DEPARTMENT OF
SOFTWARE ENGINEERING

PROJECT REPORT

SOFTWARE CONSTRUCTION LAB

PREPARED BY
MUHAMMAD MOOSA KHALIL (01-131222-035)
TAYYAB AAMIR ALI (01-131222-048)
RAFIA TARIQ (01-131222-039)

SUBMIT TO
MAAM RAFIA
MAAM RAHEELA

ABSTRACT

The Bank Management System is a comprehensive desktop application designed to streamline and automate fundamental banking operations such as user authentication, account creation, fund deposits, withdrawals, and balance inquiries. Developed using Java Swing for the graphical user interface and SQL Server for backend data management, the system ensures secure, efficient, and reliable performance.

This project leverages a modular, 2-tier architecture, separating the user-facing client tier from the database tier for enhanced scalability and maintainability. Key features include robust authentication mechanisms, real-time transaction management, and an intuitive user interface designed for a seamless user experience.

By employing modern software development practices, the system is equipped to handle the requirements of small to medium-sized financial institutions. Its structured design, emphasis on data integrity, and use of advanced tools like JDBC and SQL Server make it a practical solution for both academic learning and real-world banking scenarios. This report provides detailed insights into the system's architecture, features, and implementation, along with use case scenarios and database integration.

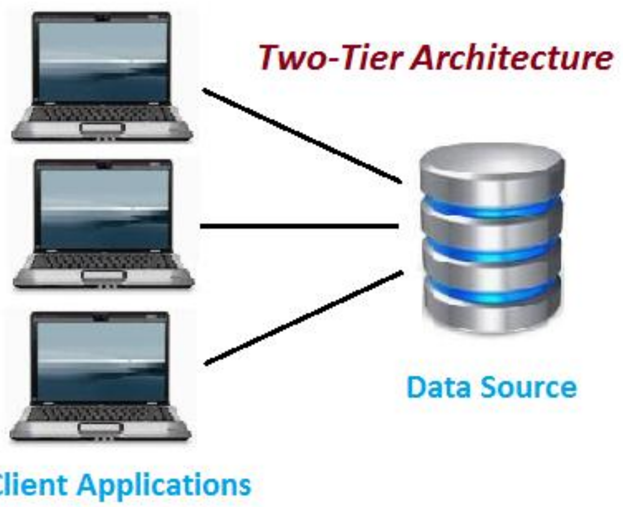
INTRODUCTION

The Bank Management System is a robust, Java-based desktop application meticulously designed to streamline essential banking operations. It serves as a comprehensive solution for performing tasks such as user authentication, account creation, fund deposits, withdrawals, and balance inquiries.

Built with the Java Swing framework, the system provides an intuitive and user-friendly graphical interface, ensuring seamless interaction for users of varying technical expertise. The backend integrates with a SQL Server database, offering secure and efficient data storage, management, and retrieval. This ensures the integrity and reliability of sensitive financial information.

Designed with modularity and scalability in mind, the Bank Management System is ideal for small to medium-sized financial institutions and serves as a practical learning model for students exploring modern software development and database integration practices.

SYSTEM ARCHITECTURE



The Bank Management System is designed with a 2-Tier Architecture, ensuring simplicity and efficiency in operation while maintaining a clear separation of concerns. The two primary tiers are:

1. **Client Tier**
   The client tier serves as the user-facing component of the system, developed using the Java Swing framework. This tier provides an interactive and visually appealing Graphical User Interface (GUI) that allows users to perform banking tasks with ease. The GUI is responsive, user-friendly, and capable of handling input validations to ensure smooth user interactions.

2. **Database Tier**
   The backend database is implemented using SQL Server, which acts as the core repository for all sensitive and operational data. This tier handles the storage and management of:
   - User credentials (card numbers and PINs)
   - Account details
   - Transaction histories (deposits, withdrawals, and balance records)

The application architecture is modular, with dedicated classes encapsulating specific functionalities (e.g., login, deposits, withdrawals). This design promotes code reusability, readability, and maintainability. A centralized database connection class (Connn) ensures secure and efficient communication between the client tier and the database, adhering to best practices for resource management and query execution.

This architecture strikes a balance between simplicity for smaller-scale implementations and the potential for future scalability in larger deployments.

LIBRARIES AND TECHNOLOGIES USED

The Bank Management System leverages a range of powerful technologies and libraries to ensure a smooth and efficient user experience, robust database interaction, and overall system performance. Below is a detailed explanation of the key libraries and technologies used:

1. Java Swing

- **Purpose**: Java Swing is the primary library used for developing the graphical user interface (GUI) of the application. It provides a set of lightweight, platform-independent components for building interactive user interfaces.
- **Key Features**:
  - **Components**: Buttons, text fields, labels, panels, etc., that are used to create the interface.
  - **Event Handling**: Allows the application to respond to user actions (e.g., clicks, key presses).
  - **Layouts**: Provides flexible layout managers like BorderLayout, FlowLayout, etc., for arranging GUI components in a structured way.
  - **Customization**: Offers methods to customize the appearance and behavior of components.
- **Libraries Used**:
  - javax.swing.*: Core Swing components.

- java.awt.*: For event handling and basic window components.

2. JDBC (Java Database Connectivity)

- **Purpose**: JDBC is used for connecting to the SQL Server database, executing SQL queries, and retrieving results. It enables interaction between the Java application and the relational database.
- **Key Features**:
  - **Database Connectivity**: Establishes a connection to the database using the DriverManager or DataSource.
  - **Statement and PreparedStatement**: Executes SQL queries and prepared statements for CRUD (Create, Read, Update, Delete) operations.
  - **ResultSet**: Retrieves and processes data returned from SQL queries.
  - **Transaction Management**: Supports commit and rollback mechanisms to ensure data integrity during transactions.
- **Libraries Used**:
  - java.sql.*: Includes classes like Connection, Statement, PreparedStatement, ResultSet, and SQLException.

3. JDK 1.8+ (Java Development Kit)

- **Purpose**: JDK provides the necessary tools and libraries for compiling and running Java applications. It is essential for the development, testing, and execution of the Bank Management System.
- **Key Features**:
  - **Java Compiler**: Translates Java source code into bytecode that can run on the Java Virtual Machine (JVM).
  - **Standard Libraries**: Includes essential libraries like java.lang, java.util, java.sql, and others that are fundamental for building Java applications.
  - **Java 8 Features**: Introduces advanced features like Lambda expressions, Streams API, and the java.time package for modern date and time handling.

4. Calendar API (java.util.Calendar)

- **Purpose**: The Calendar class is used for managing date and time within the application. This is particularly useful when implementing features like account creation dates, transaction timestamps, and expiration dates for certain banking operations.

- **Key Features**:
    - **Date Calculation**: Allows manipulation of dates (e.g., adding or subtracting days).
    - **Date Formatting**: Provides methods to format dates into readable strings.
    - **Time Zone Handling**: Supports different time zones for internationalization.
- **Libraries Used**:
    - java.util.Calendar: Main class for date and time manipulation.
    - java.text.SimpleDateFormat: Used for formatting and parsing date strings.

CLASSES AND RESPONSIBILITIES

1. **LOGIN**

    Purpose:

    - Handles user authentication through card number and PIN.

    Responsibilities:

    1. **Displays a login form:**
        - Shows a form with fields for card number and PIN, along with buttons for signing in, clearing fields, and signing up.
        - Utilizes images and background for a visually appealing interface.
    2. **Validates user credentials:**
        - Connects to the database to check if the entered card number and PIN match an existing user.
        - Executes SQL queries to verify user credentials.
    3. **Handles user interactions:**
        - **Sign In**: Validates entered credentials and redirects to the main application interface if valid.
        - **Clear**: Clears the entered card number and PIN fields.
        - **Sign Up**: Opens the signup form for new user registration.
    4. **Redirects to the main application interface on successful login:**

    - If the credentials are correct, hides the login form and opens the main application interface.

2. **SIGNUP**

   Purpose:

   - The Signup class is responsible for handling the user registration process by collecting personal details and validating them before saving to the database.

   Responsibilities:

   1. **Displays a Signup Form:**
      - Shows a form with fields for name, father's name, date of birth, gender, email address, marital status, address, and city.
      - Utilizes images and background for a visually appealing interface.
   2. **Validates User Input:**
      - Ensures all fields are filled out.
      - Validates the length of name and father's name.
      - Ensures the user is at least 18 years old.
      - Restricts input in name and father's name fields to letters and spaces only.
   3. **Handles User Interactions:**
      - **Next Button:** Collects the data entered, validates it, and if valid, saves it to the database and proceeds to the next signup step.
   4. **Saves User Data:**
      - Connects to the database and inserts the user's data into the signup table.

3. **SIGNUP2**:

   Purpose:

   - The Signup2 class is responsible for collecting additional user details as part of the signup process.

   Responsibilities:

   1. **Displays an Additional Details Form:**

   - Shows a form with fields for religion, category, income, education, occupation, CNIC number, senior citizen status, and existing account status.
   - Utilizes images and a custom background for a visually appealing interface.

2. **Validates User Input:**
   - Ensures all fields are filled out.
   - Validates the CNIC number format.
   - Ensures appropriate data types are entered.
3. **Handles User Interactions:**
   - **Next Button:** Collects the data entered, validates it, and if valid, saves it to the database and proceeds to the next signup step.
4. **Saves User Data:**
   - Connects to the database and inserts the additional user details into the Signuptwo table.

4. **SIGNUP3**:

Purpose:

- The Signup3 class handles the final step of the signup process, collecting account details and additional service requests from the user.

Responsibilities:

- **Displays an Account Details Form:**
  - Shows a form with options for account type, card number, PIN, and services required.
  - Utilizes images and a custom background for a visually appealing interface.
- **Validates User Input:**
  - Ensures all necessary fields are filled out.
  - Validates the selection of an account type.

- **Handles User Interactions:**
    - **Submit Button:** Collects the data entered, validates it, and if valid, saves it to the database and proceeds to the deposit form.
    - **Cancel Button:** Exits the signup process.
- **Saves User Data:**
    - Connects to the database and inserts the collected account details and service requests into the signupthree and login tables.

5. DEPOSIT

Purpose:

- The Deposit class is responsible for handling the deposit functionality in the bank management system. It allows users to deposit a specified amount into their account.

Responsibilities:

1. **Displays a Deposit Form:**
    - Shows a form where users can enter the amount they wish to deposit.
    - Utilizes images and a custom background for a visually appealing interface.
2. **Validates User Input:**
    - Ensures the input amount is numeric.
    - Validates that the deposit amount is within the allowed range (e.g., between Rs. 1000 and Rs. 50000).
3. **Handles User Interactions:**
    - **Deposit Button**: Validates the amount, saves the deposit transaction to the database, and provides confirmation to the user.
    - **Back Button**: Returns the user to the main application interface without performing any deposit.
4. **Saves User Data:**
    - Connects to the database and inserts the deposit transaction details into the bank table.

6. WITHDRAWL

Purpose:

- The Withdrawl class handles the withdrawal functionality in the bank management system. It allows users to withdraw a specified amount from their account.

Responsibilities:

1. **Displays a Withdrawal Form:**
    - Shows a form where users can enter the amount they wish to withdraw.
    - Utilizes images and a custom background for a visually appealing interface.
2. **Validates User Input:**
    - Ensures the input amount is numeric.
    - Validates that the withdrawal amount is within the allowed range (e.g., between Rs. 500 and Rs. 20000).
    - Checks if the account has sufficient balance for the withdrawal.

3 . Handles User Interactions:

1. **Withdraw Button:** Validates the amount, checks for sufficient balance, saves the withdrawal transaction to the database, and provides confirmation to the user.
2. **Back Button:** Returns the user to the main application interface without performing any withdrawal.

4. Saves User Data:

1. Connects to the database and inserts the withdrawal transaction details into the bank table.

7. FASTCASH

Purpose:

- The FastCash class handles quick withdrawal transactions in the bank management system, providing predefined amounts for users to withdraw quickly.

Responsibilities:

1. **Displays a FastCash Withdrawal Form:**
    - Shows buttons for predefined withdrawal amounts (e.g., Rs. 100, Rs. 500, Rs. 1000, etc.).
    - Utilizes images and a custom background for a visually appealing interface.
2. **Validates User Input:**
    - Ensures the selected amount can be withdrawn based on the user's current balance.
3. **Handles User Interactions:**
    - **Withdrawal Buttons:** Processes the withdrawal for the selected amount if sufficient balance is available.
    - **Back Button:** Returns the user to the main application interface without performing any withdrawal.
4. **Saves User Data:**
    - Connects to the database and inserts the withdrawal transaction details into the bank table.

8. BALANCEENQUIRY

Purpose:

- The BalanceEnquiry class is responsible for displaying the current balance of the user in the bank management system.

Responsibilities:

1. **Displays the Balance Enquiry Form:**
    - Shows a form with the current balance of the user.
    - Utilizes images and a custom background for a visually appealing interface.

2. **Fetches User Balance:**
    - Connects to the database and retrieves the user's current balance based on transactions.
3. **Handles User Interactions:**
    - Back Button: Returns the user to the main application interface.
4. **Displays User Balance:**
    - Calculates and displays the user's balance by summing deposits and subtracting withdrawals.

9. **PIN CLASS**

Purpose:

- **The Pin class is responsible for allowing users to change their ATM PIN in the bank management system.**

Responsibilities:

1. **Displays the PIN Change Form:**
    - Shows a form where users can enter a new PIN and confirm it.
    - Utilizes images and a custom background for a visually appealing interface.
2. **Validates User Input:**
    - Ensures the new PIN is exactly 4 digits long.
    - Ensures the new PIN matches the confirmed PIN.
    - Checks if the new PIN is already assigned to another user.
3. **Handles User Interactions:**
    - **Change Button:** Validates the new PIN and updates it in the database if valid.
    - Back Button: Returns the user to the main application interface without changing the PIN.
4. **Updates User Data:**
    - Connects to the database and updates the PIN in the bank, login, and signupthree tables.

10. MAIN CLASS

Purpose:

- The main_Class provides the primary interface for users to interact with various banking services like deposit, withdrawal, balance enquiry, etc.

Responsibilities:

1. **Displays the Main Menu:**
   - Shows buttons for different banking operations (e.g., Deposit, Cash Withdrawal, Fast Cash, Mini Statement, PIN Change, Balance Enquiry, Exit).
   - Utilizes images and a custom background for a visually appealing interface.
2. **Handles User Interactions:**
   - **Deposit Button:** Opens the deposit form.
   - **Cash Withdrawal Button:** Opens the withdrawal form**.**
   - **Fast Cash Button:** Opens the fast cash form.
   - **Mini Statement Button:** Opens the mini statement form.
   - **PIN Change Button:** Opens the PIN change form.
   - **Balance Enquiry Button:** Opens the balance enquiry form.
   - **Exit Button:** Exits the application.

11. CONN CLASS

Purpose:

- To establish a connection with the SQL Server database.
- To create a statement object for executing SQL queries.

Responsibilities:

1. **Load the JDBC Driver:**
   - Loads the Microsoft SQL Server JDBC driver using Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver").
2. **Establish Database Connection:**
   - Connects to the SQL Server using the provided connection string and credentials via DriverManager.getConnection.
3. **Create SQL Statement:**
   - Creates a Statement object for executing SQL queries using connection.createStatement().

12. MINI CLASS

Purpose:

- The Mini class displays the mini statement of the user's transactions in the bank management system.

Responsibilities:

1. **Displays the Mini Statement:**
   - Shows a list of recent transactions including date, type, and amount.
   - Utilizes a custom background for a visually appealing interface.
2. **Fetches User Transactions:**
   - Connects to the database and retrieves the user's recent transactions.
   - Summarizes the total balance based on deposits and withdrawals.
3. **Handles User Interactions:**

- **Exit Button:** Closes the mini statement window.

SQL DATABASE

Database Overview

The BankManagementSystem database is designed to manage various aspects of a banking application, including user registration, login, and transaction management. The database consists of multiple tables to store user details, transaction records, and other relevant information.

Tables and Their Structure

- **signup**
  - **Purpose**: To store personal details of users during the registration process.
  - **Columns**:
    - formno (varchar(20)): Form number for registration.
    - name (varchar(20)): User's name.
    - father_name (varchar(20)): User's father's name.
    - dob (varchar(20)): Date of birth.
    - gender (varchar(20)): Gender.
    - email (varchar(30)): Email address.
    - marital_status (varchar(20)): Marital status.
    - address (varchar(40)): Address.
    - city (varchar(25)): City.
    - pincode (varchar(20)): Pincode.
    - state (varchar(25)): State.
- **signuptwo**
  - **Purpose**: To store additional details of users during the second step of registration.
  - **Columns**:
    - formno (varchar(20)): Form number for registration.
    - religion (varchar(20)): Religion.
    - category (varchar(20)): Category.
    - income (varchar(20)): Income.
    - education (varchar(20)): Educational qualification.

        - occupation (varchar(20)): Occupation.
        - Cnic (varchar(20)): Aadhar number.
        - seniorcitizen (varchar(20)): Senior citizen status.
        - existingaccount (varchar(20)): Existing account status.

- **signupthree**
    - **Purpose**: To store account details and facilities selected by the user during the final step of registration.
    - **Columns**:
        - formno (varchar(20)): Form number for registration.
        - accountType (varchar(40)): Type of account.
        - cardnumber (varchar(25)): Card number.
        - pin (varchar(10)): PIN.
        - facility (varchar(100)): Facilities opted by the user.
- **login**
    - **Purpose**: To manage user login information.
    - **Columns**:
        - formno (varchar(20)): Form number for registration.
        - cardnumber (varchar(25)): Card number.
        - pin (varchar(10)): PIN.
    - **Note**: The column cardnumber was renamed to card_number.
- **bank**
    - **Purpose**: To store transaction details.
    - **Columns**:
        - pin (varchar(10)): PIN used for the transaction.
        - date (varchar(50)): Date of the transaction.
        - type (varchar(20)): Type of transaction (Deposit or Withdrawal).
        - amount (varchar(20)): Amount involved in the transaction

- The BankManagementSystem database is a comprehensive structure designed to handle user registration, authentication, and transaction management efficiently. Each table plays a crucial role in storing and managing different aspects of user information and transactions. The use of SQL Server ensures data integrity, security, and efficient query execution, providing a robust backend for the banking application.

KEY FEATURES AND FUNCTIONALITIES

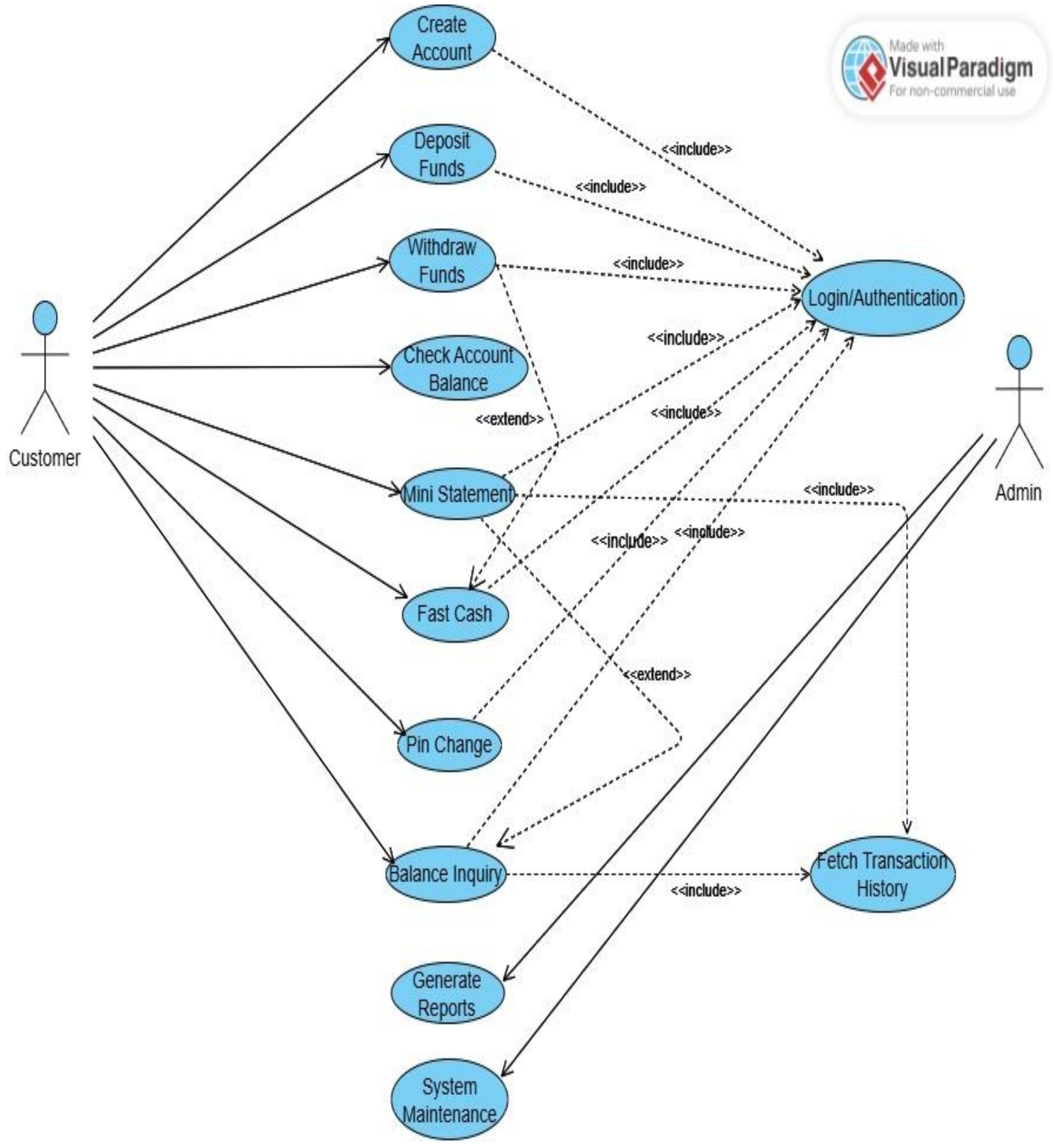
1. **Login System**
    - **Secure Authentication**:
        - Utilizes card number and PIN for user identification and login.
        - Implements hashing and salting of PINs for added security.
    - **Error Handling**:
        - Provides clear error messages for invalid credentials.
        - Limits the number of failed login attempts to prevent brute-force attacks.
2. **Transaction Management**
    - **Deposit and Withdrawal**:
        - Input validation to ensure amounts are within acceptable ranges.
        - Confirmation prompts to avoid accidental transactions.
    - **Fast Cash Option**:
        - Predefined withdrawal amounts for quick transactions.
        - Option to customize fast cash amounts based on user preference.
    - **Transaction History**:
        - Provides a detailed history of all transactions, accessible to the user.
        - Real-time updates to ensure the latest information is displayed.

3. **User Registration**
   - **Multi-Step Process**:
     - Step-by-step user-friendly registration with form validation.
     - Validation for unique email addresses and phone numbers to prevent duplicates.
   - **Progress Indicators**:
     - Visual indicators to show the user their progress through the registration steps.
   - **Field Validation**:
     - Real-time validation for input fields to ensure data integrity.
     - Custom error messages to guide users in providing correct information.
4. **Database Integration**
   - **Persistent Data Storage**:
     - Securely stores user data and transaction history in a SQL Server database.
     - Regular backups to prevent data loss.
   - **Efficient Query Execution**:
     - Uses prepared statements to prevent SQL injection attacks.
     - Optimized queries for fast data retrieval and transaction processing.
   - **Scalability**:
     - Designed to handle increasing amounts of data as the user base grows.
     - Implements indexing for faster query performance.

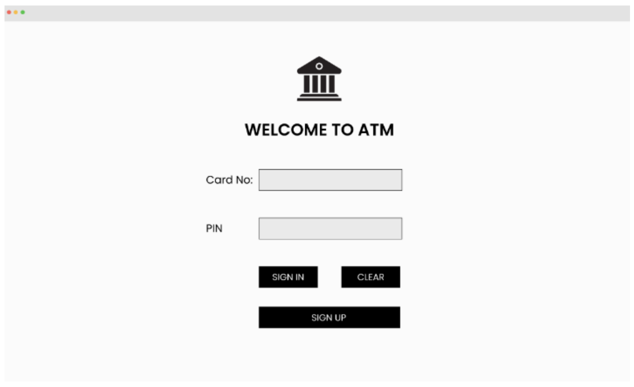
5. **User Interface**

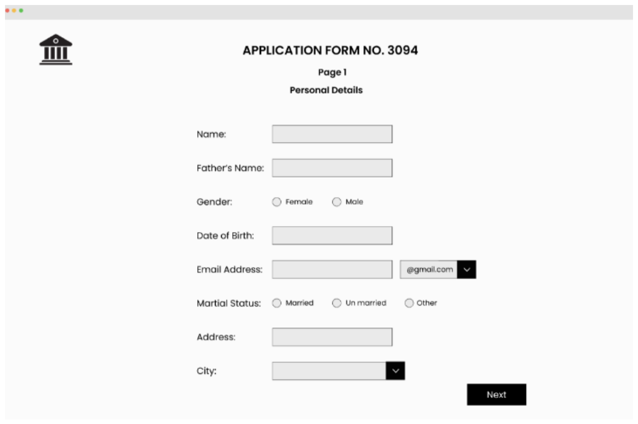
- **Clean and Interactive GUI**:
  - Built using Java Swing for a responsive and user-friendly interface.
  - Consistent design language and easy navigation.
- **Visual Aids**:
  - Icons and images to enhance the user experience and make navigation intuitive.
  - Tooltips to provide additional context and help for UI elements.
- **Accessibility**:
  - Keyboard shortcuts for common actions.
  - High-contrast mode and support for screen readers to ensure accessibility for all users.

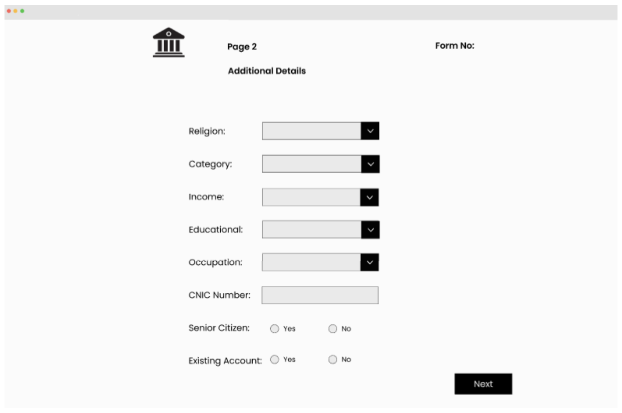
USE CASE DIAGRAM

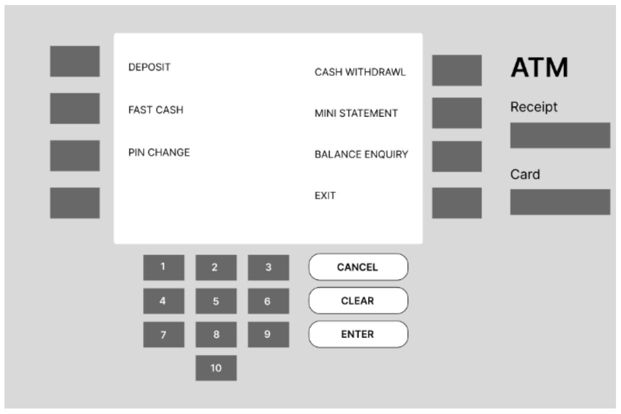


PROTOTYPES

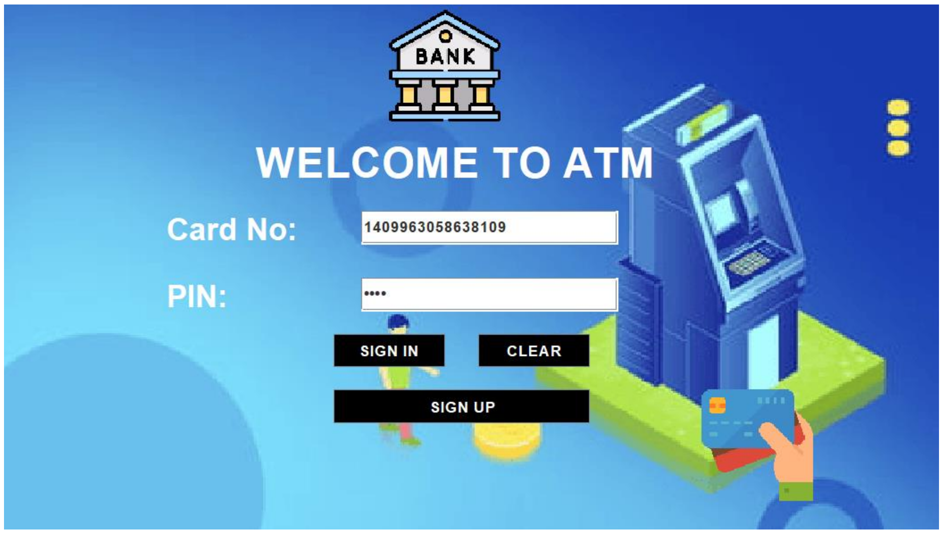

WELCOME TO ATM
Card No:
PIN
SIGN IN
CLEAR
SIGN UP


APPLICATION FORM NO. 3094
Page 1
Personal Details
Name:
Father's Name:
Gender:
Female
Male
Date of Birth:
Email Address:
@gmail.com
Martial Status:
Married
Un married
Other
Address:
City:
Next


Page 2
Form No:
Additional Details
Religion:
Category:
Income:
Educational:
Occupation:
CNIC Number:
Senior Citizen:
Yes
No
Existing Account:
Yes
No
Next

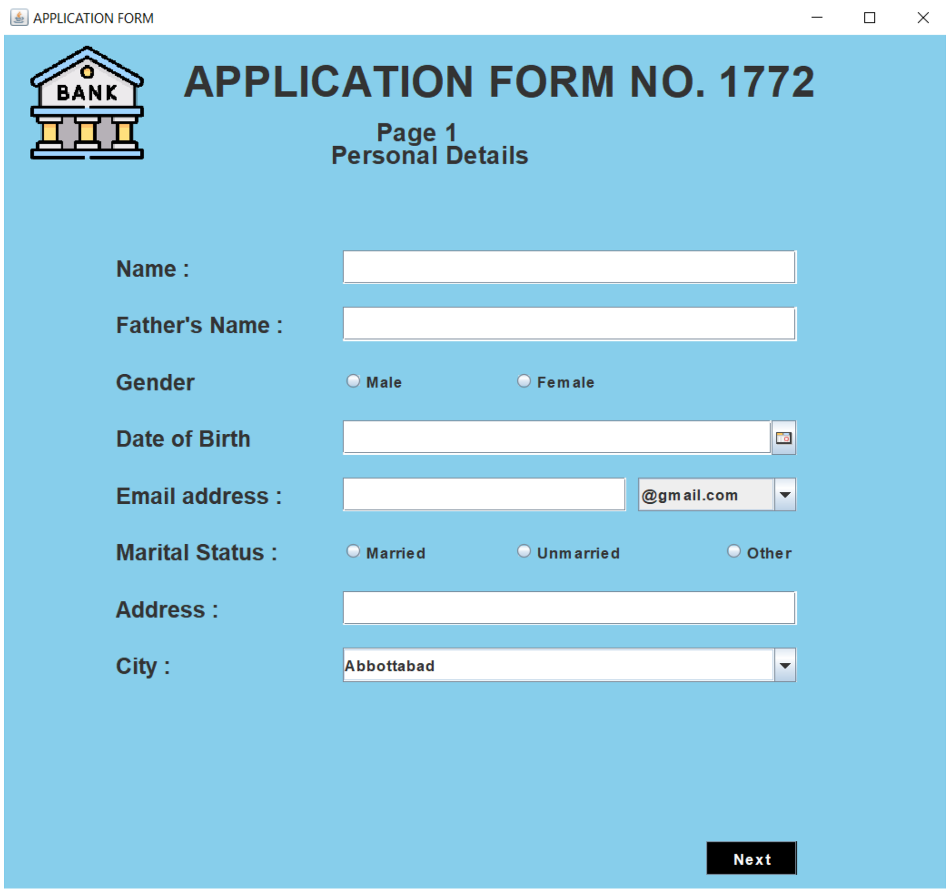

DEPOSIT
CASH WITHDRAWL
FAST CASH
MINI STATEMENT
PIN CHANGE
BALANCE ENQUIRY
EXIT
ATM
Receipt
Card
1
2
3
CANCEL
4
5
6
CLEAR
7
8
9
ENTER
10

PROJECT UI SNIPPETS



Card No: 1409963058638109

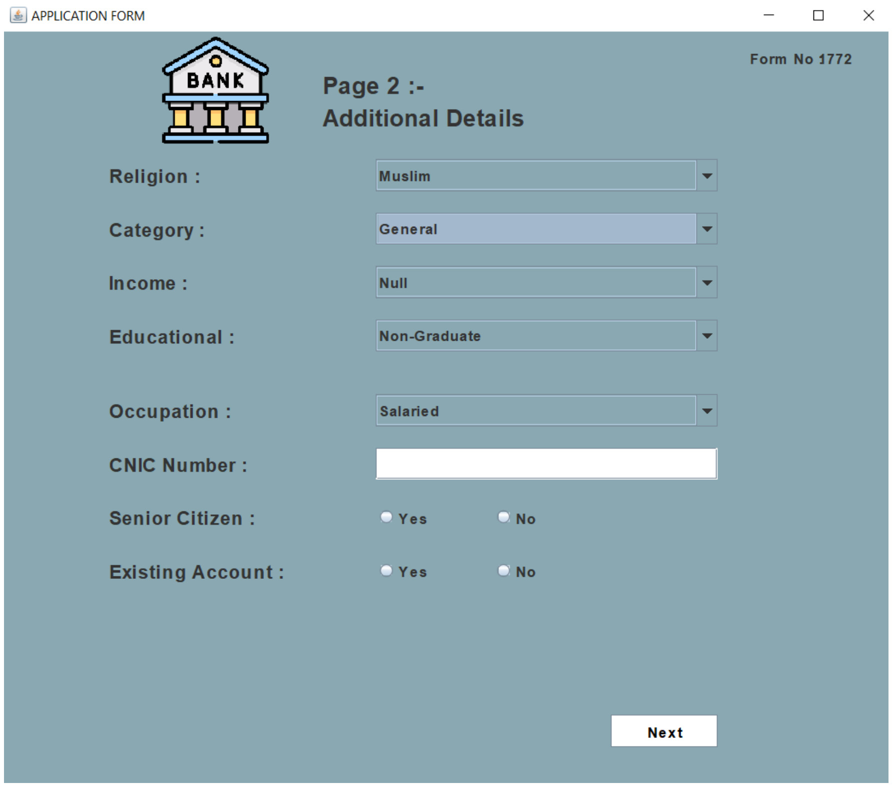
PIN: ••••



APPLICATION FORM NO. 1772

Page 1
Personal Details

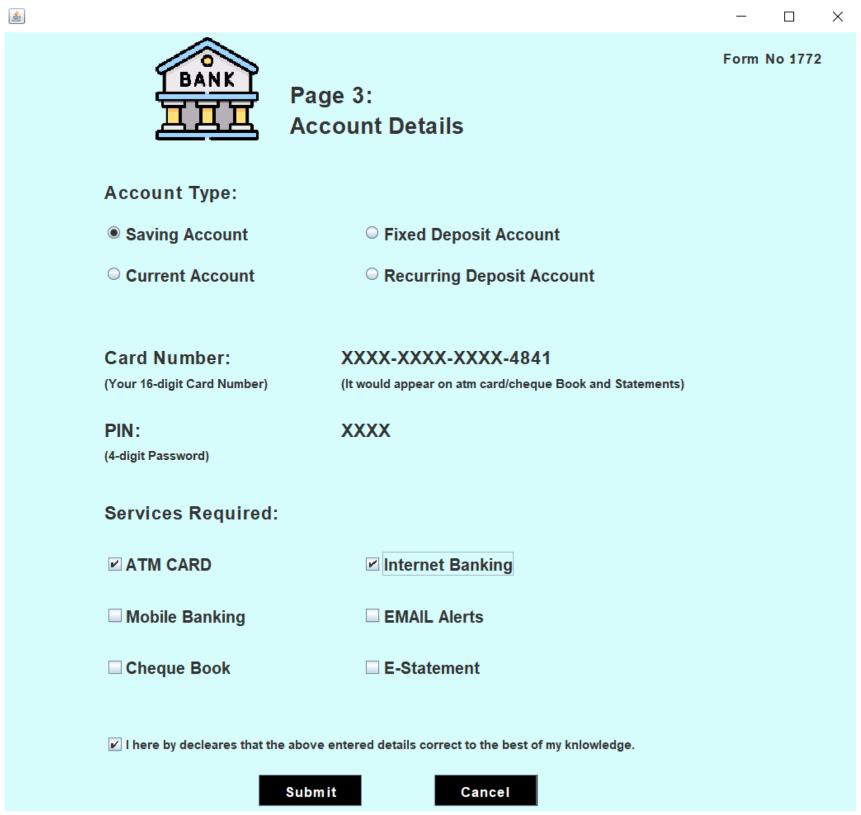
Name :	
Father's Name :	
Gender	Male Female
Date of Birth	
Email address :	@gmail.com
Marital Status :	Married Unmarried Other
Address :	
City :	Abbottabad


APPLICATION FORM

Form No 1772

Page 2 :-
Additional Details

Religion :	Muslim
Category :	General
Income :	Null
Educational :	Non-Graduate
Occupation :	Salaried
CNIC Number :	
Senior Citizen :	Yes No
Existing Account :	Yes No


Form No 1772

Page 3:
Account Details

Account Type:

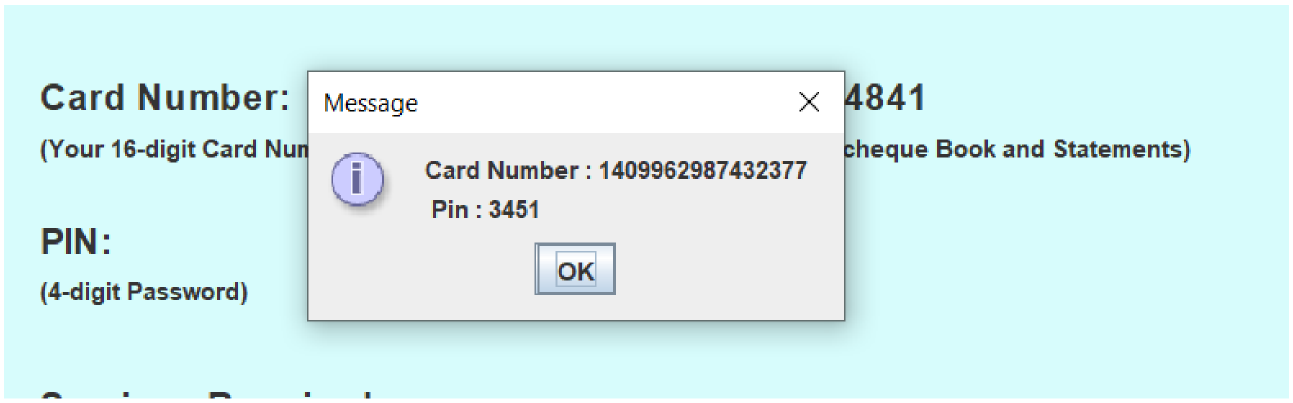
[✓ Saving Account] Fixed Deposit Account
Current Account Recurring Deposit Account

Card Number: (Your 16-digit Card Number)	XXXX-XXXX-XXXX-4841 (It would appear on atm card/cheque Book and Statements)
PIN: (4-digit Password)	XXXX

Services Required:

[✓ ATM CARD] [✓ Internet Banking]
Mobile Banking EMAIL Alerts
Cheque Book E-Statement

[✓ I here by decleares that the above entered details correct to the best of my knlowledge.]



Card Number: 4841

(Your 16-digit Card Num cheque Book and Statements)

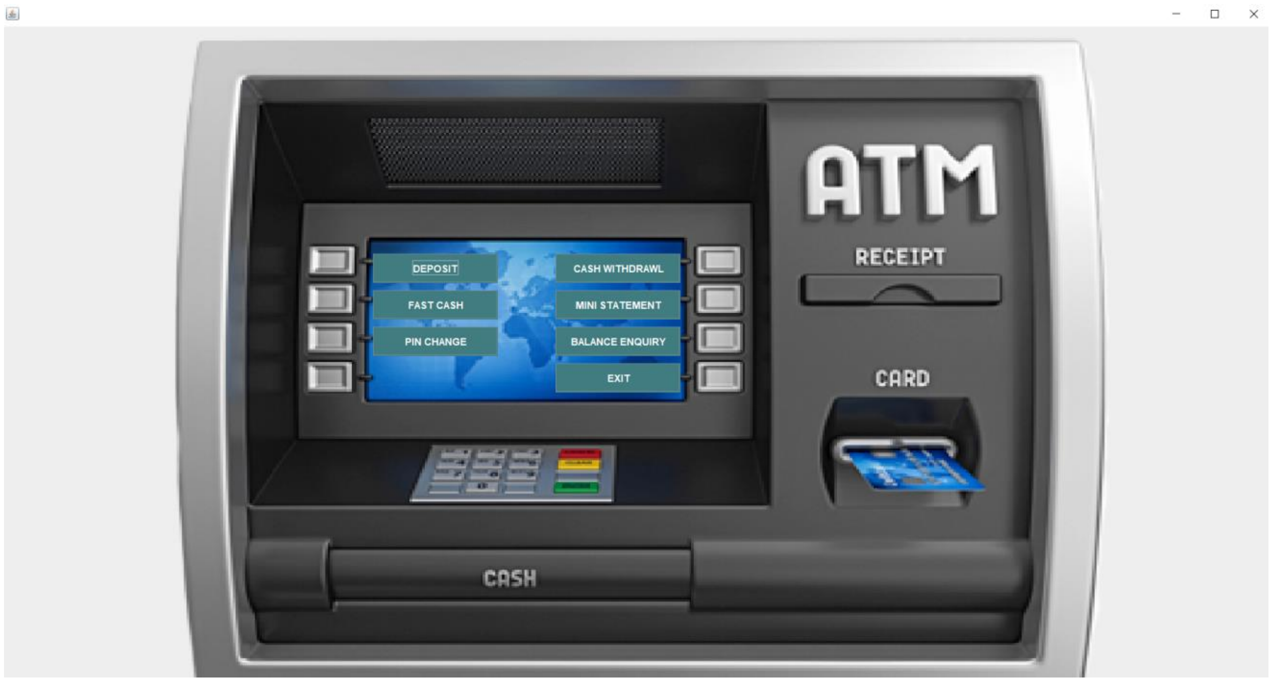
PIN:

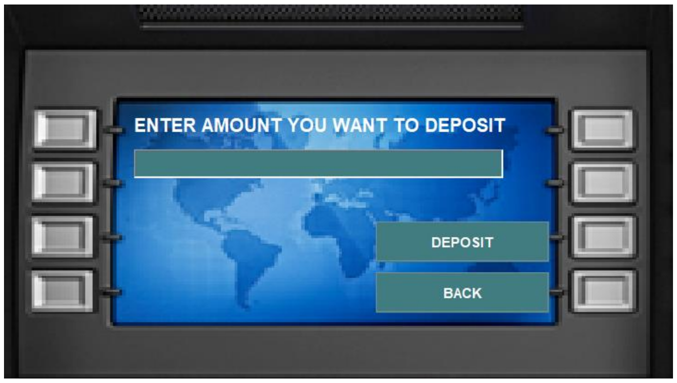
(4-digit Password)

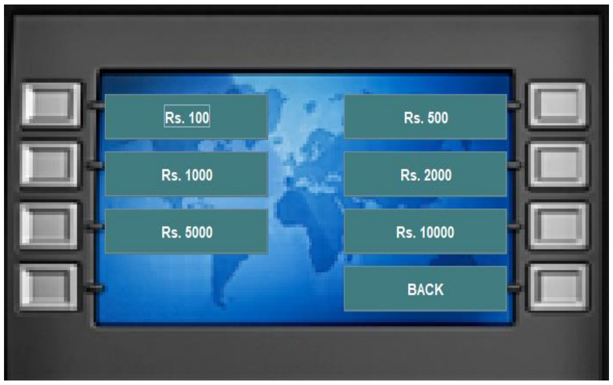
Message

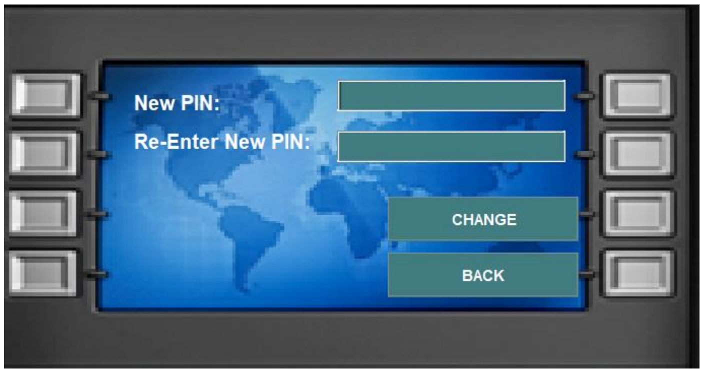
Card Number : 1409962987432377

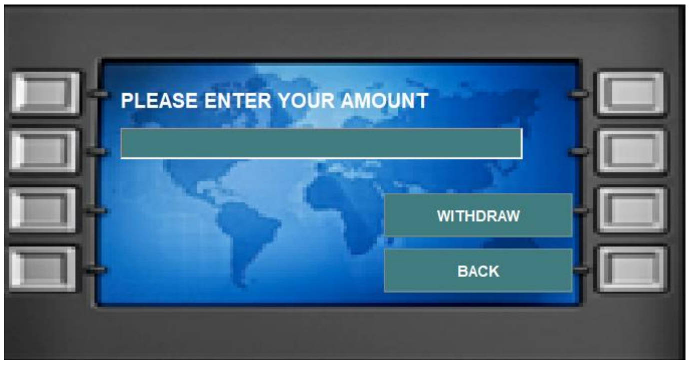
Pin : 3451

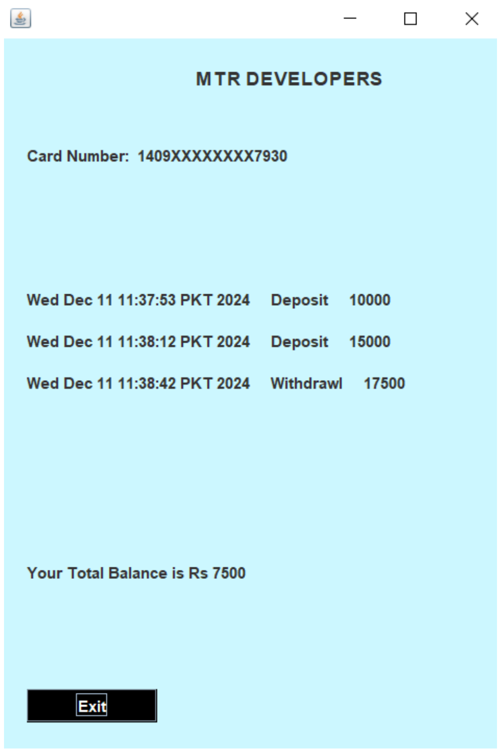










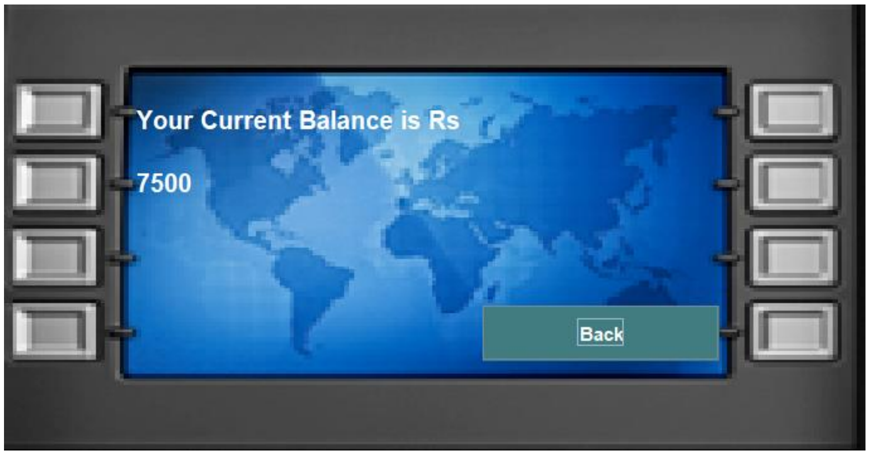


MTR DEVELOPERS

Card Number: 1409XXXXXXXX7930

```
Wed Dec 11 11:37:53 PKT 2024    Deposit     10000
Wed Dec 11 11:38:12 PKT 2024    Deposit     15000
Wed Dec 11 11:38:42 PKT 2024    Withdrawl    17500
```

Your Total Balance is Rs 7500



Your Current Balance is Rs 7500

CONCLUSION

The Bank Management System project effectively demonstrates the design and implementation of a comprehensive desktop application tailored to meet the operational requirements of small to medium-sized financial institutions. By integrating a robust graphical user interface using Java Swing with a secure and efficient SQL Server backend, the system delivers a seamless banking experience for both users and administrators.

Key accomplishments of the project include the development of modular functionalities for core banking operations such as user authentication, account creation, deposits, withdrawals, and balance inquiries. The use of a 2-tier architecture ensures a clear separation of concerns, enhancing maintainability and scalability. Moreover, the implementation of features such as fast cash withdrawals, detailed transaction histories, and PIN management addresses critical user needs while ensuring operational security.

The focus on modern software practices, including the use of JDBC for database connectivity and prepared statements to prevent SQL injection, highlights the system's adherence to best practices in software development. Additionally, the GUI's intuitive design fosters accessibility, accommodating users with diverse technical proficiencies.

From an academic perspective, this project provides a practical understanding of building enterprise-grade applications using Java and SQL. The structured approach to system design, database integration, and user experience ensures that the application is not only functional but also a viable learning model for aspiring developers.

Future enhancements could include transitioning to a 3-tier architecture for improved scalability, incorporating mobile and web-based interfaces, and leveraging cloud-based solutions for database management to support larger user bases. Despite these potential advancements, the current implementation stands as a robust, secure, and user-friendly solution that meets its intended goals.

In conclusion, the Bank Management System project successfully combines functionality, security, and usability, offering a strong foundation for further innovation and deployment in real-world banking scenarios.