Assignment Cover Sheet

Qualification		Module Number and Title
HND in Computing/HND in Software Engineering		**Introduction to OOP- SEC4207**
Student Name & No.		**Assessor**
Hand out date		**Submission Date**
Assessment type WRIT1- Coursework	**Duration/Length of Assessment Type** 3000 words equivalent	**Weighting of Assessment** 100%

Learner declaration
I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.

Marks Awarded			
First assessor			
IV marks			
Agreed grade			
Signature of the assessor		Date	

FEEDBACK FORM
INTERNATIONAL COLLEGE OF BUSINESS & TECHNOLOGY

Module:

Student:

Assessor:

Assignment:

Strong features of your work:

Areas for improvement:

Marks Awarded:

Contents

Table of figures

Task 1

Provide design solution (UML diagrams) for the above mention Scenario. Provide clear explanation for all the diagrams mention below.

System Design

System functionalities are identified during the system design. Also followings are derived from the system design phase.

- Find and define the users of the system.
- Find and define the classes, objects and their attributes and behaviors.
- Describe how the classes interact with one another.
- Define the external behavior of the objects.
- Define the internal behavior of the objects.

Use Case Diagram

Use cases provide a structured view of the system functionality. This view is mainly supported by the Use Case Diagrams, in order to give a better understanding of the system and to define the specific users of the system. This is very important since the next phase of the SDLC is the system designing (Shen & Liu, 2003). Two types of users can be identified who directly interact directly with the system.

- Manager
- Cashier

Assumption: Since customers do not directly interact with the system they are not considered as actors.

Use Case diagrams are consisted of several use cases and the actors of the system. Here the use cases are the different tasks the users will do in order to interact with the system. Actors are the users who interact with the system.

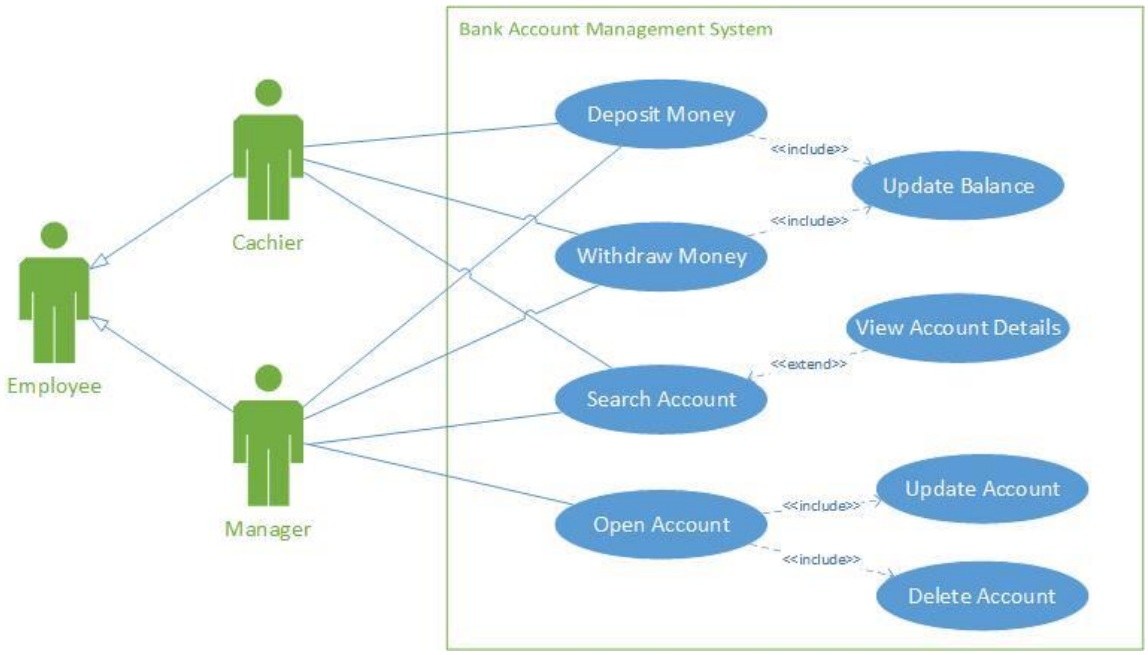


Figure 1: Use case diagram for accounts and transactions

Use Case Description

Open Account

- Description: Manager creates an account.
- Pre-conditions: The cashier cannot create accounts for customers.
- Post-conditions: Both manager and the cashier should be able to update account information. However, only manager is able to delete an account.

Search Account

- Description: Both manager and cashier can search for an account.
- Pre-conditions: Should enter an account number to search.
- Post-conditions: If an account is available the system displays the account information.

Deposit Money

- Description: Both manager and cashier can deposit money.
- Pre-conditions: There should be an account to deposit money.

- Post-conditions: Can deposit any amount of money.

Withdraw Money

- Description: Both manager and cashier can withdraw money.
- Pre-conditions: There should be an account to withdraw money.
- Post-conditions: Can't withdraw than the current balance.
- Only manager can create, update and delete bank accounts. Cashiers can update the ban account information of an account.

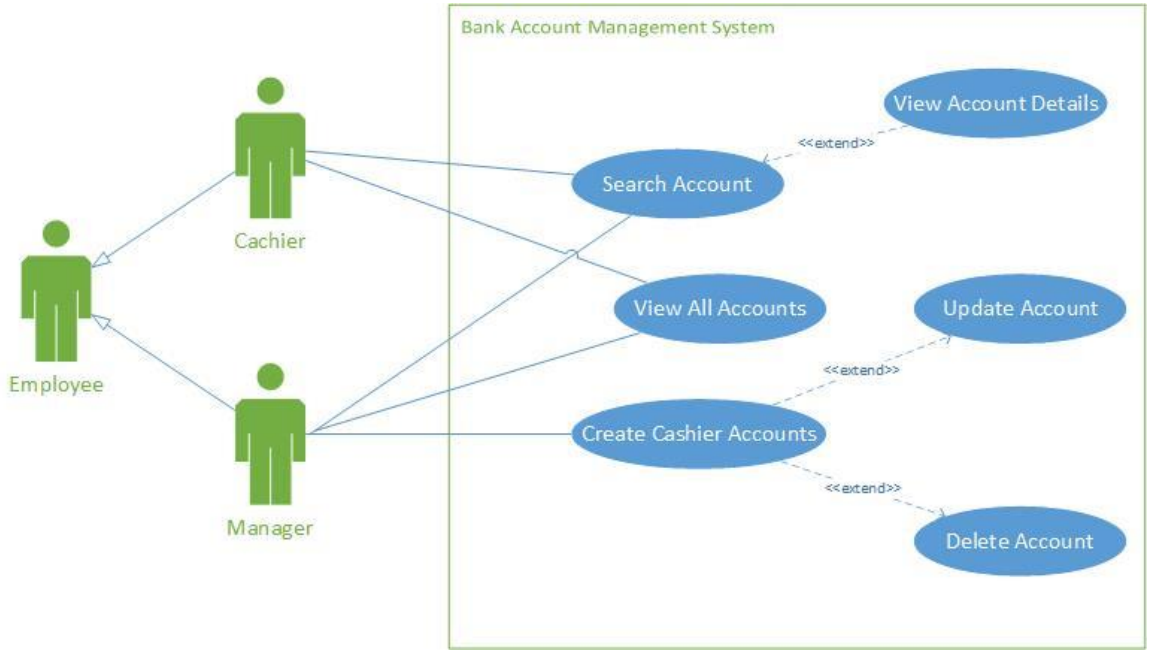


Figure 2: Cashier accounts creation

Use Case Description

Search account

- Description: Both the manager and the cashier can search for an account.
- Pre-conditions: A cashier account should be created first in order to get successful results.

View all accounts

- Description: Both the manager and the cashier can search for all the accounts.
- Pre-conditions: There should be accounts in order to get successful results.

Create cashier accounts

- Description: Manager creates cashier accounts.
- Pre-conditions: Only the manager can create the cashier accounts.
- Post-conditions: Only manager can edit and delete cashier accounts. Editing cashier accounts include updating the username and the password.

Class Diagram

As a crucial early artefact in the development of Object Oriented software, the quality of class diagrams is important for all later design work and could be a main element for the quality of the software product that is finally delivered (Genero, et al., 2002).

The attributes and methods that explains the qualities and behavior of the model classes. And inheritance of the classes is also displayed here. There are basically types of accounts savings account and fixed accounts. Therefore, they are extended from the account class. CommonAccount class is created to implement common methods. AccountDAO class is created to handle creating, updating, reading and deleting accounts. AccountDAO is associated with FixedAccount, SavingsAccount as well as CommonAccount classes.

TransactionDAO class is created to handle creating, updating, reading and deleting transactions. TransactionDAO is associated with Transaction class as well as CommonAccount classes.

UserDAO class is created to handle creating, updating, reading and deleting cashier accounts. UserDAO is associated with User class.

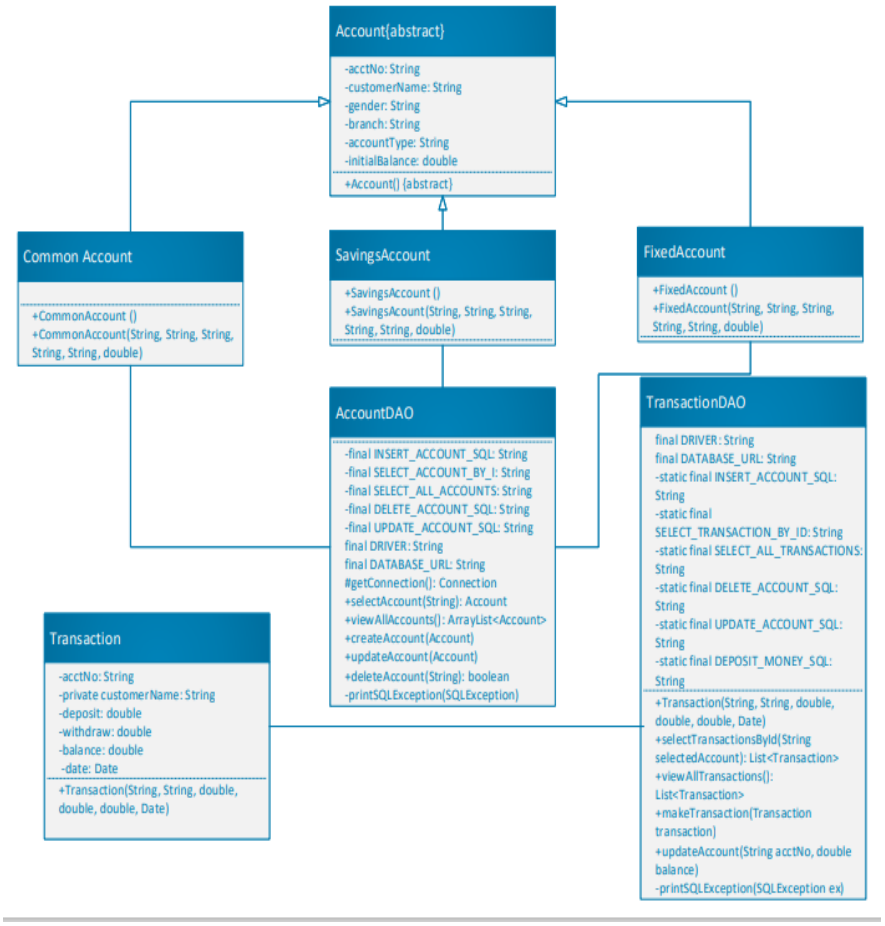


Figure 3: Class diagram: bank accounts and transaction

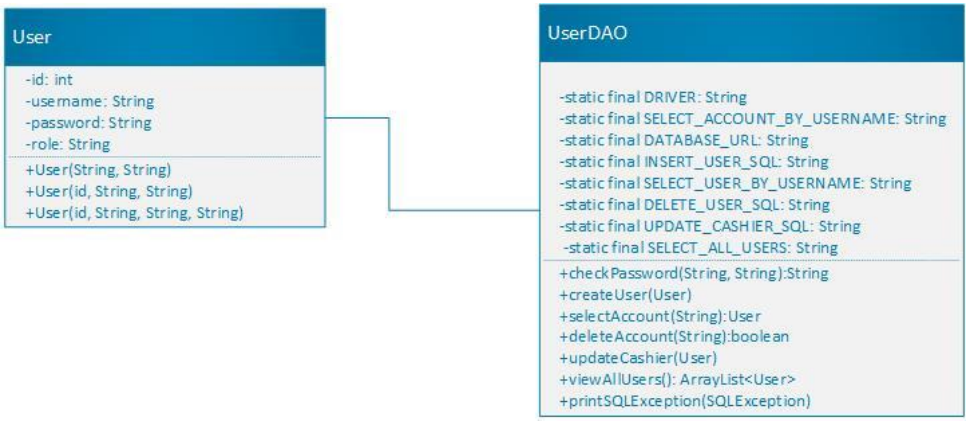


Figure 4: Class diagram: account management

The class diagrams are drawn separately for clearance.

Sequence Diagram

Sequence Diagrams show how the System interacts with the actors in a use case functionality. Each actor is represented with a horizontal life line and the data transactions are drawn from one life line to another or within one life line. Following Sequence diagrams describe some of the main use cases which are a bit difficult to understand with only having Use case descriptions (Li, et al., n.d.).

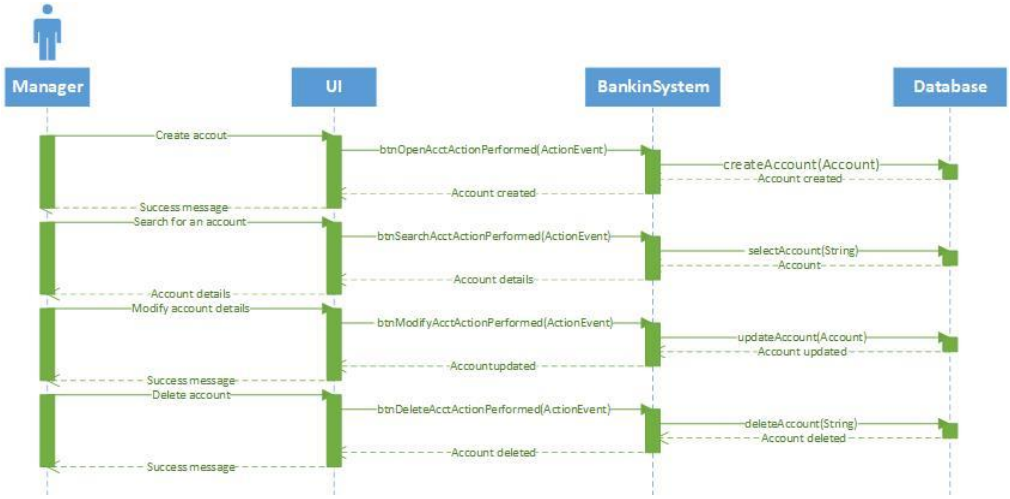


Figure 5: Sequence diagram bank account creation, modification, deletion and search

This sequence diagram depicts what functions are invoked when creating, modifying, deleting and searching for a specific bank account.

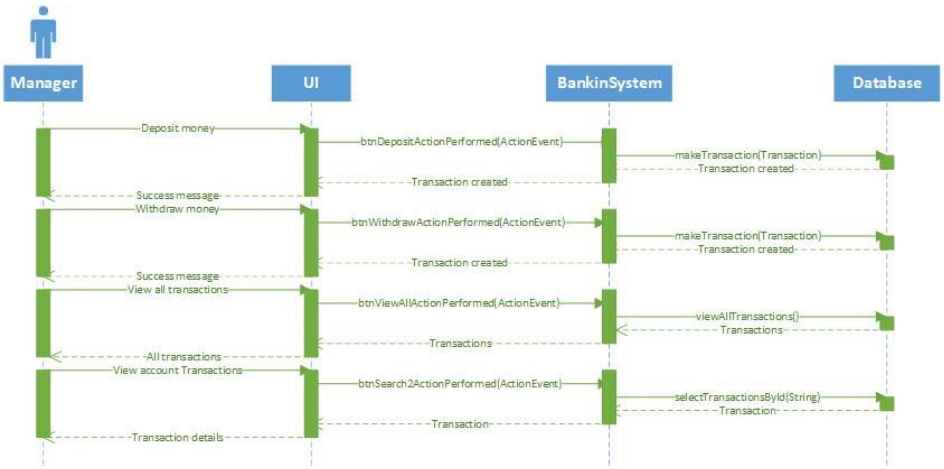


Figure 6: Sequence diagram account transactions

This sequence diagram depicts what functions are invoked when selecting transactions, making deposits and making withdrawals.

Through the sequence diagrams, it is described how method calls are done and how the objects are created can be understood using these sequence diagrams. Since the graphical user interfaces are the front end that the users interact with, the sequence diagrams are initiated mostly with an object of a user interface.

Assumptions

- There can’t be multiple bank accounts with the same account number.
- Cashiers can’t create or delete bank accounts they can only modify bank account details.
- Only the manager can create/ update/ delete cashier accounts.
- There can’t be multiple cashier accounts with the same username.
- Since end customers are not directly associating with the system they are not considered in the diagrams.

Task 2

Develop suitable system for the above scenario based on the design. Required to use Object Oriented concepts (Object, Class, Abstraction, Inheritance, Encapsulation and Polymorphism) for the development. Document the main functionalities and Object Oriented concepts applied with proper explanation and source code.

System Design and Analysis

System design is done by using either Structured Systems Analysis and Design (SSADM) or Object Oriented System Analysis and Design (OOSAD). I selected OOSAD method because it is very much understandable for everyone it will and it is based on object oriented concepts (Mathiassen & Madsen, 2000). Since I am using object oriented concepts it will much helpful for implementation stage too.

Use of Object Oriented Concepts

Classes and Objects

In Object Oriented Programming everything is associated with classes and objects, along with its attributes and methods (W3Schools, 2021). An account has attributes, such as account number, customer name, branch etc. and methods, such createAccount, modifyAccount, deleteAccount etc.

A class can be defined as a blueprint of objects. In this system we have defined 9 classes.

- Account
- FixedAccount
- SavingsAccount
- CommonAccount
- AccountDAO
- Transaction
- TransactionDAO
- User
- UserDAO

For an example, every time when the manager creates an account, an object is created form SavingsAccount of FixedAccount class.

Inheritance

There are two types of accounts and they are Savings Accounts and Fixed Accounts. Both these classes child classes of the Account class. Therefore, all the parent class features are inherited to these classes.

```
public class FixedAccount extends Account{

public class SavingsAccount extends Account{
```

Figure 7: Inheritance

Abstraction

Abstraction can be defined as hiding certain details and showing only essential information to the user. It can be achieved through abstract classes of interfaces. In this system we have defined Account class as an abstract class.

```
public abstract class Account {
```

Figure 8: Abstract class

Since abstract classes are restricted from creating objects. Therefore, to access it, it is inherited from sub class (FixedAccount, SavingsAccount).

Encapsulation

Encapsulation, make sure that the sensitive data is hidden from outside. To achieve this, we have

- declared class variables as private
- provide public get and set methods to access and update the value of a private variable

```java
public abstract class Account {
    private String acctNo;
    private String customerName;
    private String sex;
    private String branch;
    private String accountType;
    private double initialBalance;

    public void setAcctNo(String acctNo) {
        this.acctNo = acctNo;
    }

    public String getAcctNo() {
        return acctNo;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public String getCustomerName() {
        return customerName;
    }
```

Figure 9: Private variables/ setters and getters

Polymorphism

Polymorphism is known as having multiple forms, and it occurs when we have many classes that are related to each other by inheritance.

For an example objects from SavingsAccount and FixedAccount classes are created and then createAccount(newAccount) method is called.

```java
if (accountType == "Savings") {
    Account newAccount = new SavingsAccount(acctNo, customerName, sex, branch, accountType, initial_balance1);
    account.createAccount(newAccount);
}else if (accountType == "Fixed") {
    Account newAccount = new FixedAccount(acctNo, customerName, sex, branch, accountType, initial_balance1);
    account.createAccount(newAccount);
}
```

Constructor overloading is another example for polymorphism in the system.

```java
public User(String username, String password, String role) {
    this.username = username;
    this.password = password;
    this.role = role;
}

public User(String username, String password) {
    this.username = username;
    this.password = password;
}

public User(int id, String username, String password) {
    this.id = id;
    this.username = username;
    this.password = password;
}
public User(int id, String username, String password, String rol
    this.id = id;
    this.username = username;
    this.password = password;
    this.role = role;
}
```

Figure 10: Constructor overloading in User class

```java
public Account(String acctNo, String customerName, String sex, String branch, double initialBalance) {
    super();
    this.acctNo = acctNo;
    this.customerName = customerName;
    this.sex = sex;
    this.branch = branch;
    this.initialBalance = initialBalance;
}

public Account(String acctNo, String customerName, String sex, String branch, String accountType, double initialBalance) {
    super();
    this.acctNo = acctNo;
    this.customerName = customerName;
    this.sex = sex;
    this.branch = branch;
    this.initialBalance = initialBalance;
    this.accountType = accountType;
}
```

Figure 11: Constructor overloading in Account class

This is an example of polymorphism because the constructor overload to be executed is chosen at compile time. This is quite similar to the regular methods, where the overload of a method to be invoked is chosen at compile-time also, although different parts of the language specification describe the behavior.

Database Connectivity

MySQL data base is used for the system. We have used MySQL JDBC driver (Java Database Connectivity) to connect to the database. JDBC is a Java API to connect and execute the query with the database (Oracle, 2021).

We have defined 3 tables for the system.

tblAccount(acct_no, customer name, sex, branch, account_type, initial_balance)

	#	Name	Type
☐	1	acct_no	varchar(20)
☐	2	customer_name	varchar(30)
☐	3	sex	varchar(10)
☐	4	branch	varchar(50)
☐	5	account_type	varchar(25)
☐	6	initial_balance	double

Figure 12: tblAccount table

transaction(acct_no, custormer_name, deposit, withdraw, balance, date)

	#	Name	Type
☐	1	acct_no	varchar(20)
☐	2	customer_name	varchar(30)
☐	3	deposit	double
☐	4	withdraw	double
☐	5	balance	double
☐	6	date	timestamp

Figure 13: Transaction table

user(id, username, password, role)

	#	Name	Type
☐	1	id	int(11)
☐	2	username	varchar(25)
☐	3	password	varchar(25)
☐	4	role	varchar(20)

Figure 14: User table

Special considerations and assumptions

- A custormer_name column is available in both the tblAccount and transaction tables. In a practical scenario a different person can deposit money for a specific account. It doesn't always have to be the account holder. Therefore, even though the customer_name column is defined in two tables it won't be a redundant column.
- Both the deposits and withdrawals are stored in the same table. When a withdrawal is processed the deposit value would be 0 and when a deposit is processed the withdrawal amount would be 0.

Main functionalities

There are two direct user types of the system. Therefore, all the functionalities are based on these two user types. This section gives a scope description and overview of everything included in the functionalities. The purpose for this is described and a listed below. It gives a detailed description of the functionalities of the bank account management system. It will illustrate the purpose and complete declaration for the development of system.

Cashier

1. Cashier shall be able to view account holder details.
2. Cashier shall be able to manage transactions.
    a. Withdraw money
    b. Deposit money
3. Check bank account balance.
4. Search bank account details.
5. View all available cashier accounts.

Manager

1. Cashier shall be able to view account holder details.

2. Cashier shall be able to manage transactions.
    a. Withdraw money
    b. Deposit money
3. Check bank account balance.
4. Search bank account details.
5. Manage cashier accounts.
    a. Create cashier accounts.
    b. Update cashier accounts.
    c. Delete cashier accounts.
    d. View all the cashier accounts.

Task 3

Provide a user manual for the developed solution.

Login Screen

This is the login window of bank accounts management system.

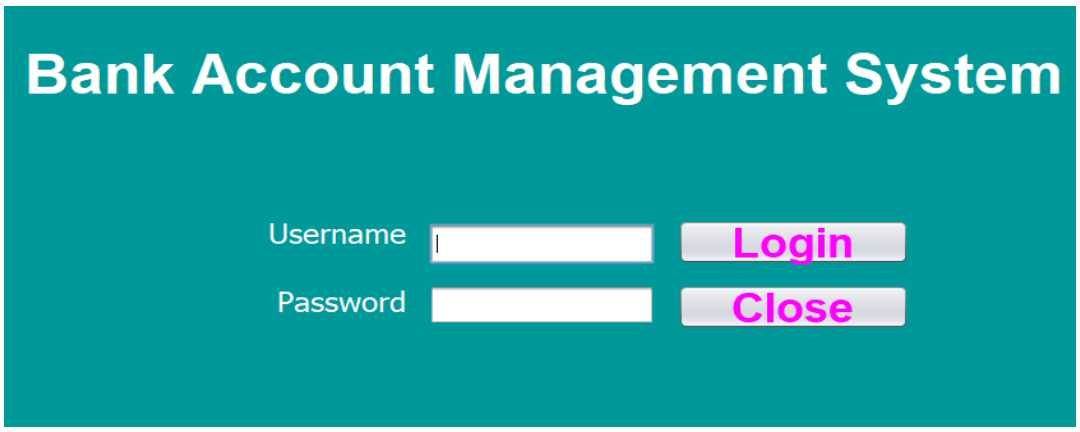


Figure 15: Login screen

If the user decides to continue as the manager user can click on the "Manager button" or else the user can continue as a cashier by clicking on the "Cashier" button. Both the type of users have different level of access to some functionalities of the system.

Manager

This is the home screen of the banking system. When the user decides to continue as the manager the user is directed to the following screen. If you login as a cashier you won't be able to see open and delete buttons.

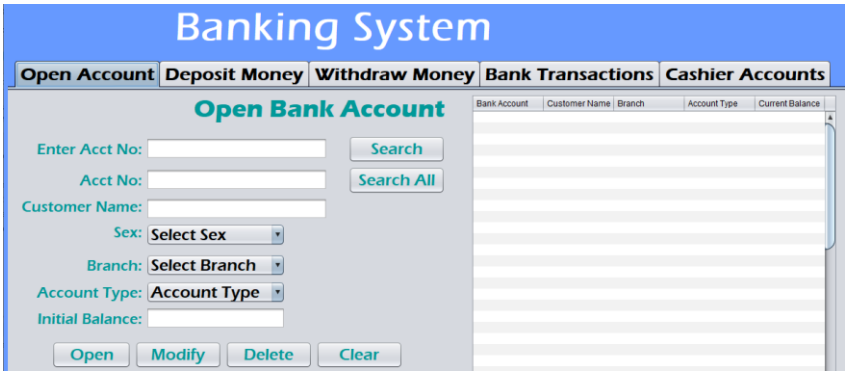


Figure 16: Manager home screen

Create new bank account.

After inserting all the required information, the manager can click “Open” button to open a new bank account.

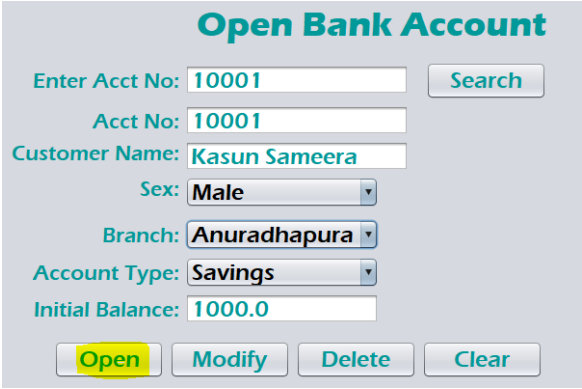


Figure 17: Create new account

All the required information should be provided to create an account.

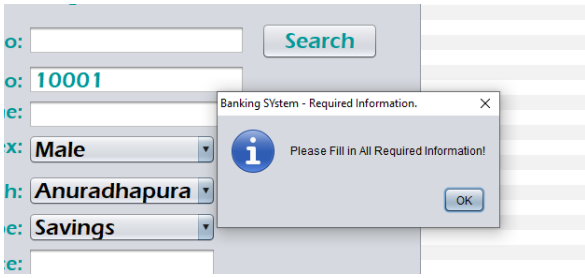


Figure 18: Information required warning message

Search for an account

Search button can be used to search for an account.

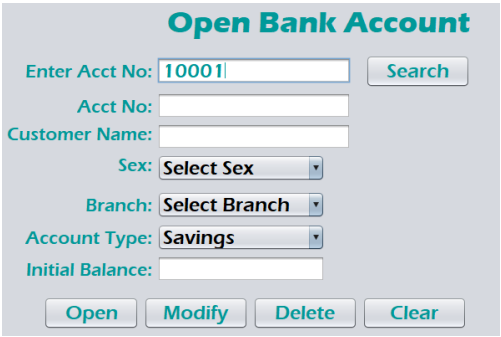


Figure 19: Search account

Update and delete account

After searching for an account, the user can modify the account information.

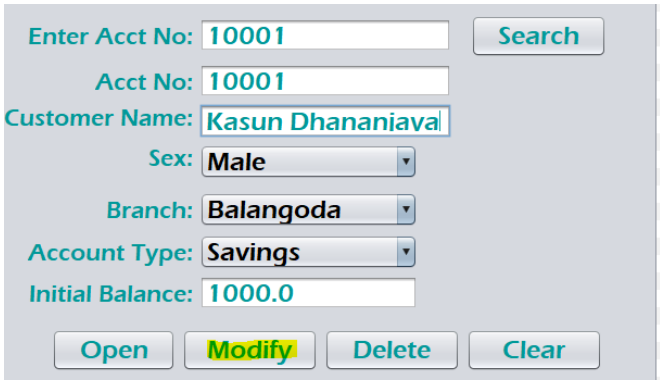


Figure 20: Modify account information

After searching for an account the manager can delete the selected account.

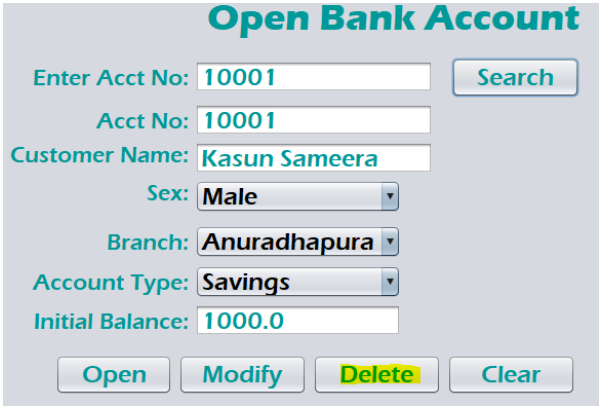
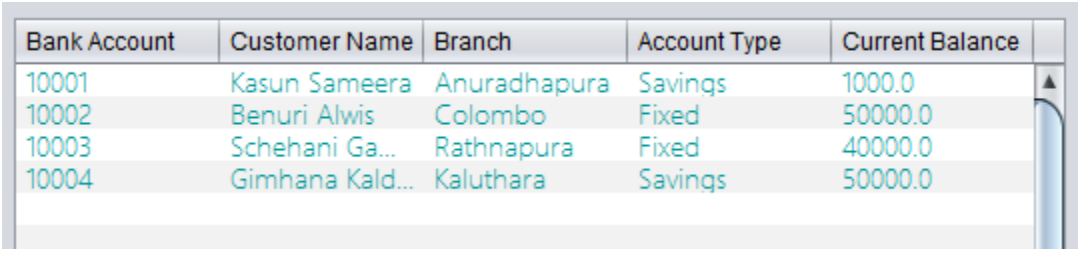


Figure 21: Delete an account

View all accounts

All the accounts created can be seen in the right side of the application window.



Bank Account	Customer Name	Branch	Account Type	Current Balance
10001	Kasun Sameera	Anuradhapura	Savings	1000.0
10002	Benuri Alwis	Colombo	Fixed	50000.0
10003	Schehani Ga...	Rathnapura	Fixed	40000.0
10004	Gimhana Kald...	Kaluthara	Savings	50000.0

Figure 22: All bank accounts

Deposit Money

Click deposit money from the navigation pane to navigate to deposit money window. An account must be selected before depositing money therefore, search for an account and add the depositing amount.

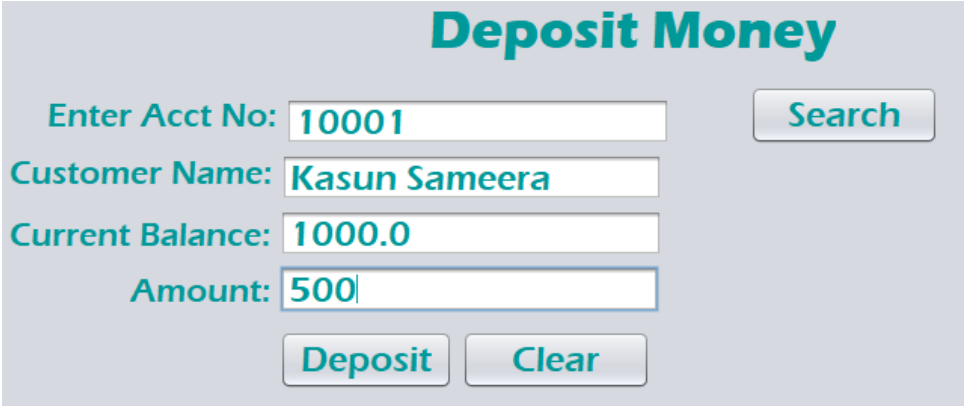


Figure 23: Deposit money

Withdraw Money

Click withdraw money from the navigation pane to navigate to withdraw money window. An account must be selected before withdrawing money therefore, search for an account and add the withdraw amount.

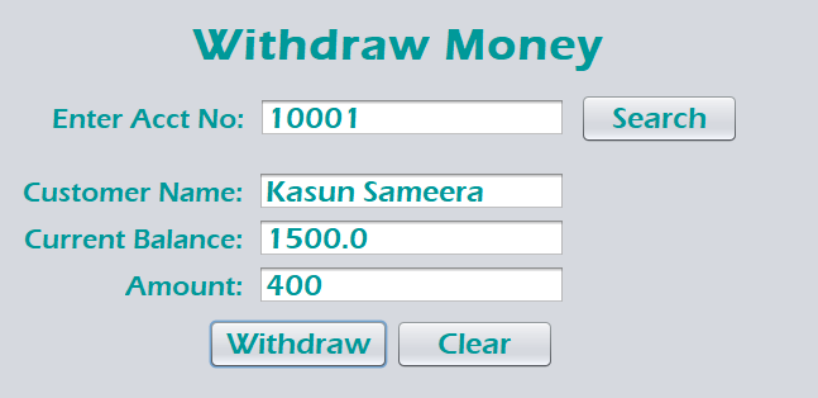


Figure 24: Withdraw money

If the user tris to withdraw an amount more than the current account balance the system will pop up an error massage.

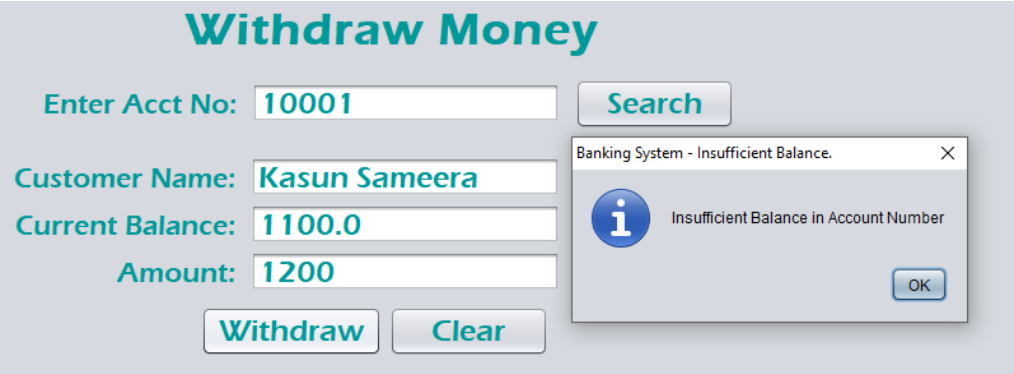


Figure 25: Insufficient balance message

View transaction details

Navigate to "Bank Transaction" section to view details about the transactions done. Click on "View All" button to view all the transactions done within the day.

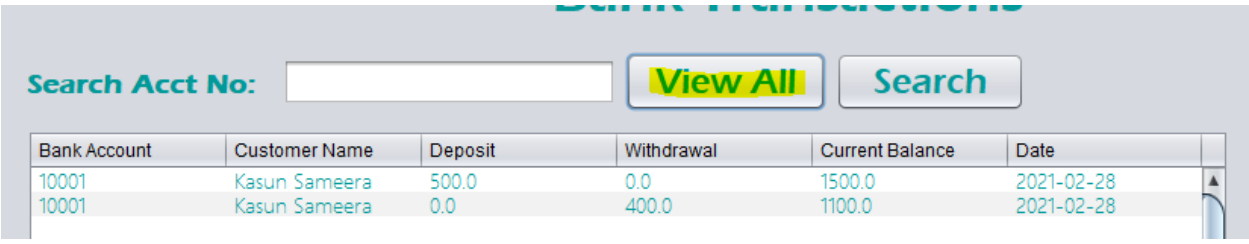


Figure 26: View all transactions

Enter an account number on the search bar and click "Search" button to get all transactions done from a selected account.

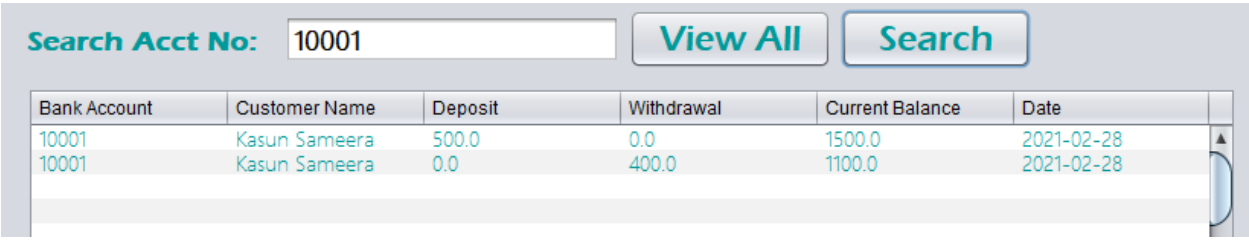


Figure 27: View transactions done by an account

Cashier

Cashier has all the permissions same as the manager except creating and deleting bank accounts. Therefore, when a user continues as a cashier, create account and delete account buttons are not visible.

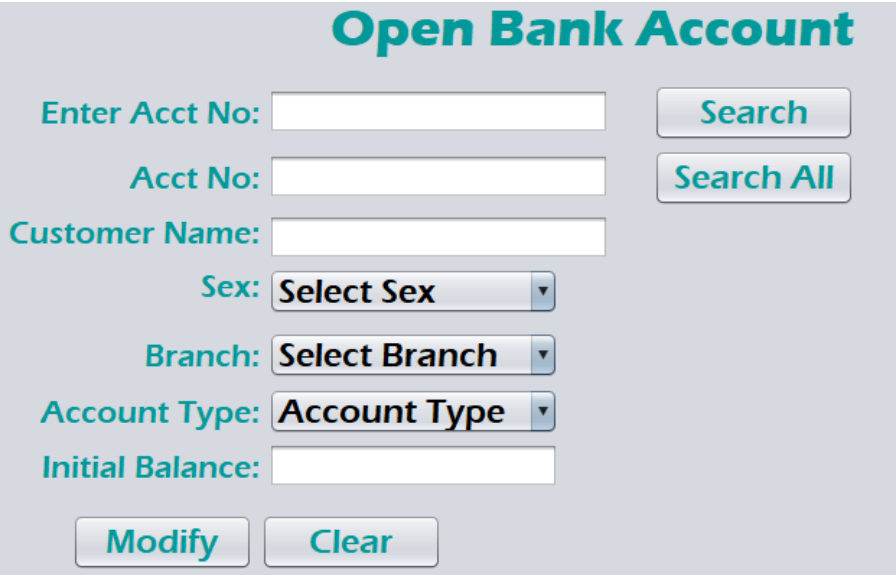


Figure 28: Cashier view

Cashier accounts

Manager can create, update, delete, search and view all cashier accounts casher accounts.

Figure 29: Cashier accounts

Both the manager and the cashiers can search for accounts.

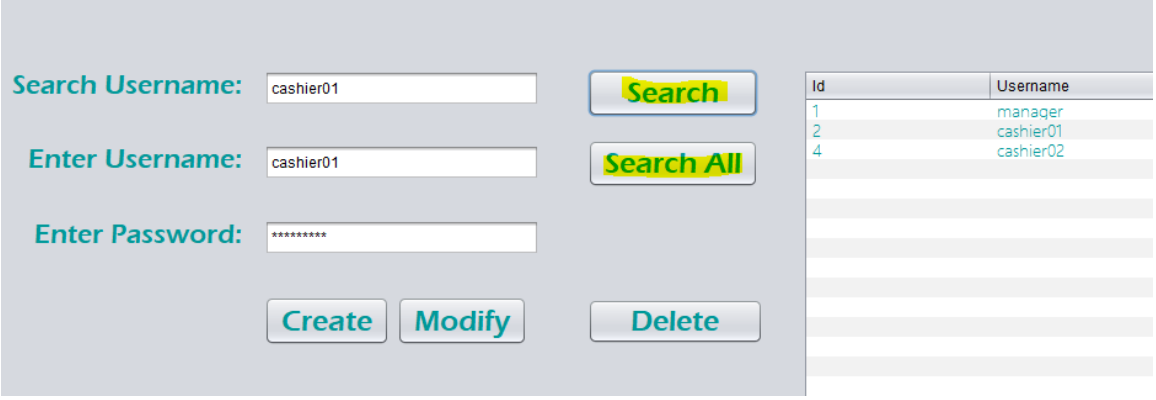


Figure 30: Search accounts

"Search All" button will fetch all the account records to the table. However, "Search" button will only fetch the details of a specific account.

References

Genero, M., Piattini, M. & Calero, C., 2002. *Empirical validation of class diagram metrics.* s.l., s.n., pp. 195-203.

Li, X., Liu, Z. & Jifeng, H., n.d. *A formal semantics of UML sequence diagram.* s.l., 2004, pp. 168-177.

Mathiassen, L. & Madsen, M. A., 2000. *Object-oriented Analysis & Design.* s.l.:s.n.

Oracle, 2021. *Java Documentation: JDBC Basics.* [Online] Available at: https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

Oracle, 2021. *Lesson: Object-Oriented Programming Concepts.* [Online] Available at: https://docs.oracle.com/javase/tutorial/java/concepts/index.html

Shen, W. & Liu, S., 2003. *Formalization, Testing and Execution of a Use Case Diagram.* s.l., s.n., p. 68–85.

W3Schools, 2021. *What is OOP.* [Online] Available at: https://www.w3schools.com/java/java_oop.asp