system shall verify login credentials against the database before access is granted.

FR4: The system should allow administrators (bank staff) to manage customer accounts.

Account Management

FR5: The system shall allow opening of different account types (Savings, Cheque, Investment).

FR6: The system shall allow staff to approve customer accounts after verification.

FR7: The system shall store account details, including account type, balance, and status.

FR8: The system shall allow closing of accounts when requested and approved.

Transactions

FR9: The system shall allow deposits into accounts.

FR10: The system shall allow withdrawals, subject to account type rules and available balance.

FR11: The system shall allow transfer of funds between accounts.

FR12: The system shall calculate interest in savings and investment accounts and update balances.

FR13: The system shall maintain a complete transaction history (date, time, type, amount).

FR14: The system shall display updated balances after every successful transaction.

Reporting and Monitoring

FR15: The system shall allow customers to view their transaction history.
FR16: The system should provide bank staff with reports on accounts and transactions.

Non-Functional Requirements (NFR)

Security

NFR1: User passwords must be stored in encrypted form.

NFR2: Access must be role-based (customer, staff, admin).

NFR3: All sensitive data transfers must use secure communication.

Reliability and Availability

NFR4: The system must ensure consistent recording of all transactions.

NFR5: The system should recover gracefully from system or network failures.

Performance

NFR6: Login validation must complete in less than 3 seconds.

NFR7: Balance updates after a transaction must be reflected immediately.

Usability

NFR8: The interface should be simple and user-friendly for both staff and customers.

NFR9: Error messages should be clear and guide the user to correct input.

Maintainability

NFR10: The system must be easy to extend with new account types or transaction rules in the future.

Business Rules (BR)

BR1: Savings accounts do not allow withdrawals beyond the deposited balance.

BR2: Investment accounts require a minimum initial deposit of 500.

BR3: Cheque accounts require employer details for approval.

BR4: All transactions (deposit, withdrawal, transfer, interest) must be logged in transaction history.

BR5: Only bank staff can approve new customer accounts.

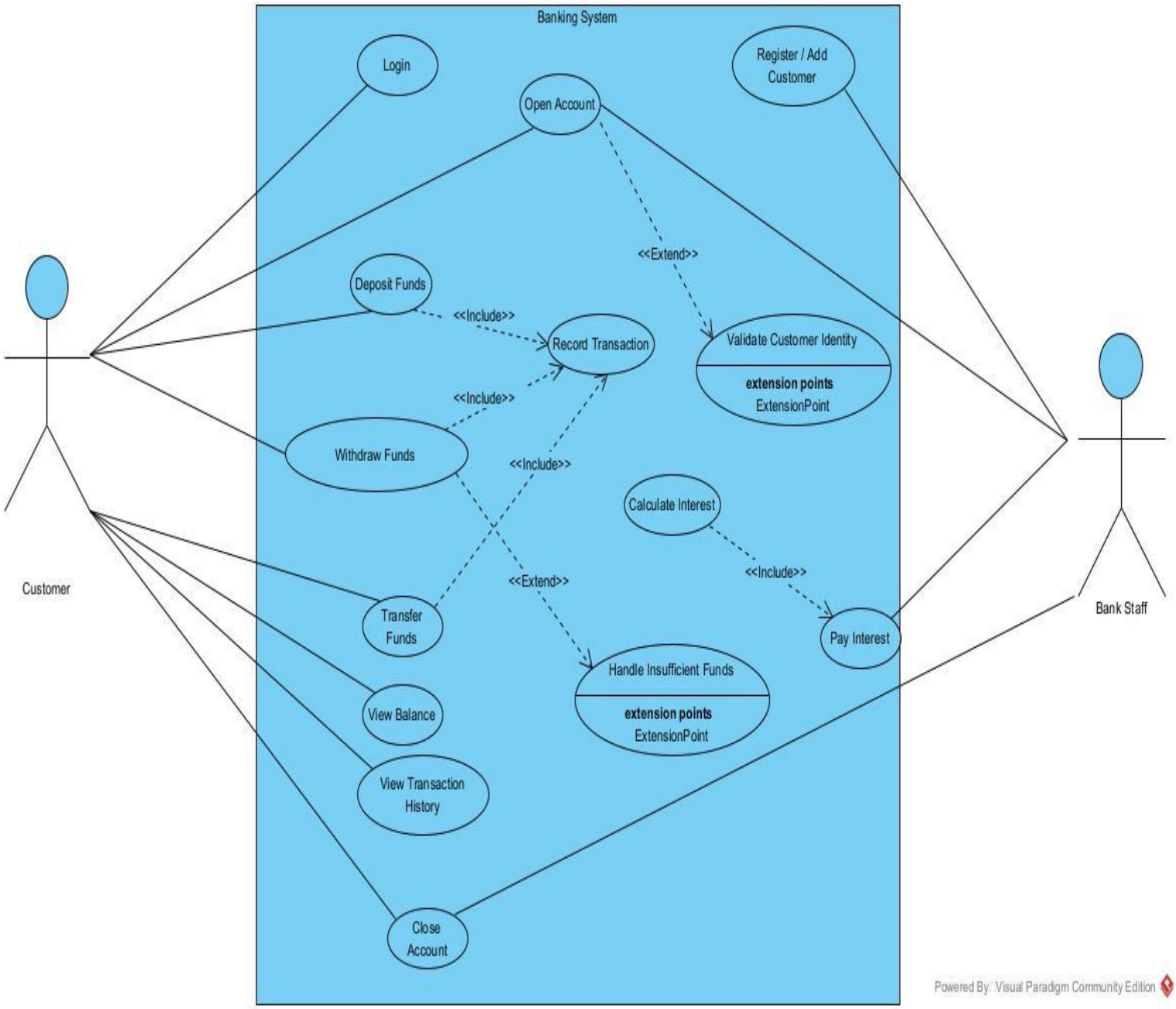
BR6: Passwords must always be encrypted in the database.

Traceability Matrix

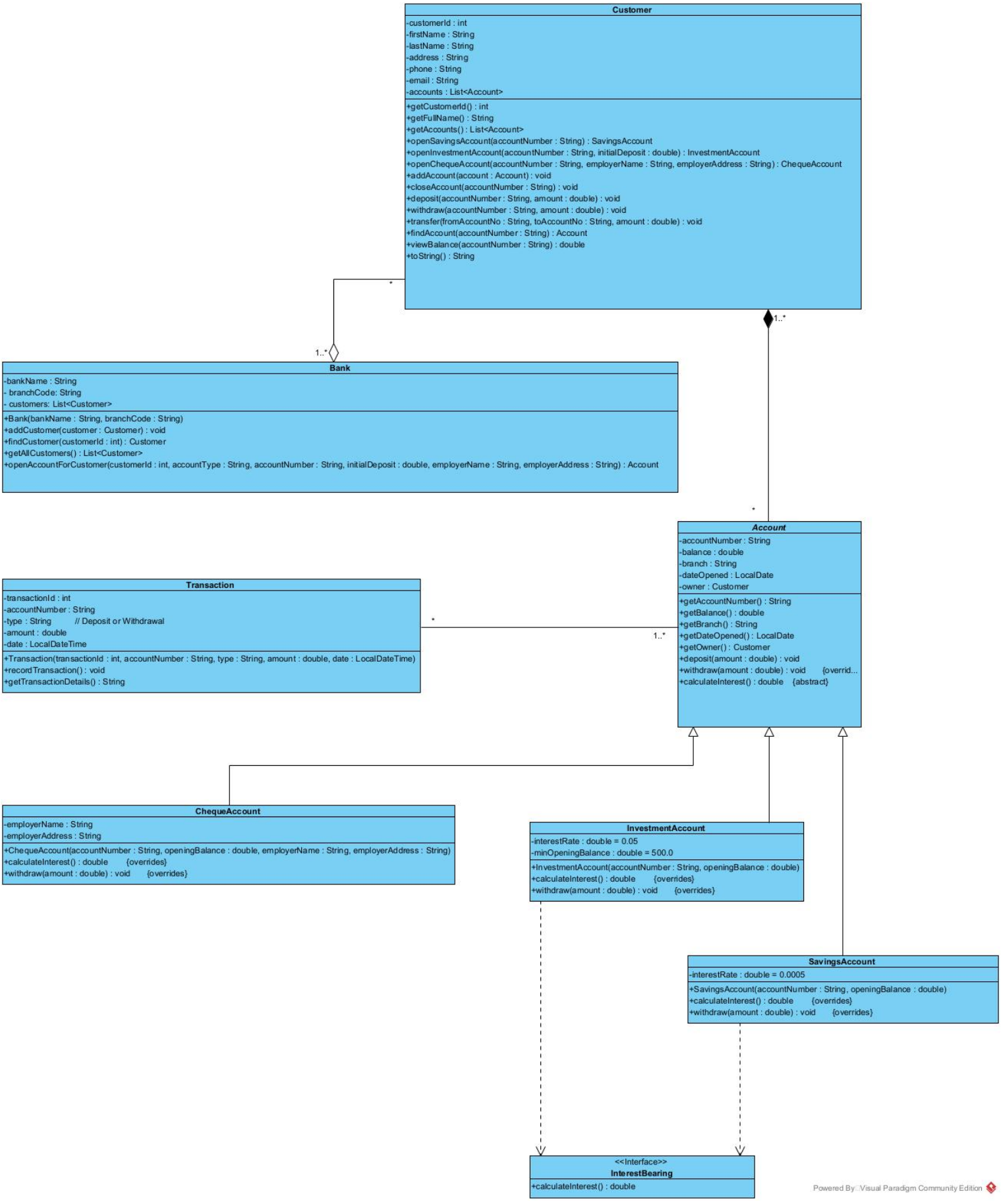
Requirement ID	Use Case / Diagram Reference
FR1 – Register Customer	Use Case: Register / Class: Customer
FR2 – Login User	Use Case: Login / Sequence: Login
FR5 – Open Account	Use Case: Open Account / Sequence: Open Account
FR9 – Deposit Funds	Use Case: Deposit / Sequence: Deposit
FR10 – Withdraw Funds	Use Case: Withdraw / Sequence: Withdraw
FR11 – Transfer Funds	Use Case: Transfer / Sequence: Transfer
FR12 – Calculate Interest	Use Case: Interest / Sequence: Interest
FR13 – Transaction History	Use Case: View History / Sequence: History
NFR1 – Security (hashed passwords)	Interview Q3 / Database class
BR1 – No Savings Withdrawal	Use Case: Withdraw (alt flow)

STRUCTURAL UML MODELLING

System Use Case Diagram



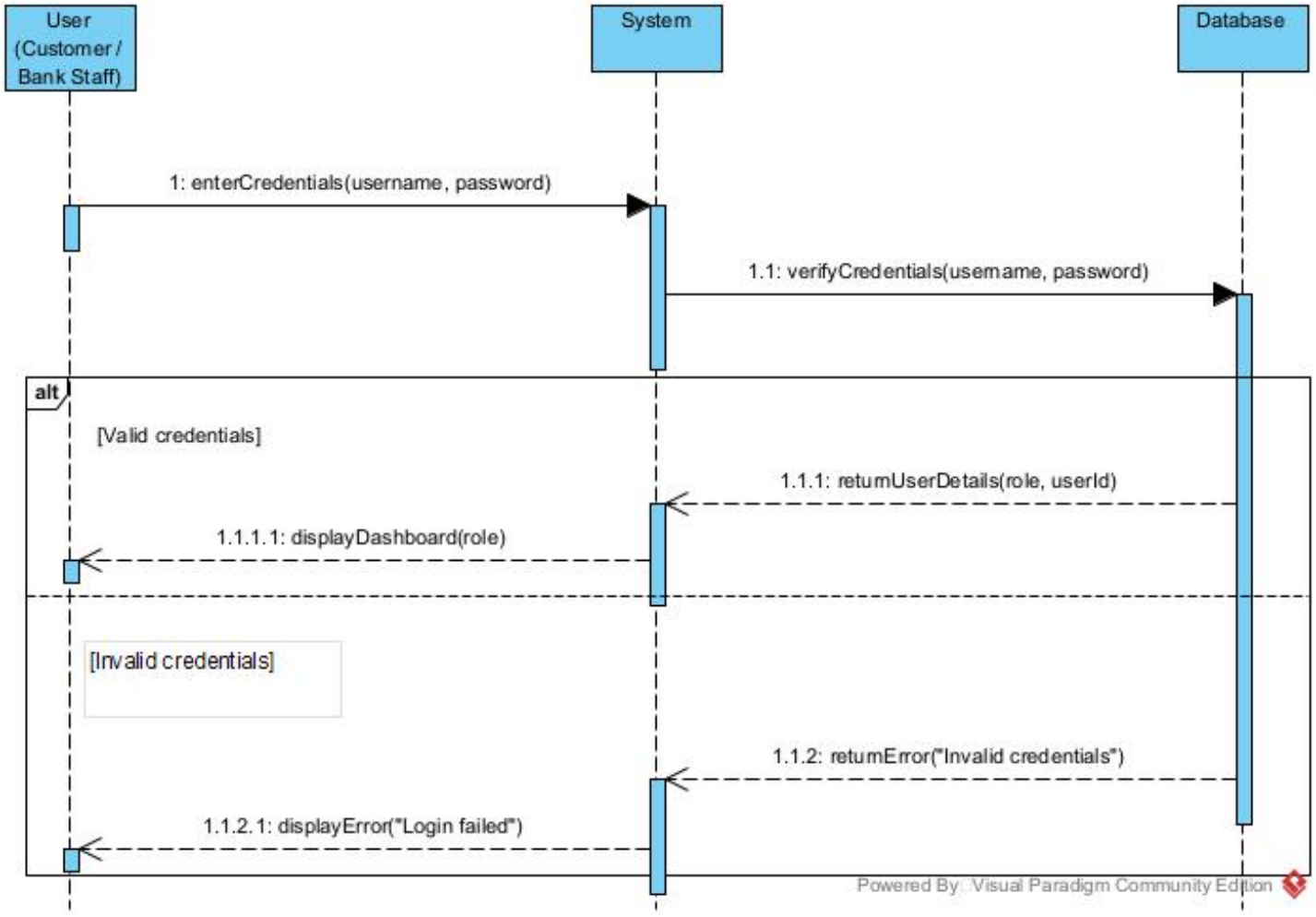
Class diagram



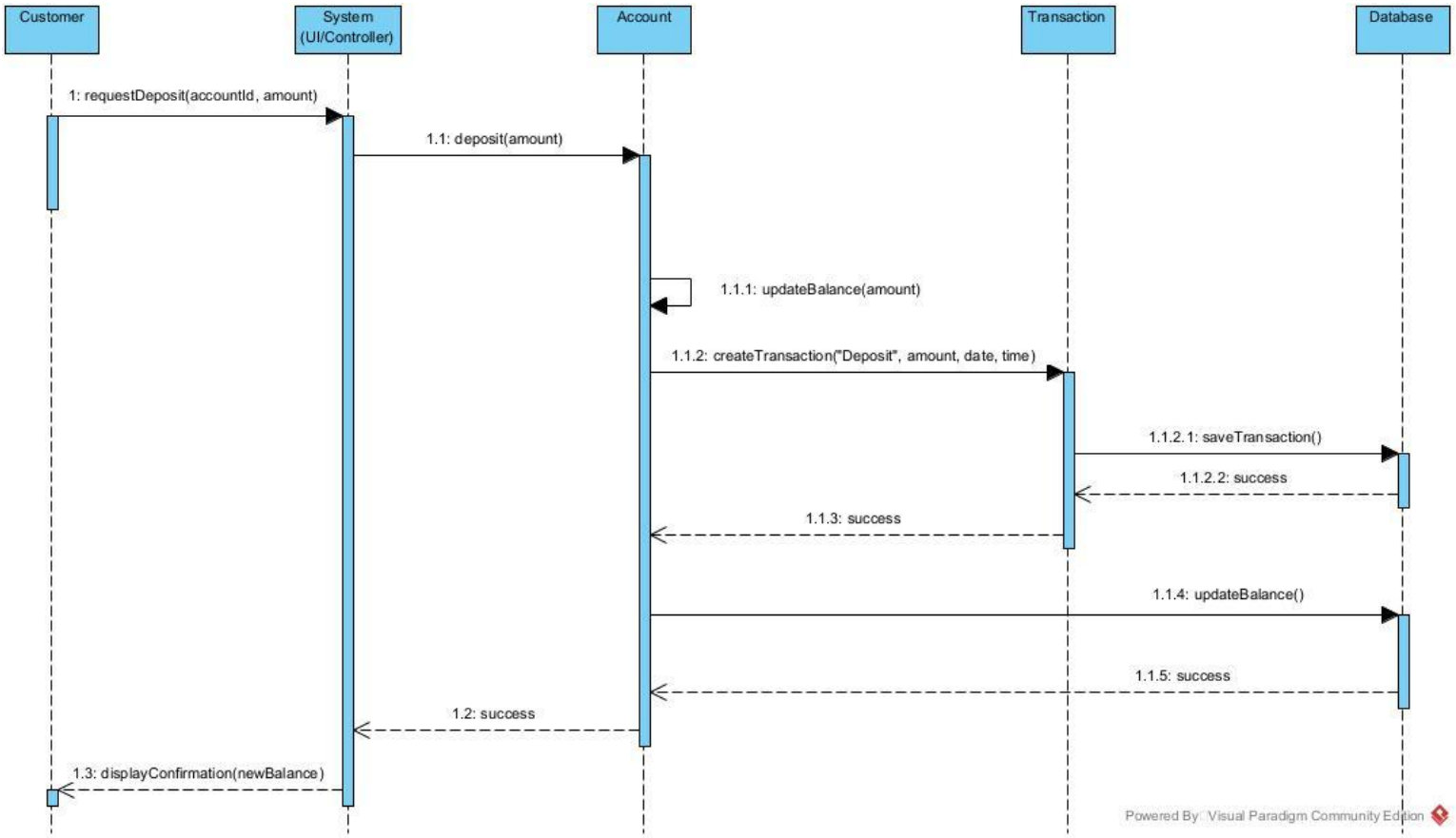
Behavioral UML Modelling

Sequence Diagrams

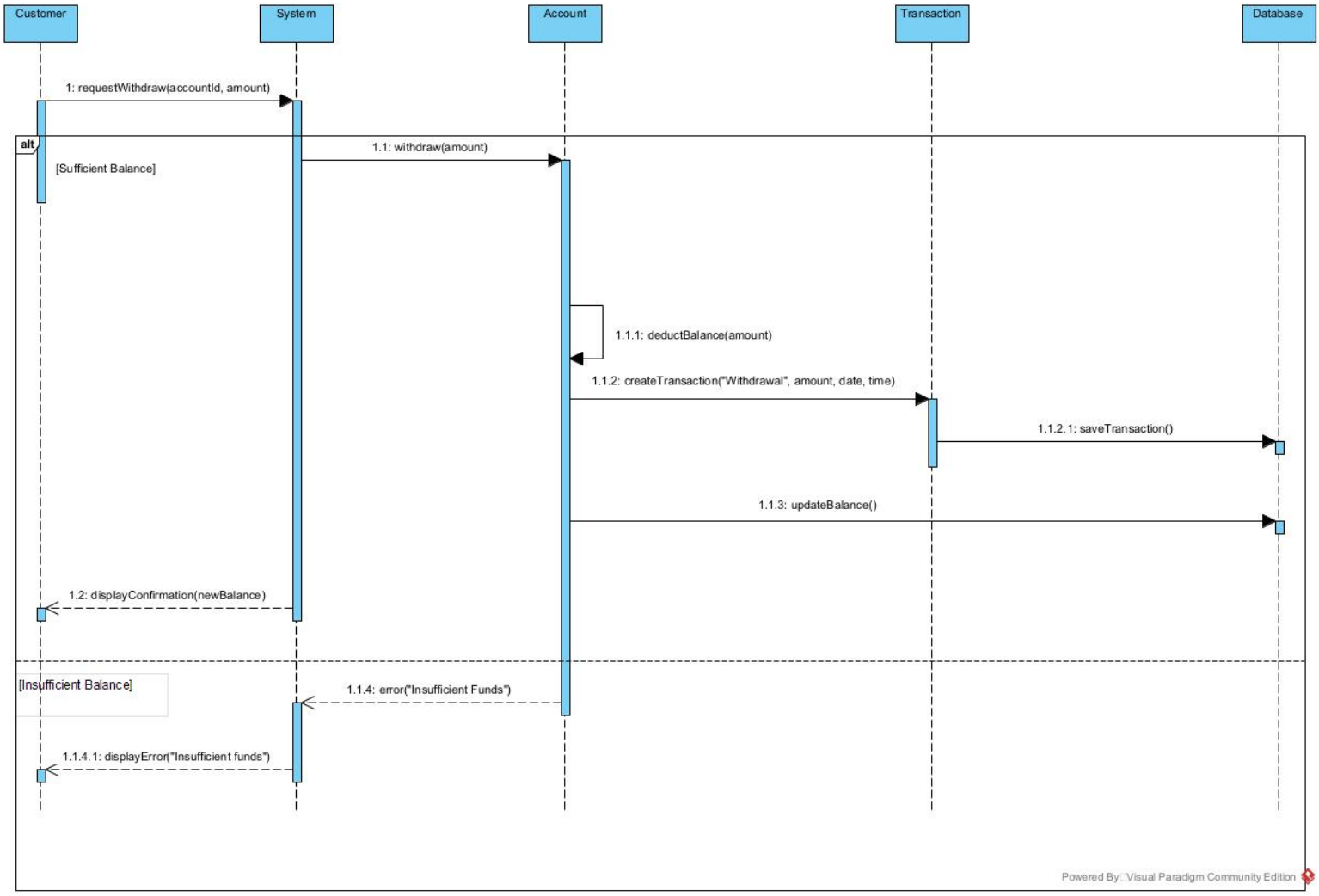
Login



Deposit

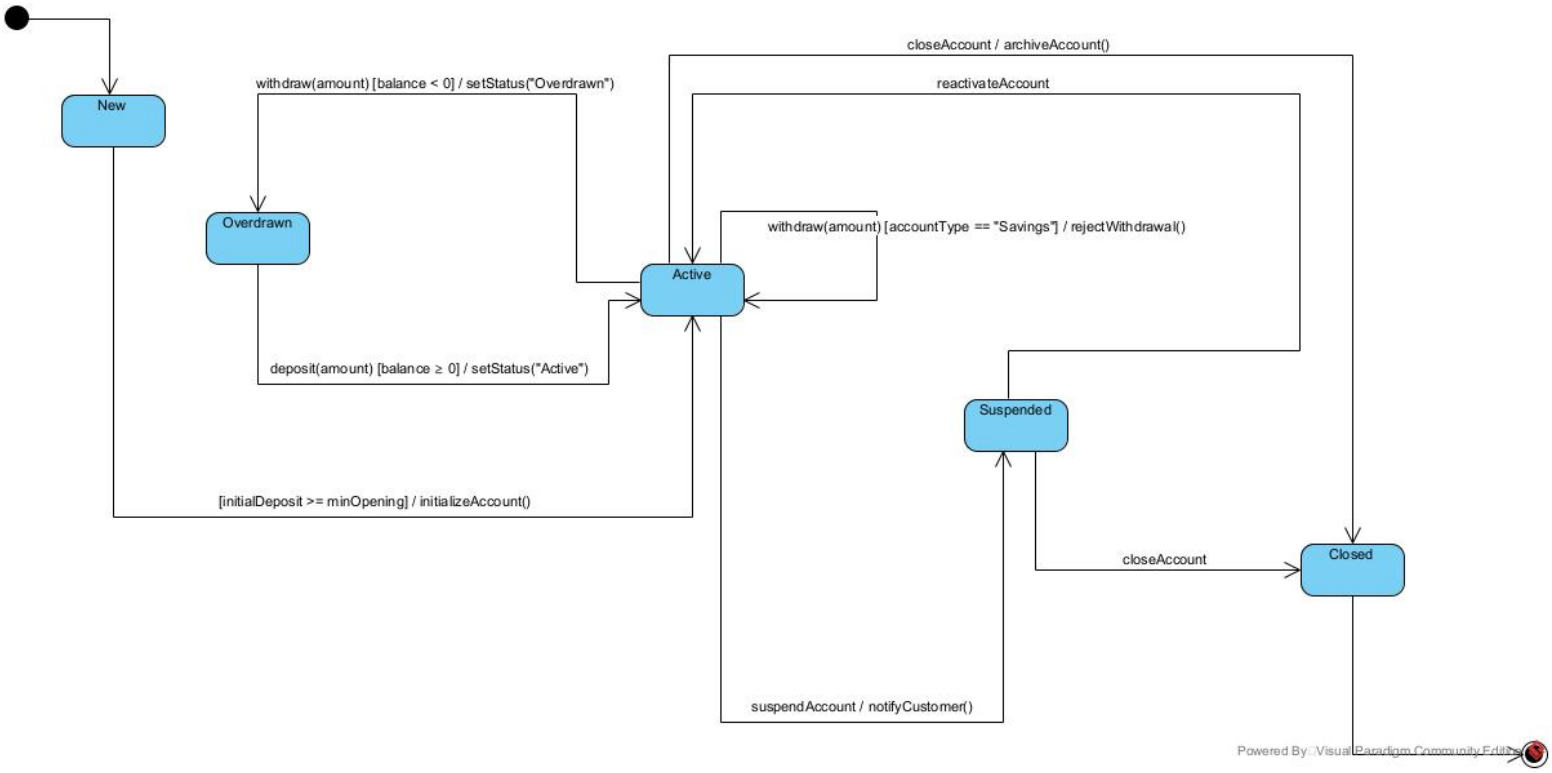

Customer
System (UI/Controller)
Account
Transaction
Database
1: requestDeposit(accountId, amount)
1.1: deposit(amount)
1.1.1: updateBalance(amount)
1.1.2: createTransaction("Deposit", amount, date, time)
1.1.2.1: saveTransaction()
1.1.2.2: success
1.1.3: success
1.1.4: updateBalance()
1.1.5: success
1.2: success
1.3: displayConfirmation(newBalance)
Powered By: Visual Paradigm Community Edition

Withdraw


Customer
System
Account
Transaction
Database
1: requestWithdraw(accountId, amount)
alt
[Sufficient Balance]
1.1: withdraw(amount)
1.1.1: deductBalance(amount)
1.1.2: createTransaction("Withdrawal", amount, date, time)
1.1.2.1: saveTransaction()
1.1.3: updateBalance()
1.2: displayConfirmation(newBalance)
[Insufficient Balance]
1.1.4: error("Insufficient Funds")
1.1.4.1: displayError("Insufficient funds")
Powered By: Visual Paradigm Community Edition

State Diagram

Account State Diagram



Appendix Mock Interview

The following are key questions and answers from the requirements interview session:

Q1. Who is responsible for registering customers — the system or staff?

Answer: "The employees will carry out the process. They verify the customers as they input the data."

Q2. Should the system maintain a transaction history?

Answer: "Yes, every deposit and withdrawal must keep a record of type, amount, account, and date/time."

Q3. How should passwords be stored?

Answer: "They must be encrypted in the database."

Q4. Does the system need a database?

Answer: "Yes. Each entity class maps to a table: customers, accounts, transactions, and credentials."

Q5. Who approves customers? The system or staff?

Answer: "Bank staff handle the verification and approval when registering customers and opening accounts."

Conclusion

The elicited requirements clearly define both the **functional operations** and the **non-functional qualities** of the banking system. Business rules ensure that banking policies are followed, while security and reliability requirements safeguard customer data. With these requirements in place, the design and implementation can proceed confidently, ensuring the final system meets stakeholder expectations.