KHULNA UNIVERSITY

Project on Banking Management System

Course Title : **Advanced Programming Laboratory**

Course Code : **0714 02 CSE 2100**

Submitted To

Dr. Engr. Kazi Masudul Alam
Professor
CSE Discipline
Khulna University,
Khulna.

Submitted By

Name : **MD. Ishrak Dewan**
Student ID : **230212**
Year : **2nd**
Term : **1st**
Session : **2023-2024**
Discipline : **CSE**

Name : **Tamal Paul**
Student ID : **230213**
Year : **2nd**
Term : **1st**
Session : **2023-2024**
Discipline : **CSE**

➢ Introduction:

The Banking Management System is a software solution designed to handle a variety of banking operations, ranging from managing customer accounts to handling investments, loans, and transactions. The system is structured using the **Model-View-Controller (MVC)** design pattern to ensure separation of concerns, making it scalable, maintainable, and easy to extend. Each functionality of the banking system, such as account management, loan processing, transaction handling, and investment management, is controlled by specific controllers, which interact with their corresponding models and views to provide a user-friendly experience**.**

➢ Objectives:

1. **Account Management:** To provide the ability to manage customer accounts, including deposit, withdrawal, and balance display.
2. **Investment Management:** To handle customer investments, including adding new investments, viewing investments by customers, and displaying all investments.
3. **Loan Processing:** To manage loan operations such as adding loans, viewing loan details, and handling loan repayments.
4. **Transaction Processing:** To allow for processing and displaying of deposit and withdrawal transactions.
5. **Notification System:** To manage notifications related to banking activities, including sending alerts and messages to customers.
6. **Person Management:** To manage details of customers and employees, including adding new persons and displaying their information.
7. **Scalability & Extensibility:** The system is designed with scalability in mind, allowing for easy addition of new features in the future.

➢ Description:

This system is designed to manage various operations within a bank, including account management, investments, loan management, notifications, transactions, and people management. Each area of functionality is handled by separate controllers, models, and views, following the **MVC** architecture. The core components include:

1. Account Management:

- **AccountModel:** This class defines an account with fields like accountNumber, accountHolderName, and balance. It also includes methods for depositing and withdrawing money from an account.
- **AccountView:** This class interacts with the user, displaying account-related menus and prompting for actions like deposit or withdrawal amounts.
- **AccountController:** This class manages the logic for interacting with the AccountModel and updating the AccountView. It includes methods like deposit(), withdraw(), and displayAccountDetails(), though the actual logic is yet to be implemented.

2. Investment Management:

- **InvestmentModel**: Manages investments, storing information about each investment, such as investmentId, customerName, amount, and type. The model supports adding, removing, and retrieving investments.
- **InvestmentView:** Provides a user interface for managing investments, allowing users to add investments, display all investments, or view a customer's investments.
- **InvestmentController**: Handles the logic related to investments, such as adding, displaying, and removing investments.

3. Loan Management:

- **LoanModel**: This class manages loan data, including fields for loanId, customerName, loanAmount, interestRate, and loanTerm. It supports adding loans, retrieving loan details, and removing loans.
- **LoanView:** Displays loan-related menus and facilitates user input for loan operations, such as adding a loan, retrieving loan details, and viewing all

loans. It interacts with the user to collect loan data and displays information about loans.

- **LoanController:** Controls the loan operations, including methods to add loans, get loan details, view loan details, and remove loans.

4. Notification System:

- **NotificationModel:** It stores a list of notifications and provides methods for adding new notifications, retrieving all notifications, and sending notifications to users. The class also provides getter and setter methods for accessing and modifying the notification list.
- **NotificationView:** Displays a menu for interacting with the notification system and prompts for sending notifications.
- **NotificationController:** Controls the logic of managing notifications, such as sending messages and view sending messages.

5. Person Management:

- **PersonModel:** Represents a person, which could be a customer or an employee in the banking system. It stores essential personal information like ID, name, contact, email, and extra information. The class provides methods for accessing (getter) and modifying (setter) these attributes, as well as displaying person details.
- **PersonView:** Displays a menu for interacting with the person system and handles user input for adding or displaying people.
- **PersonController:** Controls the actions for managing persons (adding, viewing, and displaying details).

6. Transaction Management:

- **TransactionModel**: Manages transaction data, such as deposits and withdrawals. It keeps track of all transactions in a list and provides methods to process deposits and withdrawals.
- **TransactionView:** Displays transaction menus and facilitates user input for making deposits or withdrawals. It interacts with the user to collect transaction details and display relevant information, such as transaction history or messages.

- **TransactionController:** Manages transaction logic, including deposit, withdrawal and displayTransaction operations.

7. Main Program (Main Class):

- The Main class is where the program starts. It creates **instances of all the models, views, and controllers**, allowing the user to interact with the system through the MVC structure. The system is initialized, and different operations like account management, people management, loans, investments, and transactions can be performed.

❖ **System Flow**

1. **Initialization**: The Main class initializes the models (data), views (user interface), and controllers (logic) for each area of the system.
2. **Interaction**: The user interacts with the system through various views (like AccountView, InvestmentView, LoanView, etc.). Each view prompts the user for input, which is then passed to the respective controller.
3. **Processing**: The controller processes the input and updates the corresponding model. For example, the AccountController will deposit or withdraw money based on user input, updating the AccountModel.
4. **Display Results**: After processing, the view updates the user with the results (such as showing updated account details or transaction history).

❖ **Areas for Further Development**

The system is structured, but several key areas still need to be implemented:

1. **Logic Implementation**: Each method like deposit(), withdraw(), addInvestment(), etc., needs to be filled with actual logic to perform the operations.
2. **Persistence**: The system currently operates in memory. To make it more robust, data could be saved to a database or file system for persistence across sessions.
3. **Error Handling**: Implement proper error handling for various operations, such as insufficient balance for withdrawals or invalid input from the user.

❖ **Key Features**

- **Account Operations:** Deposit, withdrawal, and account balance display.
- **Investment Handling:** Ability to add, remove, and view investments.
- **Loan Management:** Add, view, and remove loans, along with calculating monthly repayments.
- **Transaction History:** Track and display deposit and withdrawal transactions.
- **User Notifications:** Ability to send and view notification services.
- **People Management:** Manage customers and employees, including adding and displaying their information.

➢ Summary:

The banking management system is a well-structured MVC-based application, where:

- The **Model** represents the data and business logic.
- The **View** handles the user interface.
- The **Controller** acts as an intermediary between the model and view, processing user inputs and updating the view.

The system is modular, allowing for extensions or modifications to the various banking services (account, loan, transaction, etc.) with minimal disruption to other parts of the codebase. The system also follows object-oriented principles, encapsulating responsibilities into specific classes, which is beneficial for future maintainability.

➢ Conclusion:

In conclusion, the Banking Management System provides an efficient and structured solution for managing various aspects of a banking operation. By following the **Model-View-Controller (MVC) design pattern**, the system ensures that different components—such as **business logic (models)**, **user interfaces (views)**, and **control flow (controllers)**—are neatly separated, making the system both scalable and maintainable.The various functionalities—account management, investment tracking, loan processing, transaction handling, and more—serve to make banking

operations more streamlined, secure, and user-friendly. With a well-organized architecture, the system is designed to handle future extensions, such as integrating new services, adding features, or adapting to changing requirements. Furthermore, this system can be a foundation for building real-world banking solutions with enhanced features like data persistence, real-time processing, and more sophisticated user interfaces. By maintaining a modular and robust structure, this system is poised to offer an adaptable, scalable, and reliable banking management solution.