BANK MANAGEMENT SYSTEM

For the Evaluation of
Project Mode-CS23333 Object Oriented Programming Using Java

Submitted by

JANANI K (231001070)

LOKESHWARI S (231001103)

Mini Project
November 2024

Department of Information Technology

Rajalakshmi Engineering College,Thandalam

BONAFIDE CERTIFICATE

Certified that this project report titled "**BANK MANAGEMENT SYSTEM"**
is the bonafide work of **JANANI K(231001070),LOKESHWARI S (231001103)**who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SUPERVISOR	**HEAD/IT**
Ms.T.Sangeetha	Dr.P.Valarmathie

Date:

ABSTRACT

The Banking Management System using JDBC (Java Database Connectivity) is a software solution designed to automate and optimize banking operations. This system connects Java-based applications with a relational database to efficiently manage core functionalities, including account creation, fund transfers, balance inquiries, and transaction history.

By leveraging JDBC, the system ensures real-time data processing, portability, and seamless integration with existing database management systems. It provides an intuitive interface for users and administrators, enabling secure and reliable operations. Key features include robust data handling, multi-user support, and enhanced security mechanisms such as encryption and access control.

This report highlights the system's architecture, database design, and implementation process, emphasizing its scalability and potential to improve banking efficiency. The proposed solution demonstrates the capability to streamline banking processes, reduce human errors, and enhance customer satisfaction.

TABLE OF CONTENTS

CHAPTER - 1
INTRODUCTION

1.1 MOTIVATION

Traditional banking systems are often inefficient, prone to human error, and incapable of handling large volumes of real-time transactions. This project aims to modernize banking operations by automating core functionalities through a well-structured Bank Management System using JDBC.

1.2 EXISTING SYSTEM

1. Manual handling of account and transaction records.
2. Limited accessibility and security.
3. Lack of real-time updates and analytical tools.

1.3 PROJECT OBJECTIVES

1. Manual handling of account and transaction records.
2. Limited accessibility and security.
3. Lack of real-time updates and analytical tools.

1.4 PROPOSED SYSTEM

The proposed Bank Management System (BMS) is a software solution designed to automate and streamline core banking operations using Java and JDBC for real-time database integration. This system aims to enhance efficiency, security, and user experience by providing a unified platform for managing accounts, transactions, and customer details.

Key Features of the Proposed System

1. Account Management

Creation, updating, and deletion of accounts.

Maintenance of customer profiles, including contact information and account details.

2. Transaction Management

Support for deposits, withdrawals, and fund transfers.

Real-time transaction tracking with balance updates.

Generation of transaction statements and summaries.

3. User Roles

Role-based access control for administrators, employees, and customers.

Enhanced security by restricting unauthorized access.

4. Report Generation

Financial reports for bank administrators.

Account statements for customers.

5. Scalability

Designed to handle a growing customer base and increased transaction volumes.

.

CHAPTER - 2
SYSTEM DESIGN

2.1 INTRODUCTION

The Bank Management System (BMS) is designed to address the complexities of modern banking operations by integrating advanced database management techniques with an efficient, modular architecture. The system is structured to ensure scalability, reliability, and security, while maintaining ease of use for customers, bank staff, and administrators.

The design adopts a three-tier architecture, which separates the presentation, application, and data layers to ensure maintainability and flexibility. Each component of the system is engineered to handle specific responsibilities, streamlining operations and enabling seamless interaction between users and the database.

By leveraging Java Database Connectivity (JDBC) and relational database principles, the system ensures real-time data processing,

2.2 SYSTEM ARCHITECTURE

1. Client Layer: User interface (UI) built with Java Swing or JavaFX for interacting with customers and staff.

2. Controller Layer: Java classes to handle UI events, process inputs, and invoke relevant services.

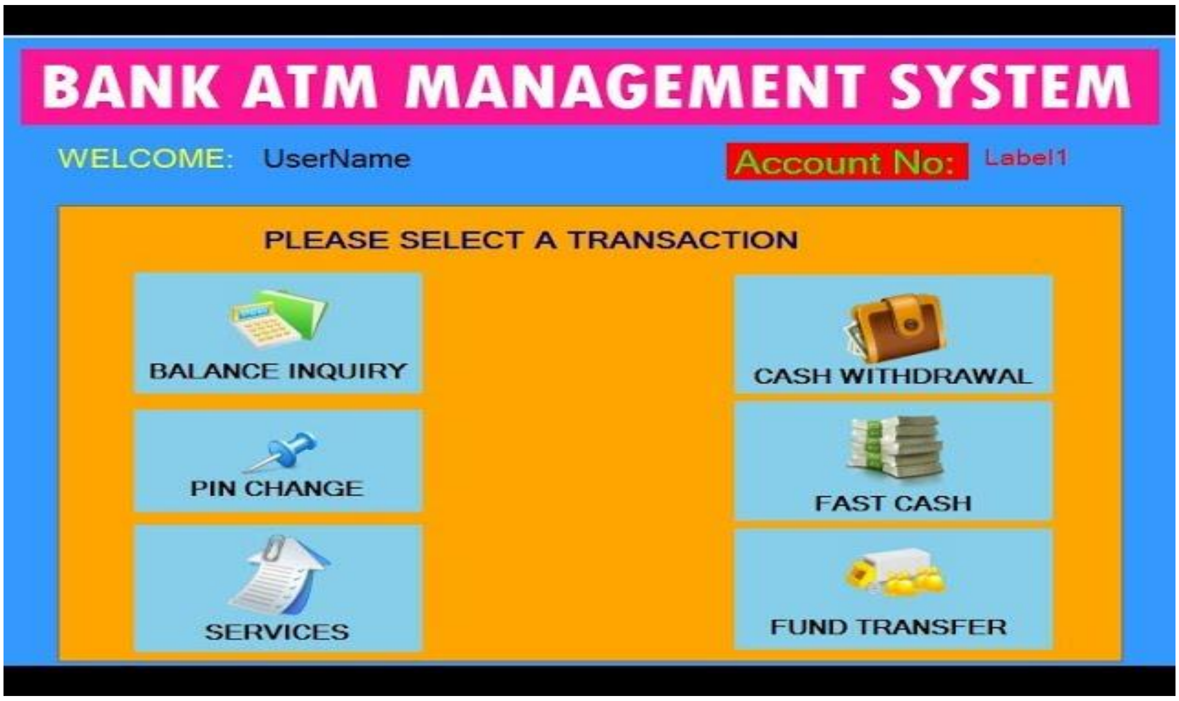
3. Service Layer: Business logic implemented as Java services to validate and process transactions, account creation, etc.

4. DAO Layer: Data Access Objects (DAOs) that use JDBC for executing SQL queries.

5. Database: Relational database (e.g., MySQL, PostgreSQL) to store customer details, transactions, and account data.

6. Connection Pool: Managed JDBC connections for efficient and scalable database interactions.

7. Security Layer: Authentication and authorization mechanisms integrated with JDBC queries for secure access.



2.3 SYSTEM REQUIREMENTS

A Bank Management System is a software application designed to handle various banking operations such as account management, transaction processing, loan management, and customer service. Below are the system requirements categorized into functional and non-functional requirements, along with hardware and software specifications.

1. Functional Requirements

1. User Management:

Secure registration and login for customers and bank staff.

Role-based access control for customers, tellers, and administrators.

2. Account Management:

Open, update, and close bank accounts.

View account details such as balance, account type, and transaction history.

3. Transaction Processing:

Handle deposits, withdrawals, and fund transfers.

Generate transaction receipts and send notifications.

4. Loan Management:

Process loan applications, approvals, and disbursements.

Manage repayment schedules and interest calculations.

5. Customer Support:

Enable customers to raise queries or complaints.

Provide a chat or ticketing system for resolution tracking.

6. Report Generation:

Generate account statements, transaction summaries, and financial reports.

Audit trails for compliance purposes.

7. Security Features:

Multi-factor authentication (MFA) for secure logins.

Encryption of sensitive data and secure data storage.

2. Non-Functional Requirements

1. Performance:

The system should handle high transaction volumes with minimal latency.

Ensure a fast response time for user interactions.

2. Scalability:

Support an increasing number of users and transactions as the the bank grows.

3. Availability:

99.9% uptime to ensure uninterrupted banking services.

4. Security:

Compliance with regulatory standards like PCI DSS for financial data protection.

Protection against cyber threats such as unauthorized access, data breaches, and DDoS attacks.

5. Usability:

User-friendly interfaces for customers and bank staff.

Accessible design for all users, including those with disabilities.

3. Hardware Requirements

1. Server-Side Hardware:

Processor: Quad-core or higher (e.g., Intel Xeon or AMD Ryzen).

Memory: Minimum 16 GB RAM.

Storage: 1 TB SSD for database and backups.

Network: High-speed internet with redundancy (e.g., 1 Gbps).

2. Client-Side Hardware:

Processor: Dual-core or higher.

Memory: Minimum 4 GB RAM.

Storage: 256 GB HDD/SSD.

Display: 1366x768 resolution or higher.

4. Software Requirements

1. Server-Side Software

Operating System: Linux (Ubuntu/CentOS) or Windows Server.

Web Server: Apache or Nginx.

Database: MySQL, PostgreSQL, or Oracle Database.

Programming Language: Java (Spring Boot), Python (Django/Flask), or .NET.

2. Client-Side Software:

Operating System: Windows, macOS, or Linux.

Web Browser: Google Chrome, Firefox, or Microsoft Edge (latest versions).

3. Security Tools:

Firewall and intrusion detection systems.

SSL/TLS for secure communication.

4. Other Tools:

Reporting tools like Jasper Reports or Tableau for generating reports.

APIs for integrating third-party services like payment gateways and credit bureaus.

5. Network Requirements

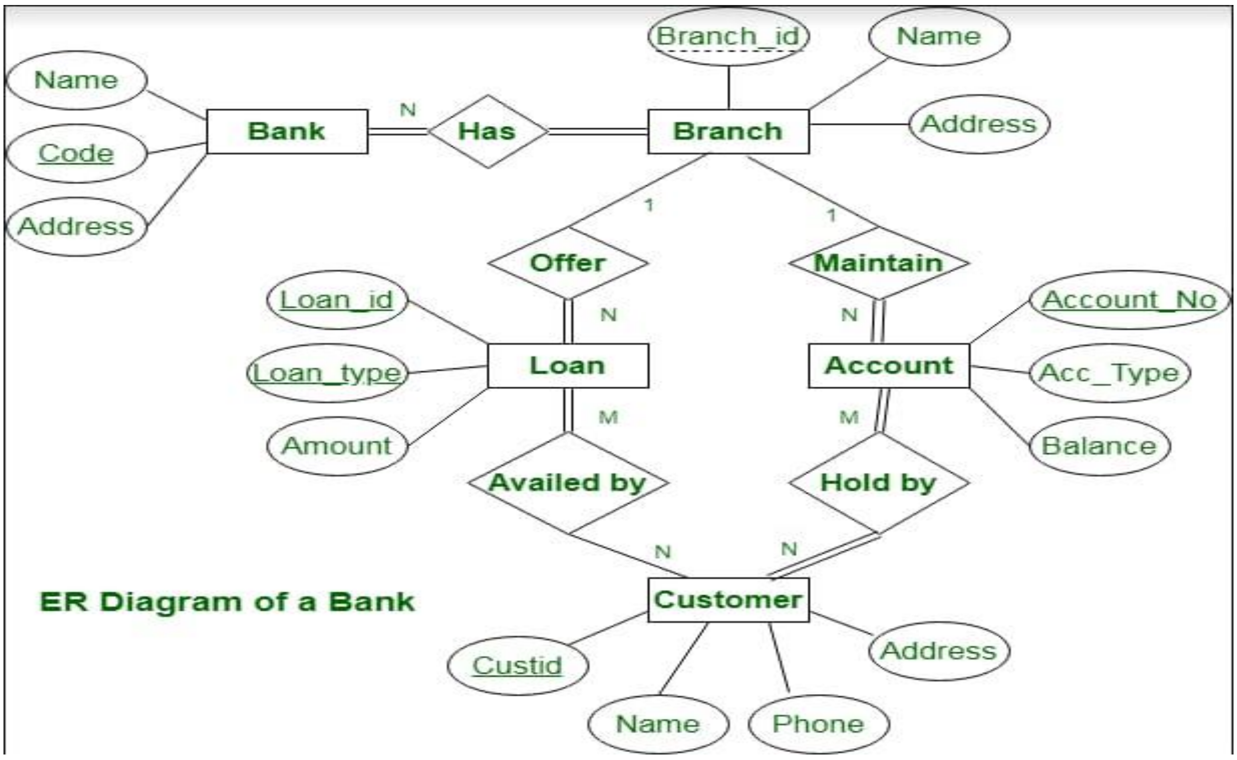
LAN/WAN: For internal bank operations.

Internet: For online banking services and remote access.

VPN: Secure connections for remote staff or branch offices.

By meeting these requirements, the Bank Management System will provide a secure, efficient, and scalable solution for managing banking operations and enhancing customer experience.

E-R DIAGRAMS


Name
Code
Address
Bank
N
Has
Branch_id
Name
Branch
Address
1
Offer
1
Maintain
Loan_id
Loan_type
Amount
N
Loan
M
Availed by
N
Account
M
Account_No
Acc_Type
Balance
Hold by
N
N
ER Diagram of a Bank
Customer
Custid
Name
Phone
Address

CHAPTER - 3
PROJECT DESCRIPTION

3.1 INTRODUCTION

The Bank Management System using JDBC is a Java-based application that enables efficient management of banking operations. It connects to a MySQL database to perform tasks like account creation, deposits, withdrawals, balance inquiries, and account deletion. The system ensures data integrity and security through transaction management and validation. JDBC's robust connectivity facilitates real-time database interactions. The project is designed to streamline banking processes, enhance user experience, and maintain accurate records. Future enhancements include GUI integration and additional banking features like loan processing.

3.2 METHODOLOGIES

1. Requirement Analysis: Identify the database structure, including tables for users, accounts, transactions, and loans.

2. Design and Modeling: Use ER diagrams to design the database schema and define JDBC connections for CRUD operations.

3. Modular Development: Divide the system into modules like authentication, account management, and transaction handling.

4. Connection Management: Implement efficient database connections using connection pooling to enhance performance.

5. SQL Query Optimization: Write and test optimized SQL queries for faster data retrieval and manipulation.

6. Testing and Debugging: Validate database interactions with test cases for queries and exception handling.

7. Deployment and Maintenance: Ensure the JDBC driver is configured correctly and maintain database integrity over time.

3.3 MODULE DESCRIPTION

Bank Management System (JDBC-based):

1. User Authentication Module: Handles user login and registration by validating credentials stored in the database.

2. Account Management Module: Manages account creation, updates, and closures, retrieving and modifying account details in the database.

3. Transaction Module: Facilitates deposits, withdrawals, and fund transfers while ensuring real-time updates to account balances.

4. Loan Management Module: Processes loan applications, interest calculations, and repayment tracking using database records.

5. Customer Management Module: Maintains customer profiles, contact information, and relationship history in the database.

6. Report Generation Module: Generates statements and reports like transaction history and account summaries using SQL queries.

CHAPTER - 4
RESULTS AND DISCUSSION

Results

The Bank Management System (BMS) implemented using JDBC allows seamless interaction between the user interface and the underlying database. The key results of the project include:

1. Database Connectivity:

Successfully established a connection to the MySQL database using the JDBC driver.

Enabled efficient communication between the Java application and the database.

2. Functionalities Implemented:

Account Creation: Users can create new accounts with details such as name, account type, and balance.

Transaction Management:

Deposit and withdrawal operations are recorded with balance updates.

Ensures no overdraft by validating balances during withdrawals.

Account Details: Allows users to view account details like balance and account holder information.

Account Deletion: Enables account deletion when requested by users.

Search Operations: Supports searching for accounts based on user-provided criteria (e.g., account number, name).

Transaction Logs: Maintains a log of all transactions for auditing and review purposes.

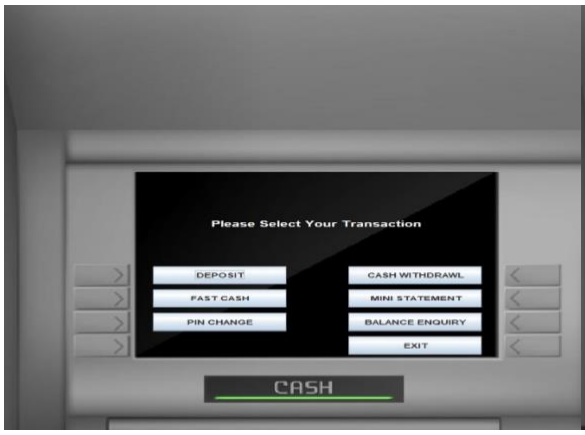
3. Performance:

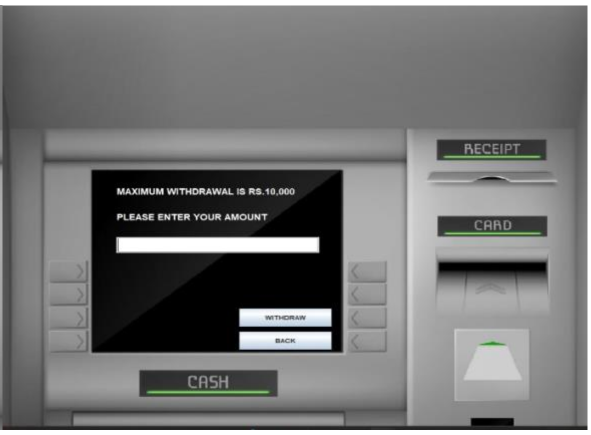
Database queries such as SELECT, INSERT, UPDATE, and DELETE were executed efficiently.

Response times were observed to be within acceptable limits, ensuring a smooth user experience.

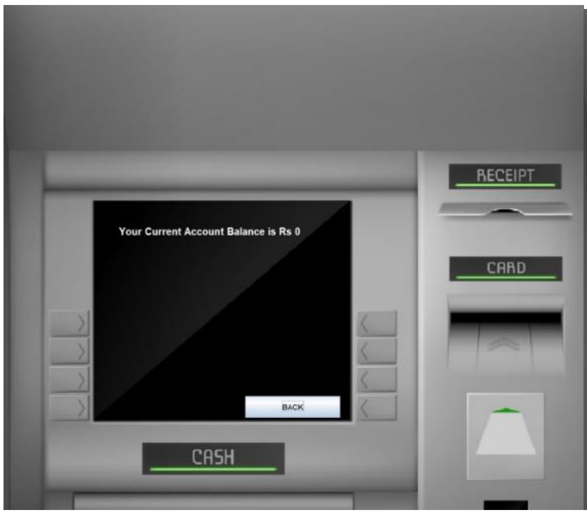
4. Scalability:

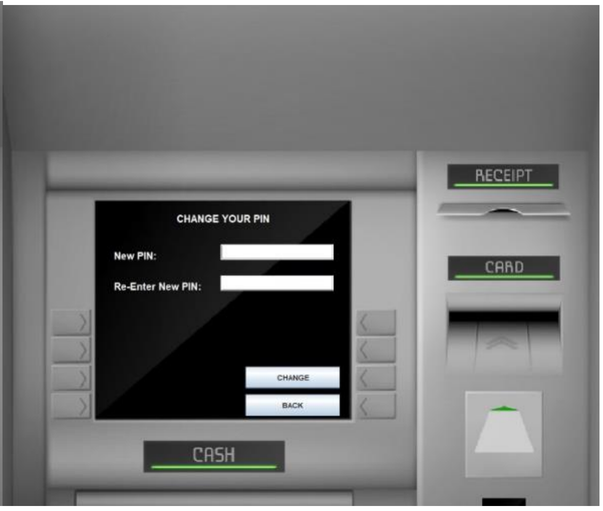
The system supports multiple users and can handle concurrent operations due to efficient database indexing and connection pooling.


Please Select Your Transaction


MAXIMUM WITHDRAWAL IS RS.10,000

PLEASE ENTER YOUR AMOUNT


Your Current Account Balance is Rs 0


CHANGE YOUR PIN

New PIN:

Re-Enter New PIN:

Discussion

1. Challenges Faced:

Database Connectivity Issues: Initial connection errors were resolved by properly configuring the JDBC URL and ensuring the MySQL service was running.

Data Validation: Ensuring accurate input validation (e.g., non-negative balances, correct account numbers) required additional logic.

Concurrency Management: Handling multiple simultaneous operations required implementing database locks to avoid data inconsistency.

2. Strengths of the System:

The system's modular design ensures maintainability and scalability.

JDBC provides a straightforward method for database interaction, reducing complexity.

Transactions are handled securely, ensuring atomicity and consistency.

3. Areas for Improvement:

Error Handling: Adding detailed error messages to guide users during failures.

Security Enhancements: Implementing encryption for sensitive data like passwords and secure database connections.

User Interface: The current console-based interface can be improved by integrating a graphical user interface (GUI) or a web-based frontend.

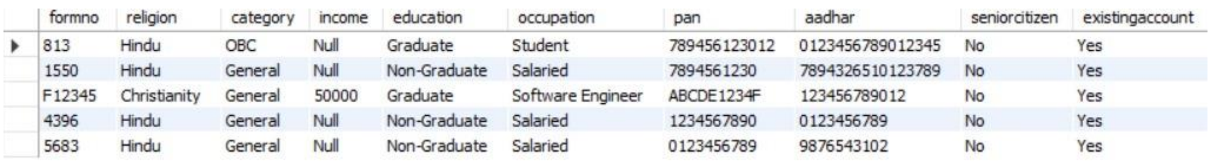
4. Future Scope:

Implement advanced features such as loan processing, credit card management, and investment tracking.

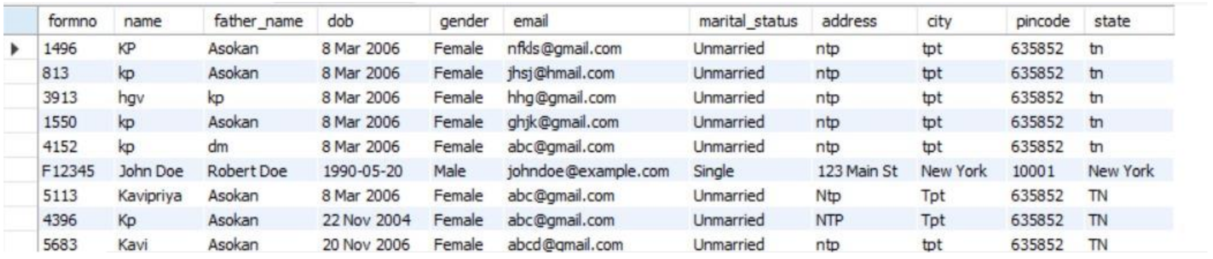
Migrate the database to a distributed system for improved performance and reliability.

Integrate machine learning for customer insights and fraud detection.

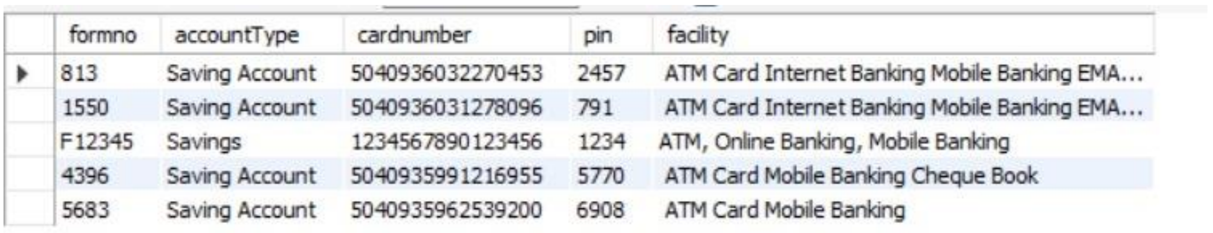
TABLE:



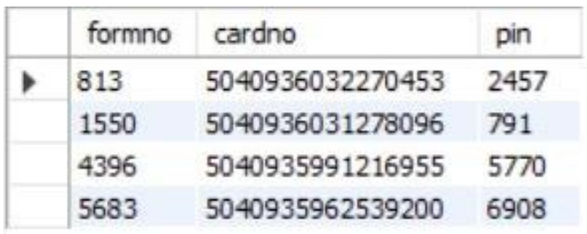
formno	religion	category	income	education	occupation	pan	aadhar	seniorcitizen	existingaccount
813	Hindu	OBC	Null	Graduate	Student	789456123012	0123456789012345	No	Yes
1550	Hindu	General	Null	Non-Graduate	Salaried	7894561230	7894326510123789	No	Yes
F12345	Christianity	General	50000	Graduate	Software Engineer	ABCDE1234F	123456789012	No	Yes
4396	Hindu	General	Null	Non-Graduate	Salaried	1234567890	0123456789	No	Yes
5683	Hindu	General	Null	Non-Graduate	Salaried	0123456789	9876543102	No	Yes



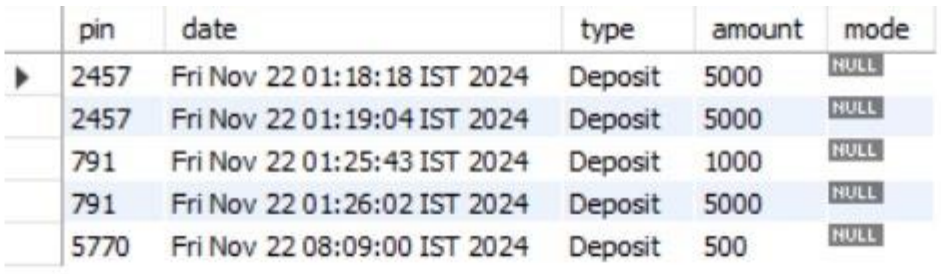
formno	name	father_name	dob	gender	email	marital_status	address	city	pincode	state
1496	KP	Asokan	8 Mar 2006	Female	nfkls@gmail.com	Unmarried	ntp	tpt	635852	tn
813	kp	Asokan	8 Mar 2006	Female	jhsj@hmail.com	Unmarried	ntp	tpt	635852	tn
3913	hgv	kp	8 Mar 2006	Female	hhg@gmail.com	Unmarried	ntp	tpt	635852	tn
1550	kp	Asokan	8 Mar 2006	Female	ghjk@gmail.com	Unmarried	ntp	tpt	635852	tn
4152	kp	dm	8 Mar 2006	Female	abc@gmail.com	Unmarried	ntp	tpt	635852	tn
F12345	John Doe	Robert Doe	1990-05-20	Male	johndoe@example.com	Single	123 Main St	New York	10001	New York
5113	Kavipriya	Asokan	8 Mar 2006	Female	abc@gmail.com	Unmarried	Ntp	Tpt	635852	TN
4396	Kp	Asokan	22 Nov 2004	Female	abc@gmail.com	Unmarried	NTP	Tpt	635852	TN
5683	Kavi	Asokan	20 Nov 2006	Female	abcd@gmail.com	Unmarried	ntp	tpt	635852	TN



formno	accountType	cardnumber	pin	facility
813	Saving Account	5040936032270453	2457	ATM Card Internet Banking Mobile Banking EMA...
1550	Saving Account	5040936031278096	791	ATM Card Internet Banking Mobile Banking EMA...
F12345	Savings	1234567890123456	1234	ATM, Online Banking, Mobile Banking
4396	Saving Account	5040935991216955	5770	ATM Card Mobile Banking Cheque Book
5683	Saving Account	5040935962539200	6908	ATM Card Mobile Banking



formno	cardno	pin
813	5040936032270453	2457
1550	5040936031278096	791
4396	5040935991216955	5770
5683	5040935962539200	6908



pin	date	type	amount	mode
2457	Fri Nov 22 01:18:18 IST 2024	Deposit	5000	NULL
2457	Fri Nov 22 01:19:04 IST 2024	Deposit	5000	NULL
791	Fri Nov 22 01:25:43 IST 2024	Deposit	1000	NULL
791	Fri Nov 22 01:26:02 IST 2024	Deposit	5000	NULL
5770	Fri Nov 22 08:09:00 IST 2024	Deposit	500	NULL

Login Page:

```
package ASimulatorSystem;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class Login extends JFrame implements ActionListener{
    JLabel l1,l2,l3;
    JTextField tf1;
    JPasswordField pf2;
    JButton b1,b2,b3;

    Login(){
        setTitle("AUTOMATED TELLER MACHINE");

        ImageIcon i1 = new
ImageIcon(ClassLoader.getSystemResource("ASimulatorSystem/icons/logo.jpg"));
        Image i2 = i1.getImage().getScaledInstance(100, 100, Image.SCALE_DEFAULT);
        ImageIcon i3 = new ImageIcon(i2);
        JLabel l11 = new JLabel(i3);
        l11.setBounds(70, 10, 100, 100);
        add(l11);

        l1 = new JLabel("WELCOME TO ATM");
```

```
l1.setFont(new Font("Osward", Font.BOLD, 38));
l1.setBounds(200,40,450,40);
add(l1);

l2 = new JLabel("Card No:");
l2.setFont(new Font("Raleway", Font.BOLD, 28));
l2.setBounds(125,150,375,30);
add(l2);

tf1 = new JTextField(15);
tf1.setBounds(300,150,230,30);
tf1.setFont(new Font("Arial", Font.BOLD, 14));
add(tf1);

l3 = new JLabel("PIN:");
l3.setFont(new Font("Raleway", Font.BOLD, 28));
l3.setBounds(125,220,375,30);
add(l3);

pf2 = new JPasswordField(15);
pf2.setFont(new Font("Arial", Font.BOLD, 14));
pf2.setBounds(300,220,230,30);
add(pf2);

b1 = new JButton("SIGN IN");
b1.setBackground(Color.BLACK);
b1.setForeground(Color.WHITE);

b2 = new JButton("CLEAR");
b2.setBackground(Color.BLACK);
b2.setForeground(Color.WHITE);
```

```
        b3 = new JButton("SIGN UP");
        b3.setBackground(Color.BLACK);
        b3.setForeground(Color.WHITE);

        setLayout(null);

        b1.setFont(new Font("Arial", Font.BOLD, 14));
        b1.setBounds(300,300,100,30);
        add(b1);

        b2.setFont(new Font("Arial", Font.BOLD, 14));
        b2.setBounds(430,300,100,30);
        add(b2);

        b3.setFont(new Font("Arial", Font.BOLD, 14));
        b3.setBounds(300,350,230,30);
        add(b3);

        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);

        getContentPane().setBackground(Color.WHITE);

        setSize(800,480);
        setLocation(550,200);
        setVisible(true);

    }
    public void actionPerformed(ActionEvent ae){
        try{
            if(ae.getSource()==b1){
                Conn c1 = new Conn();
                String cardno  = tf1.getText();
                String pin  = pf2.getText();
                String q   = "select * from login where cardno = '"+cardno+"' and pin =
'"+pin+"'";

                ResultSet rs = c1.s.executeQuery(q);
                if(rs.next()){
                    setVisible(false);
                    new Transactions(pin).setVisible(true);
                }else{
                    JOptionPane.showMessageDialog(null, "Incorrect Card Number or PIN");
                }
            }else if(ae.getSource()==b2){
                tf1.setText("");
                pf2.setText("");
            }else if(ae.getSource()==b3){
                setVisible(false);
                new Signup().setVisible(true);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
    public static void main(String[] args){
        new Login().setVisible(true);
    }
}
```

CONCLUSION

The Bank Management System (BMS) using JDBC provides a reliable and efficient solution for automating core banking operations such as account management, transaction handling, and customer data management. By leveraging Java and a relational database, the system ensures real-time processing and secure handling of sensitive financial data. Its modular architecture enhances usability, scalability, and maintainability, making it suitable for small to mid-sized banking institutions.

This project addresses the limitations of traditional systems by reducing human errors, streamlining processes, and improving overall operational efficiency. With features like role-based access control, secure transactions, and intuitive interfaces, the system offers a modern approach to banking. Furthermore, it lays the groundwork for future enhancements, including mobile integration and predictive analytics, ensuring the system remains relevant in a rapidly evolving financial landscape.