MODULE CODE: CSE202

MODULE TITLE: Object Oriented Analysis & Design with Java

ASSIGNMENT:Banking System - Part A

STUDENT NAME: Botlhe Kgomphi

STUDENT ID:Cse24-088

SUBMISSION DATE: 19-09-25

1. Introduction

This document presents the complete analysis and design documentation for the Banking System as required for Part A of the CSE202 assignment. The system is designed to facilitate banking operations including customer management, account management, transaction processing, and interest calculation.

The documentation follows Object-Oriented Analysis and Design (OOAD) principles and includes requirements elicitation, structural modeling using UML diagrams, and behavioral modeling to represent system dynamics.

2. Requirements Elicitation

2.1 Functional Requirements

The Banking System shall provide the following functionality:

Customer Management:

- Allow bank employees to create new customer profiles (individual or company)

- Capture all necessary customer information including personal details, contact information, and identification

- Allow customers to login with secure credentials

Account Management:

- Allow bank employees to open different types of accounts (Savings, Investment, Cheque) for customers

- Enforce account-specific rules:

- Savings accounts cannot allow withdrawals

- Investment accounts require minimum initial deposit of BWP 500.00

- Cheque accounts require employment information

- Allow customers to have multiple accounts of different types

Transaction Processing:

- Allow deposits to any account type

- Allow withdrawals from appropriate account types (Investment and Cheque only)

- Record all transactions with timestamps and details

Interest Calculation:

- Automatically calculate and apply monthly interest:

- 5% monthly for Investment accounts

- 0.05% monthly for Savings accounts

Reporting and History:

- Provide transaction history for each account

- Generate customer statements

- Allow viewing of all customers and their accounts

Security:

- Provide secure login for both customers and employees

- Encrypt sensitive data

- Properly logout users and save all data

Non-Functional Requirements

Security:

- All sensitive customer data must be encrypted

- Authentication must be required for all system operations

- Different access levels for customers and bank employees

Performance:

- The system should respond to user inputs within 2 seconds

- Interest calculations should process efficiently even with large customer bases

Usability:

- The interface should be intuitive for both bank employees and customers

- Clear navigation and error messages should be provided

- The system should guide users through complex operations

Reliability:

- The system must maintain data integrity through unexpected shutdowns

- Transaction processing must be atomic (all-or-nothing)

Maintainability:

- The code should follow OOP principles for easy extension

- Clear separation of concerns between different system components

Interview Record

Interview Participants:

- Themba Moeng (Domain Expert/Client)

- Botlhe Kgomphi (Analyst)

Key Points from the Interview:

1. Customer Registration Process:

- Both individual and company customers can be registered

- Different information required for each customer type

- Accounts are linked to either individuals or companies

2. Account Types and Features:

- Multiple account types with different characteristics

- Interest payment functionality for certain account types

- Interest calculation is manually triggered by employees

3. Transaction Processing:

- Deposits and withdrawals with amount validation

- Restrictions on withdrawals based on account type

- Comprehensive transaction recording

4. Reporting Capabilities:

- Ability to view customers over time periods

- Transaction history with date ranges

- Multiple display formats (tabular, graphical)

5. Security Considerations:

- Authentication required for all users

- Password policies enforced

- Database encryption implemented

- Data appears scrambled when accessed directly

6. Interest Calculation:

- Different rates for different account types (5% for investment, 0.05% for savings)

- Applied consistently across both individual and company customers

3. Structural UML Modeling  
  
  
  
  
USE CASE DIAGRAM

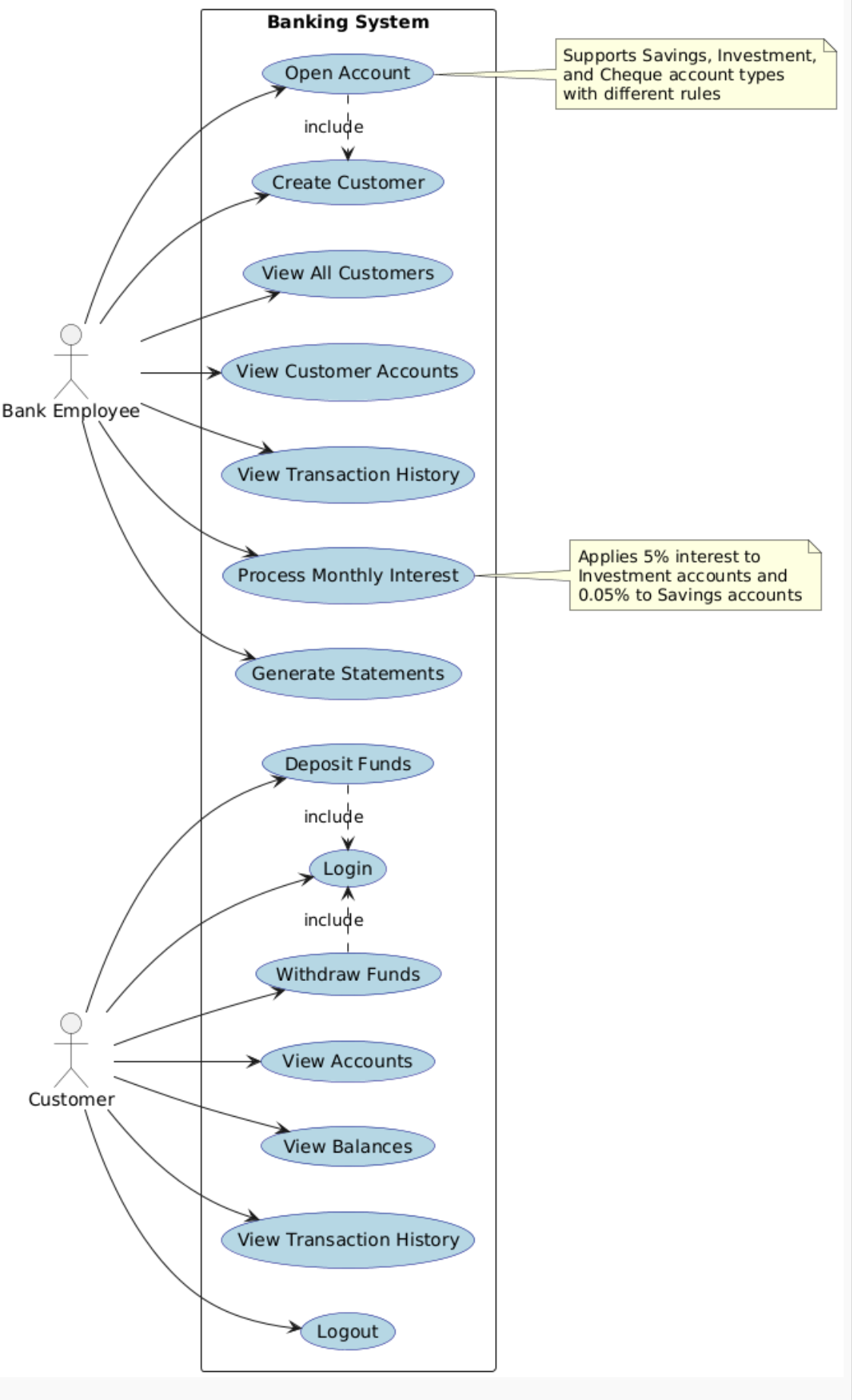


Diagram Description:

The Use Case Diagram illustrates the interactions between the two primary actors (Bank Employee and Customer) and the Banking System. It shows all the functionalities available to each actor and the relationships between different use cases.

Actors:

- Bank Employee: Can manage customers, accounts, and process system operations

- Customer: Can access their accounts, perform transactions, and view history

Use Cases:

- Create Customer: Add new individual or company customers to the system

- Open Account: Create new accounts of different types for customers

- View All Customers: Display list of all customers in the system

- View Customer Accounts: Show all accounts held by a specific customer

- View Transaction History: Display transaction history for an account

- Process Monthly Interest: Calculate and apply interest to appropriate accounts

- Generate Statements: Create formatted statements for customers

- Login: Authenticate users to the system

- View Accounts: Display all accounts belonging to the logged-in customer

- View Balances: Show current balances for accounts

- Deposit Funds: Add money to an account

- Withdraw Funds: Remove money from an account (where allowed)

- Logout: Securely end user session and save data

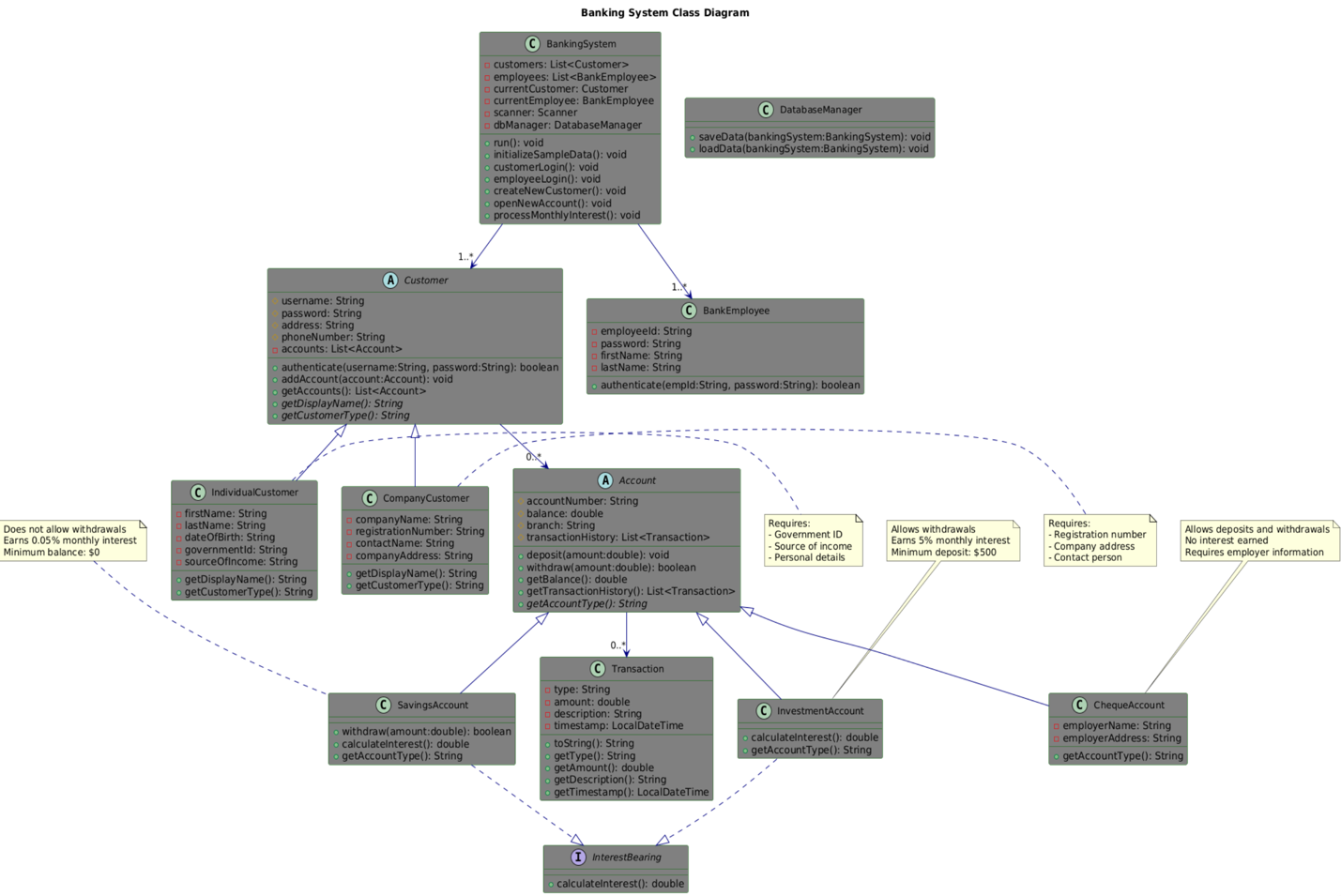
CLASS DIAGRAM  


Diagram Description:

The Class Diagram shows the static structure of the Banking System, including classes, attributes, methods, and relationships between objects.

Key Classes:

- BankingSystem: Main controller class

- Customer: Abstract base class for all customers

- IndividualCustomer: Specialized class for individual customers

- CompanyCustomer: Specialized class for company customers

- Account: Abstract base class for all accounts

- SavingsAccount: Specialized savings account

- InvestmentAccount: Specialized investment account

- ChequeAccount: Specialized cheque account

- BankEmployee: Class representing bank staff

- Transaction: Class representing financial transactions

- DatabaseManager: Class handling data persistence

- InterestBearing: Interface for interest calculation

OOP Principles Demonstrated:

- Abstraction: Customer and Account are abstract classes

- Inheritance: Individual/Company customers inherit from Customer; Account types inherit from Account

- Polymorphism: Different account types implement methods differently

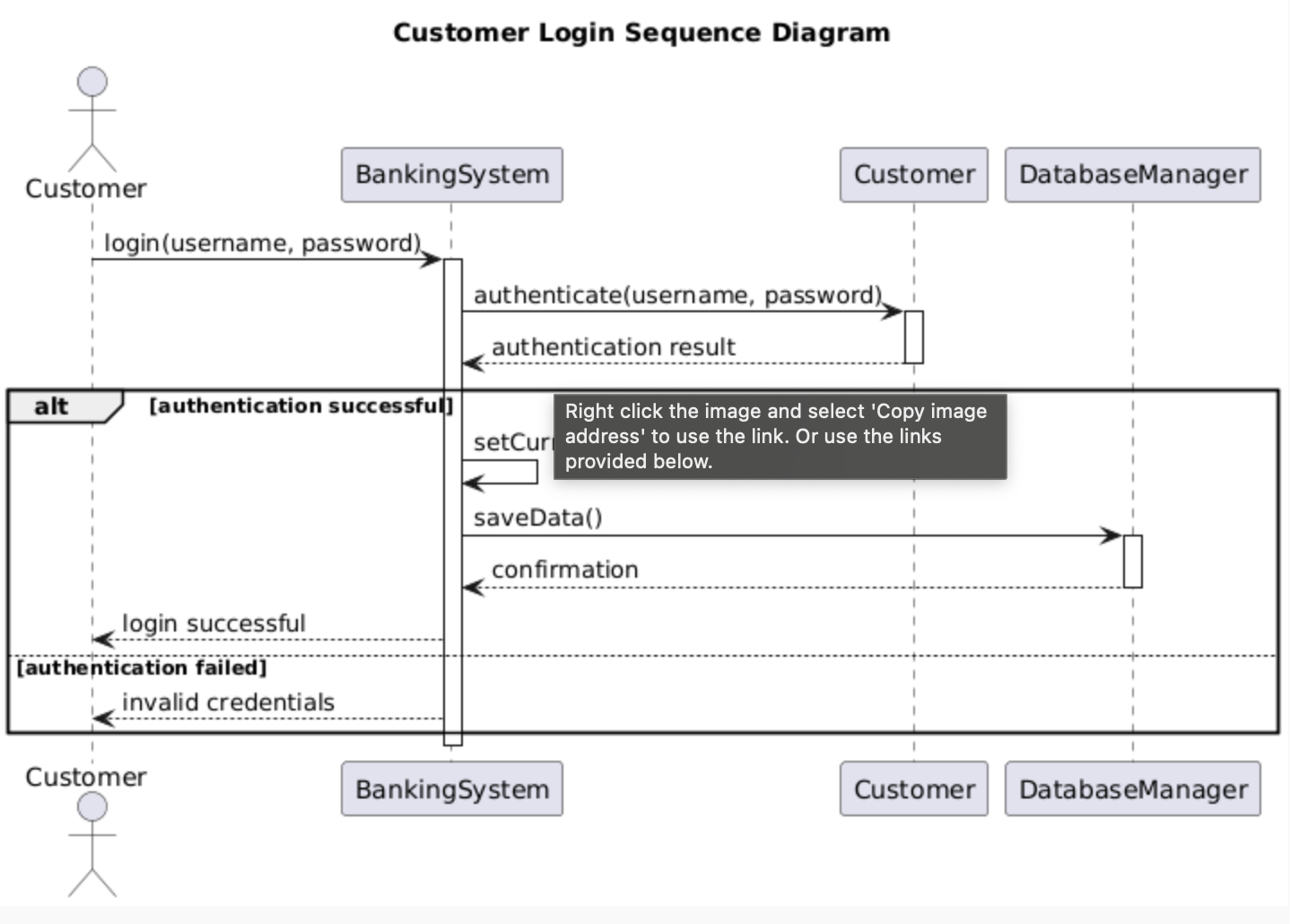
- Encapsulation: Private fields with public access methods

- Interfaces: InterestBearing interface defines contract for interest calculation

Behavioral UML Modeling

Sequence Diagrams

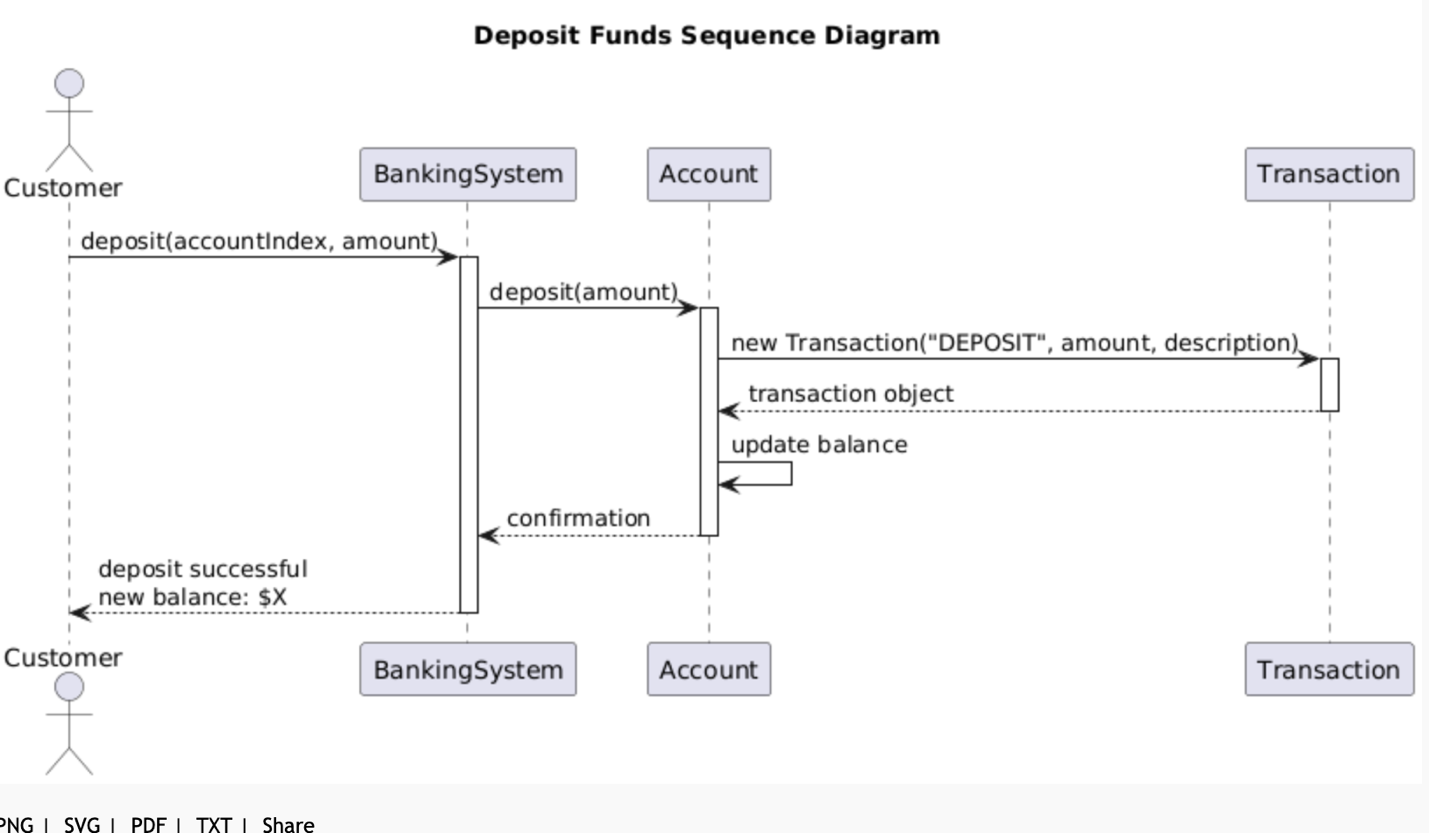
Sequence Diagram 1: Customer Login



Description:

This sequence diagram shows the process of a customer logging into the system, including authentication and session initialization.

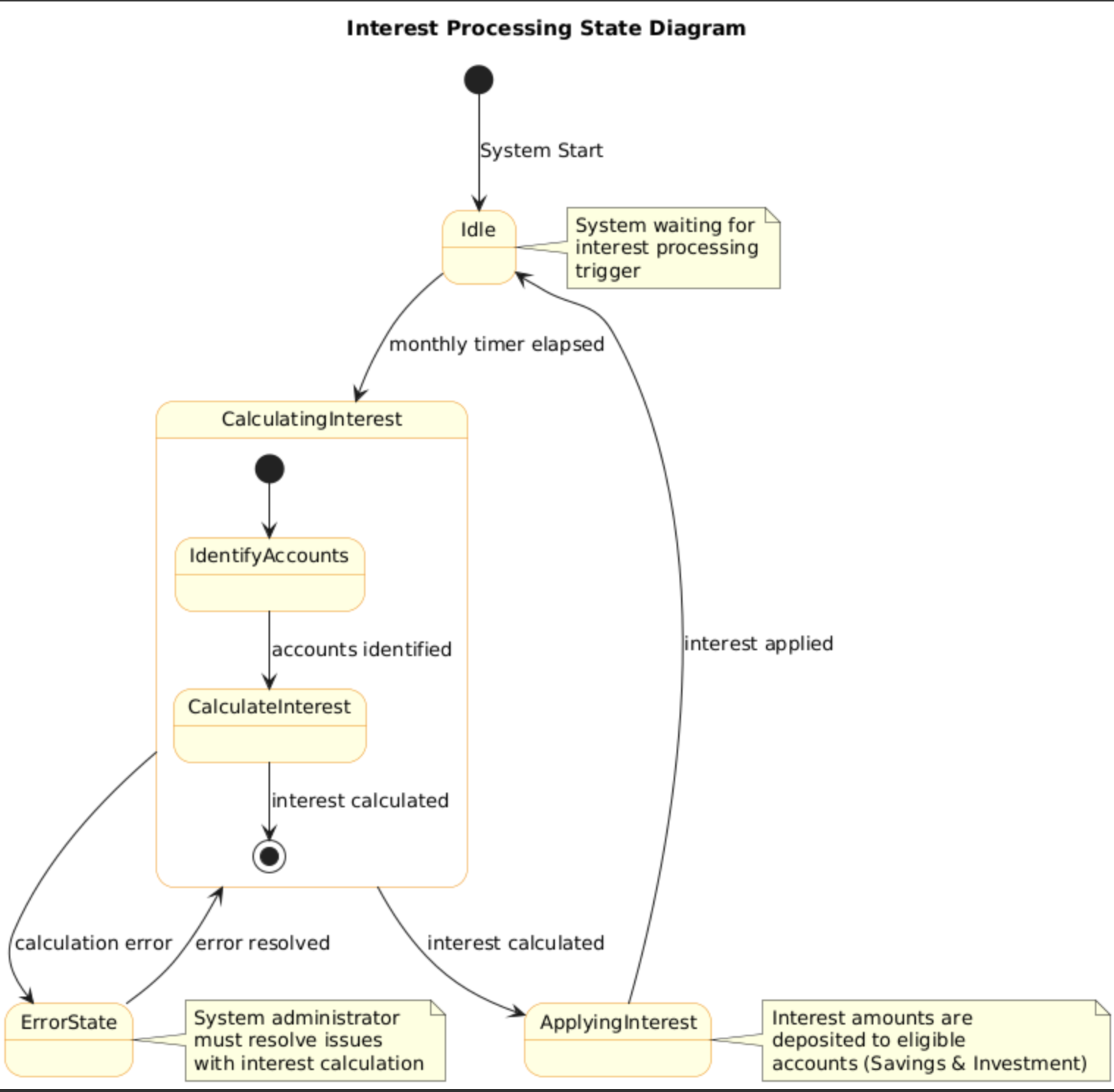
Sequence Diagram 2: Deposit Funds



Description:

This sequence diagram illustrates the process of depositing funds into an account, including validation, transaction recording, and balance updating.

State Diagram



Description:

The State Diagram represents the lifecycle of interest processing for accounts, showing the various states and transitions during the interest calculation and application process.

States:

- Idle: No interest processing occurring

- Calculating Interest: Interest is being calculated for eligible accounts

- Applying Interest: Calculated interest is being applied to account balances

- Error State: An error occurred during processing

Transitions:

- Monthly Interest Timer Elapsed: Triggers interest calculation

- Interest Calculated: Moves to applying interest

- Interest Applied: Returns to idle state

- Error in Calculation: Moves to error state

- Error Resolved: Returns to calculating interest