Appendix 1: Assignment submission cover sheet

**ASSIGNMENT SUBMISSION COVER SHEET**

|  |
| --- |
| cse24-010 |

**Student Id:**

|  |
| --- |
| Nonofo Warren Sekokonyane |

**Student names:**

|  |
| --- |
| cse24-010@thuto.bac.ac.bw |

**Student email**:

|  |
| --- |
| 2025 |

**Cohort**:

|  |
| --- |
| Object Orientied Analysis & Design Assignment |

**Assignment title**:

|  |
| --- |
| 19/09/2025 |

**Date of submission**:

|  |
| --- |
| BSC Computer Systems Engineering |

**.**

**Programme of Study**:

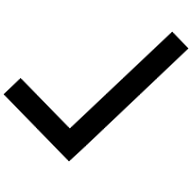
|  |
| --- |
| Year 2 |

**Year of Study**:

**Intellectual property statement**

By checking the box below, I certify that this assignment is my own work and is free from plagiarism. I understand that the assignment may be checked for plagiarism by electronic or other means and may be transferred and stored in a database for the purposes of data-matching to help detect plagiarism. The assignment has not previously been submitted for assessment in any other unit or to any other institution. **I**

**have read and understood the Botswana Accountancy College plagiarism guidelines policy.**

☐ Agree **Signature**…………………………………….

**Date**…09/11/2024…………………………

Part A: System Documentation

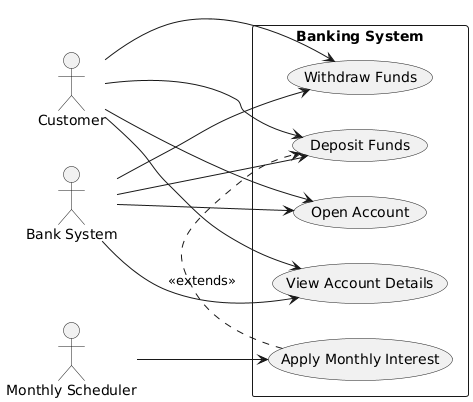
1. Requirements Elicitation
   1. Functional Requirements

* Provide personal information to customers
* Permits clients to create more than one account
* Permit money to be deposited into accounts
* Permit withdrawals (just for checks and investments)
* Stop money from being taken out of savings accounts
* Apply interest each month (invest 5%, save 0.05%)
* Show account balances and details
* Keep track of your transactions
  1. Non-Functional Requirements
* Security: Keep unwanted people out
* Performance: Real time transactions processing
* Usability: Easy-to-use and straightforward interface
* Scalability: Able to accommodate several clients and accounts
* Reliability: Prevent operations and guarantee data consistency

1. Structural UML Modelling

2.1. System Use Case Diagram

* Actors: Customer, Bank System, Monthly Scheduler
* Use Cases: Open Account, Deposit Funds, Withdraw Funds, Apply Interest, View Account Details



2.2 Class Diagram

* Show: Customer, Account (abstract), Savings Account, Investment Account, Cheque Account, Interest Bearing
* Relationships: Inheritance, interface, aggregation

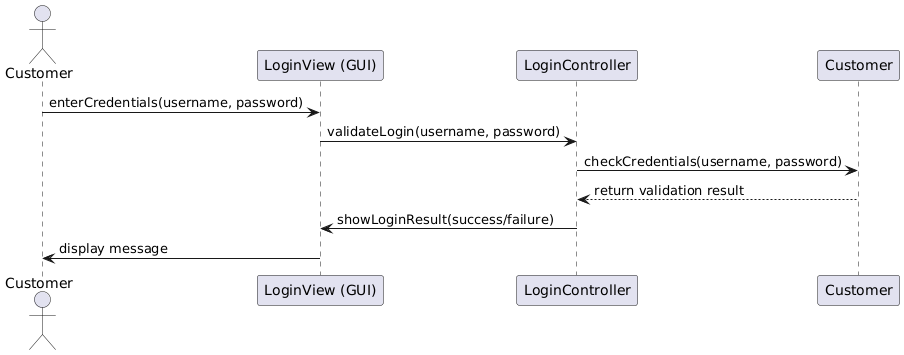
A diagram of a diagram

AI-generated content may be incorrect.

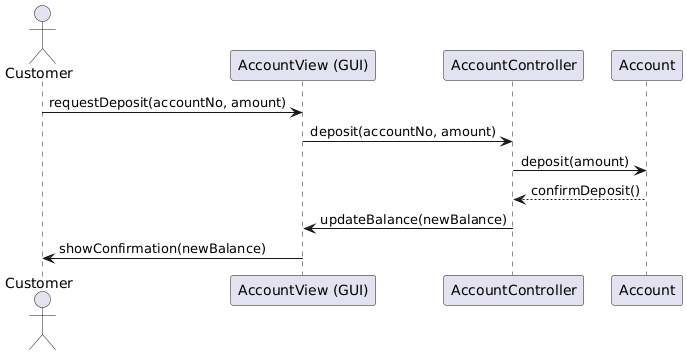
1. Behavioural UML Modelling
   1. Login and Deposit Sequence Diagrams

Login Sequence Diagram

How a customer logs into the banking system is depicted in this diagram. The LoginController receives the credentials that the customer submits through the LoginView (GUI). The Customer object (or subsequently the database) is used by the controller to verify the credentials. The customer sees the outcome whether successful or unsuccessful after it has been returned to the GUI.



Deposit Funds Sequence Diagram

The procedure for making an account deposit is depicted in this diagram. The client uses the AccountView (GUI) to request a deposit. The AccountController receives the request and uses the deposit() function on the relevant Account object. When the balance is changed by the account, the GUI displays the revised balance to the customer after receiving confirmation from the controller.

* 1. State Diagram
* Choose Account lifecycle
* States: Created 🡪Active🡪Interest Applied🡪Closed
* Events: deposit(), withdraw(), applyInterest(), closeAccount()

