# INTRODUCTION TO DATABASES (CSE 3151)

## TERM PROJECT REPORT

## (March'2023-July'2023)

## On

# Banking Management System

*Submitted By*


**Ishita Jena**

**Registration No.: 2141010031**

**B.Tech. $6^{th}$ Semester CSE(M)**




**Department of Computer Science and Engineering**

**Institute of Technical Education and Research**

**Siksha 'O' Anusandhan Deemed To Be University**

**Bhubaneswar, Odisha-751030**

# Abstract

The Banking Management System (BMS) project aims to integrate theoretical concepts with practical laboratory exercises by developing a miniature banking system using Java, Oracle, and JDBC. The system is designed to manage customer and account data through a Java-based frontend that interacts with an Oracle database backend. The application provides a menu-driven interface allowing users to perform various operations such as viewing customer records, adding, deleting, and updating customer information, as well as managing account details and transactions.

Key functionalities of the BMS include:

- **Displaying Customer Records:** Users can view all customer details in a structured format.
- **Adding Customer Records:** New customer information can be entered and stored in the database.
- **Deleting Customer Records:** Specific customer data can be removed from the database using a customer number.
- **Updating Customer Information:** Users can update customer details such as name, phone number, and city.
- **Updating Customer Information:** Users can update customer details such as name, phone number, and city.
- **Displaying Account Details:** Detailed account information for a specific customer can be retrieved.
- **Displaying Loan Details:** Loan information for a specific customer can be accessed.
- **Depositing Money:** Funds can be deposited into a customer's account.
- **Withdrawing Money:** Customers can withdraw funds from their accounts.
- **Exiting the Program:** The application can be terminated through an exit option.

The program utilizes switch-case constructs for menu navigation and performs database operations based on user input. Error handling is incorporated to manage exceptions, ensuring the system's robustness. The output is formatted for clarity and ease of understanding, enhancing the user experience. The BMS project demonstrates the effective integration of Java programming with database management, providing a practical and educational tool for managing banking operations.

# Contents

# 1 Introduction

In today's digital age, efficient management of banking operations is crucial for financial institutions to deliver high-quality services to their customers. The Banking Management System (BMS) project aims to provide a streamlined solution by integrating fundamental concepts of Java programming with database management using Oracle and JDBC. This project serves as a practical application of theoretical knowledge acquired in classroom settings, supplemented by hands-on laboratory exercises.

The BMS is designed as a menu-driven program that enables users to perform various banking operations through a simple and intuitive Java-based interface. By connecting to an Oracle database via JDBC, the system allows for seamless interaction between the frontend and backend, facilitating essential banking functions such as managing customer records, handling account details, and processing transactions.

The banking sector is a critical component of the economy, and its operations must be managed with precision and efficiency. The BMS project addresses these needs by providing a comprehensive set of features for customer and account management. Users can add, delete, and update customer records, view detailed account and loan information, and perform financial transactions such as deposits and withdrawals. Each of these operations is crucial for maintaining the integrity and functionality of a banking system, ensuring that customers' needs are met promptly and accurately. The growing complexity of financial services necessitates robust and scalable software systems. The BMS project not only meets these requirements but also demonstrates the importance of integrating various technological skills.

Furthermore, the BMS project emphasizes the significance of database connectivity and data integrity. By leveraging JDBC to interact with an Oracle database, the system ensures that all operations are performed efficiently and accurately, maintaining the consistency and reliability of the stored data. This integration of Java and Oracle showcases the practical applications of programming and database management skills in developing functional and effective banking systems. The project also highlights the importance of error handling and user-friendly interfaces, ensuring that the system is robust, reliable, and easy to use.

In conclusion, the Banking Management System project is an exemplary demonstration of how Java and Oracle can be combined to build effective management systems. It serves as an educational tool that bridges the gap between theoretical knowledge and practical implementation.

# 2 System Overview

## 2.1 Architecture

The Banking Management System (BMS) is structured as a multi-tier application that integrates a Java-based frontend with an Oracle database backend using JDBC (Java Database Connectivity). The architecture of the system can be broken down into the following layers:

1. **Presentation Layer:**
    - **Java Console Application:** The user interacts with the system through a console-based menu-driven interface implemented in Java. This layer handles user input, processes it, and displays output in a structured format. It ensures that the user interface is intuitive and straightforward, providing clear prompts and feedback to guide the user through various operations.
2. **Business Logic Layer:**
    - **Java Application Logic:** This layer contains the core functionality of the BMS. It processes user requests, performs validation, and manages the flow of data between the presentation layer and the database layer. Utilizing switch-case constructs, it navigates different operations based on user input. This layer encapsulates the business rules and logic, ensuring that all operations are executed correctly and consistently. It also includes error handling mechanisms to manage exceptions and maintain system reliability.
3. **Data Access Layer:**
    - **JDBC:** Java Database Connectivity (JDBC) is used to establish a connection between the Java application and the Oracle database. JDBC handles the execution of SQL queries and updates, and the retrieval of results from the database. This layer abstracts the complexities of database interactions, providing a simple API for the business logic layer to perform CRUD (Create, Read, Update, Delete) operations on the database.
4. **Database Layer:**
    - **Oracle Database:** This layer stores all the persistent data for the BMS, including customer records, account details, and transaction information. The database schema includes tables for customers, accounts, and loans, each with appropriate relationships and constraints. This layer ensures data integrity, consistency, and security. The Oracle database is designed to efficiently handle large volumes of data and complex queries, providing a robust foundation for

the BMS.

## 2.2 Functionality

The BMS provides the following functionalities to the user:

1. **Show Customer Records:**
    - This functionality retrieves and displays all customer details from the database in a predefined format.
    - It provides a comprehensive view of all customers, including their customer number, name, phone number, and city.
    - This feature is crucial for quickly accessing customer information and verifying data integrity.
2. **Add Customer Record:**
    - Prompts the user for customer information (customer number, name, phone number, city).
    - Inserts the new customer record into the database.
    - Displays updated customer records, ensures that new customers can be added efficiently and their information is stored accurately.
3. **Delete Customer Record:**
    - Prompts the user for a customer number.
    - Deletes the corresponding customer record from the database.
    - Displays updated customer records, essentially maintaining an up-to-date customer database and removing obsolete or incorrect records.
4. **Update Customer Information:**
    - Prompts the user for a customer number.
    - Provides options to update specific customer information (name, phone number, city).
    - Updates the customer record in the database based on the chosen option.
    - Displays updated customer records, ensuring that any changes to customer information are accurately reflected in the database
5. **Show Account Details of a Customer:**
    - Prompts the user for a customer number.

- The system retrieves and displays detailed account information for the specified customer, including account number, type, balance, branch code, branch name, and branch city.

6. **Show Loan Details of a Customer:**
    - Prompts the user for a customer number.
    - The system retrieves and displays detailed loan information for the specified customer, including loan number, loan amount, branch code, branch name, and branch city.
7. **Deposit Money to an Account:**
    - Prompts the user for an account number and the deposit amount.
    - Updates the account balance in the database.
    - Displays the updated account details ensuring that deposits are processed accurately and account balances are updated promptly.
8. **Withdraw Money from an Account:**
    - Prompts the user for an account number and the withdrawal amount.
    - Updates the account balance in the database, ensuring sufficient funds are available.
    - Displays the updated account details ensuring that withdrawals are processed accurately and account balances are updated promptly.
9. **Exit the Program:**
    - Terminates the application gracefully.
    - The system ensures that all resources are released properly and any pending operations are completed before exiting. T
    - his functionality ensures a smooth and orderly shutdown of the application.

Each operation is implemented with appropriate error handling to manage exceptions and ensure the system's reliability. The user interface presents clear and formatted messages, making the interaction intuitive and straightforward. This structured approach not only facilitates efficient banking management but also provides a practical demonstration of integrating Java applications with database systems. By providing a comprehensive set of features and ensuring robust error handling and user-friendly interfaces, the BMS project effectively demonstrates the application of software development principles in the context of banking operations.

# 3 Implementation

myjdbcproject.java

```
package JDBC_Project;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.*;

public class myjdbcproject {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver
            ");
            Connection con =
               DriverManager.getConnection("jdbc:oracle:
            thin:@localhost:1521:xe", "system",
               "ishitajena");
            System.out.println("Connection is
               established");
            Statement stmt = con.createStatement();
            BufferedReader br = new BufferedReader(new
               InputStreamReader(System.in));
            int ch = 0;
            do {
                System.out.println("\n\n***** Banking
                   Management System *****");
                System.out.println("1. Show Customer
                   Records");
                System.out.println("2. Add Customer
                   Record");
                System.out.println("3. Delete Customer
                   Record");
                System.out.println("4. Update Customer
                   Record for any attribute except
                   Customer Number");
                System.out.println("5. Show Account
                   Details of a Customer");
```

myjdbcproject.java

```
System.out.println("6. Show Loan Details
   of a Customer");
System.out.println("7. Deposit Money to
   an Account");
System.out.println("8. Withdraw Money
   from an Account");
System.out.println("9. Exit the
   Program");
System.out.println("Enter your choice
   (1-9):");
ch = Integer.parseInt(br.readLine());
switch (ch) {
    case 1:
        // Display customer records
        String sqlstr = "select * from
           CUSTOMER";
        ResultSet rs =
           stmt.executeQuery(sqlstr);
        int a = 0;
        while (rs.next()) {
            System.out.print('\n' +
               rs.getString("CUST_NO") +
               '\t');
            System.out.print('\t' +
               rs.getString("NAME"));
            System.out.print('\t' +
               rs.getString("PHONE_NO"));
            System.out.print('\t' +
               rs.getString("CITY"));
            a++;
        }
        System.out.println();
        System.out.println("The number
           of rows selected is " + a);
        break;

    case 2:
```

myjdbcproject.java

```
                    // Add customer record
                    System.out.println("Enter
                       cust_no: ");
                    String cust_no = br.readLine();
                    System.out.println("Enter the
                       name: ");
                    String name = br.readLine();
                    System.out.println("Enter the
                       phone: ");
                    long phone =
                       Long.parseLong(br.readLine());
                    System.out.println("Enter the
                       city: ");
                    String city = br.readLine();
                    String insertstmt = "insert into
                       CUSTOMER values('" +
                             cust_no + "','" + name +
                                "','" + phone + "','"
                                + city + "')";
                    int n =
                       stmt.executeUpdate(insertstmt);
                    System.out.println(n + " Row
                       Inserted");
                    break;

                case 3:
                    // Delete customer record
                    System.out.println("Enter a
                       cust_no for deletion: ");
                    String cust = br.readLine();
                    String deletestmt = "delete from
                       CUSTOMER where CUST_NO='" +
                       cust + "'";
                    n =
                       stmt.executeUpdate(deletestmt);
                    System.out.println(n + " Row
                       deleted");
```

myjdbcproject.java

```
                    break;

                case 4:
                    // Update customer record
                    System.out.println("Enter 1: For
                       Name 2: For Phone no 3: For
                       City to update:");
                    int c1 =
                       Integer.parseInt(br.readLine());
                    System.out.println("Enter Cust
                       No");
                    cust_no = br.readLine();
                    switch (c1) {
                        case 1:
                            System.out.println("Enter
                               updated name");
                            String updatedName =
                               br.readLine();
                            String queryupdateName =
                               "Update CUSTOMER set
                               NAME = '" +
                               updatedName + "' where
                               CUST_NO = '" + cust_no
                               + "'";
                            n = stmt.executeUpdate(
                            queryupdateName);
                            System.out.println(n + "
                               Updated Successfully");
                            break;

                        case 2:
                            System.out.println("Enter
                               updated Phone");
                            long updatedPhone =
                               Long.parseLong(
                            br.readLine());
                            String queryupdatePhone =
```

myjdbcproject.java

```
                                "Update CUSTOMER set
                                   PHONE_NO = '" +
                                   updatedPhone + "'
                                   where CUST_NO = '" +
                                   cust_no + "'";
                                n = stmt.executeUpdate(
                                queryupdatePhone);
                                System.out.println(n + "
                                   Updated Successfully");
                                break;
                            case 3:
                                System.out.println("Enter
                                   updated City");
                                String updatedCity =
                                   br.readLine();
                                String queryupdateCity =
                                   "Update CUSTOMER set
                                   CITY = '" +
                                   updatedCity + "' where
                                   CUST_NO = '" + cust_no
                                   + "'";
                                n = stmt.executeUpdate(
                                queryupdateCity);
                                System.out.println(n + "
                                   Updated Successfully");
                                break;
                            default:
                                System.out.println("Invalid
                                   option.");
                                break;
                        }
                        break;

                    case 5:
                        // Show Account Details of a
                           Customer
                        System.out.println("Enter Cust
```

myjdbcproject.java

```
No");
cust_no = br.readLine();
String queryAccDetails = "SELECT
   A.ACCOUNT_NO, A.TYPE,
   A.BALANCE, B.BRANCH_CODE,
   B.BRANCH_NAME, B.BRANCH_CITY "
   +
         "FROM ACCOUNT A, BRANCH
            B, CUSTOMER C,
            DEPOSITOR D " +
         "WHERE D.ACCOUNT_NO =
            A.ACCOUNT_NO AND
            A.BRANCH_CODE =
            B.BRANCH_CODE " +
         "AND C.CUST_NO =
            D.CUST_NO AND
            C.CUST_NO = '" +
            cust_no + "'";
rs = stmt.executeQuery(
queryAccDetails);
System.out.println("Acc.
   No.\tType\tBalance\tBranch
   Code\tBranch Name\tBranch
   City");
System.out.println("---------------
-------------------------------");
while (rs.next()) {
    String account_no =
       rs.getString("ACCOUNT_NO");
    String type =
       rs.getString("TYPE");
    long balance =
       rs.getLong("BALANCE");
    String branch_code =
       rs.getString("BRANCH_CODE");
    String branch_name =
       rs.getString("BRANCH_NAME");
```

myjdbcproject.java

```
                        String branch_city =
                           rs.getString("BRANCH_CITY");
                        System.out.println(account_no
                           + "\t\t" + type + "\t" +
                                 balance + "\t" +
                                    branch_code +
                                    "\t\t" +
                                    branch_name + "\t"
                                    + branch_city);
                    }
                    break;

                case 6:
                    // Show Loan Details of a
                       Customer
                    System.out.println("Enter Cust
                       No");
                    cust_no = br.readLine();
                    String queryLoanDetails =
                       "SELECT C.*, L.LOAN_NO,
                       L.AMOUNT, B.* FROM LOAN L,
                       CUSTOMER C, BRANCH B " +
                             "WHERE L.BRANCH_CODE =
                                B.BRANCH_CODE AND
                                C.CUST_NO = L.CUST_NO
                                AND C.CUST_NO = '" +
                                cust_no + "'";
                    rs = stmt.executeQuery(
                    queryLoanDetails);
                    System.out.println("Cust
                       No.\tName\tPhone\t\tCity\tLoan
                       No.\tAmount\tBranch
                       Code\tBranch Name\tCity");
                    System.out.println("-------------
                    -------------------------------");
                    while (rs.next()) {
```

myjdbcproject.java

```
                        String ccust =
                           rs.getString("CUST_NO");
                        String cname =
                           rs.getString("NAME");
                        long cphone =
                           rs.getLong("PHONE_NO");
                        String ccity =
                           rs.getString("CITY");
                        String branch_code =
                           rs.getString("BRANCH_CODE");
                        String branch_name =
                           rs.getString("BRANCH_NAME");
                        String branch_city =
                           rs.getString("BRANCH_CITY");
                        String loan_no =
                           rs.getString("LOAN_NO");
                        long amount =
                           rs.getLong("AMOUNT");
                        System.out.println(ccust +
                           "\t" + cname + "\t" +
                                 cphone + "\t" +
                                    ccity + "\t" +
                                    loan_no + "\t\t" +
                                    amount + "\t" +
                                    branch_code +
                                    "\t\t" +
                                 branch_name + "\t" +
                                    branch_city);
                    }
                    break;

                case 7:
                    // Deposit money
                    System.out.println("Enter
                       Account No");
                    String acc_no = br.readLine();
                    System.out.println("Enter the
```

myjdbcproject.java

```
                        amount to deposit");
                        long amt =
                           Long.parseLong(br.readLine());
                        String queryAddMoney = "Update
                           ACCOUNT set BALANCE = BALANCE
                           + " + amt + " where ACCOUNT_NO
                           = '" + acc_no + "'";
                        n = stmt.executeUpdate(
                        queryAddMoney);
                        System.out.println(n + " Balance
                           Updated");
                        break;

                    case 8:
                        // Withdraw money
                        System.out.println("Enter
                           Account No");
                        acc_no = br.readLine();
                        System.out.println("Enter the
                           amount to withdraw");
                        amt =
                           Long.parseLong(br.readLine());
                        String queryCheckBalance =
                           "Select BALANCE from ACCOUNT
                           where ACCOUNT_NO = '" + acc_no
                           + "'";
                        rs = stmt.executeQuery(
                        queryCheckBalance);
                        if (rs.next()) {
                            long bal =
                               rs.getLong("BALANCE");
                            if (amt <= bal) {
                                String querySubMoney =
                                   "Update ACCOUNT set
                                   BALANCE = BALANCE - "
                                   + amt + " where
                                   ACCOUNT_NO = '" +
```

myjdbcproject.java

```
                                    acc_no + "'";
                                    n = stmt.executeUpdate(
                                    querySubMoney);
                                    System.out.println(n + "
                                       Balance Updated");
                                } else {
                                    System.out.println(
                                    "Insufficient balance");
                                }
                            }
                            break;

                        case 9:
                            // Exit the program
                            System.out.println("EXITING THE
                               PROGRAM...");
                            stmt.close();
                            con.close();
                            System.exit(0);
                            break;

                        default:
                            // Handle wrong choice of option
                            System.out.println("Enter from 1
                               to 9, nothing else");
                            break;
                    }
                } while (ch != 9);
            }

            catch (Exception e) {
                System.out.println(e);
            }
        }
    }
```

# 4 Testing

- **Displaying customer records:**

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 12:40:24 am) [pid: 6680]
Connection is established


***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
1
C0001           RAJ ANAND SINGH 9861258466      DELHI
C0002           ANKITA SINGH    9879958651      BANGALORE
C0003           SOUMYA JHA      9885623344      MUMBAI
C0004           ABHIJIT MISHRA  9455845425      MUMBAI
C0005           YASH SARAF      9665854585      KOLKATA
C0006           SWAROOP RAY     9437855466      CHENNAI
C0007           SURYA NARAYAN   9937955212      GURGAON
C0008           PRANAV PRAVEEN  9336652441      PUNE
C0009           STUTI MISRA     7870266534      DELHI
C0010           ASLESHA TIWARI  null    MUMBAI
10 rows selected
```

- **Adding customer record:**

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 1:50:09 am) [pid: 12552]
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
2
Enter cust_no:
C0011
Enter the name:
ANWESHA DAS
Enter the phone:
9999999999
Enter the city:
BHUBANESWAR
1 row inserted
```

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 1:50:09 am) [pid: 12552]
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
1
C0001           RAJ ANAND SINGH 9861258466      DELHI
C0002           ANKITA SINGH    9879958651      BANGALORE
C0003           SOUMYA JHA      9885623344      MUMBAI
C0004           ABHIJIT MISHRA  9455845425      MUMBAI
C0005           YASH SARAF      9665854585      KOLKATA
C0006           SWAROOP RAY     9437855466      CHENNAI
C0007           SURYA NARAYAN   9937955212      GURGAON
C0008           PRANAV PRAVEEN  9336652441      PUNE
C0009           STUTI MISRA     7870266534      DELHI
C0010           ASLESHA TIWARI  null    MUMBAI
C0013           ARJUN MISHRA    7777777777      BHUBANESWAR
C0011           ANWESHA DAS     9999999999      BHUBANESWAR
C0012           DORAEMON        9898989898      CUTTACK
13 rows selected
```

- **Deleting customer record:**

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 1:50
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
3
Enter a cust_no for deletion:
C0013
1 row deleted
```

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 1:50
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
3
Enter a cust_no for deletion:
C0016
0 row deleted
```

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 2:42
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
1

C0001           RAJ ANAND SINGH 9861258466      DELHI
C0002           ANKITA SINGH    9879958651      BANGALORE
C0003           SOUMYA JHA      9885623344      MUMBAI
C0004           ABHIJIT MISHRA  9455845425      MUMBAI
C0005           YASH SARAF      9665854585      KOLKATA
C0006           SWAROOP RAY     9437855466      CHENNAI
C0007           SURYA NARAYAN   9937955212      GURGAON
C0008           PRANAV PRAVEEN  9336652441      PUNE
C0009           STUTI MISRA     7870266534      DELHI
C0010           ASLESHA TIWARI  null    MUMBAI
C0011           ANWESHA DAS     9999999999      BHUBANESWAR
C0012           DORAEMON        9898989898      CUTTACK
12 rows selected
```

- **Updating customer record:**

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 1:50
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
4
Enter 1: For Name 2: For Phone no 3: For City to update:
1
Enter Cust No
C0011
Enter updated name
JOHN WICK
1 Updated Successfully
```

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 1:50
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
1

C0001           RAJ ANAND SINGH 9861258466      DELHI
C0002           ANKITA SINGH    9879958651      BANGALORE
C0003           SOUMYA JHA      9885623344      MUMBAI
C0004           ABHIJIT MISHRA  9455845425      MUMBAI
C0005           YASH SARAF      9665854585      KOLKATA
C0006           SWAROOP RAY     9437855466      CHENNAI
C0007           SURYA NARAYAN   9937955212      GURGAON
C0008           PRANAV PRAVEEN  9336652441      PUNE
C0009           STUTI MISRA     7870266534      DELHI
C0010           ASLESHA TIWARI  null    MUMBAI
C0011           JOHN WICK       1234567890      CONTINENTAL
C0012           DORAEMON        9898989898      CUTTACK
12 rows selected
```

- **Show account details of a customer:**

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 2:17:01 am) [pid:
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
5
Enter Cust No
C0003
Acc. No.        Type    Balance Branch Code     Branch Name     Branch City
--------------------------------------------------------------------------------
A0001           SB      209000  B003            JUHU BRANCH     MUMBAI
```

- **Show loan details of a customer:**

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 2:17:01 am) [pid: 17200]
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
6
Enter Cust No
C0005
Cust No.        Name    Phone           City    Loan No.        Amount  Branch Code     Branch Name     City
--------------------------------------------------------------------------------------------------------------------
C0005   YASH SARAF      9665854585      KOLKATA L0001           3000000 B006            SRIRAMPURAM BRANCH      CHENNAI
```

- **Deposit money to an account:**

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 2:17
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
7
Enter Account No
A0008
Enter the amount to deposit
800
1 Balance Updated
```

- **Withdraw money from an account:**

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 2:17
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
8
Enter Account No
A0008
Enter the amount to withdraw
800
1 Balance Updated
```

- **Entering the wrong choice:**

```
Problems  Javadoc  Declaration  Console ×
myjdbcproject [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (23-May-2024, 2:36
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
10
Please enter a number from 1-9
```

- **Exiting the program:**

```
***** Banking Management System *****
1. Show Customer Records
2. Add Customer Record
3. Delete Customer Record
4. Update Customer Record for any attribute except Customer Number
5. Show Account Details of a Customer
6. Show Loan Details of a Customer
7. Deposit Money to an Account
8. Withdraw Money from an Account
9. Exit the Program
Enter your choice (1-9):
9
EXITING THE PROGRAM...
```

# 5 Conclusion

The Banking Management System (BMS) project represents a significant achievement in integrating Java programming with Oracle database management via JDBC, resulting in a comprehensive, functional system for managing banking operations. This project serves as an effective bridge between theoretical knowledge and practical application, providing a robust platform for efficient banking management.

The BMS is meticulously designed to handle essential banking functions, such as managing customer records, processing transactions, and maintaining account details. Each functionality is implemented with precision, ensuring that operations such as adding, deleting, updating customer records, and handling deposits and withdrawals are performed efficiently and accurately. The use of a menu-driven interface enhances user interaction by making the system intuitive and user-friendly, thereby facilitating seamless navigation and operation.

By leveraging JDBC to connect the Java-based frontend with the Oracle backend, the system ensures robust database connectivity and data integrity. The architecture of the BMS is structured to promote modularity and scalability, allowing for easy maintenance and future enhancements. This layered architecture separates the presentation, business logic, data access, and database layers, ensuring a clean and organized codebase that can be easily understood and modified.

The project also emphasizes the importance of error handling and exception management, which are critical for maintaining system reliability and user trust. By incorporating comprehensive error-handling mechanisms, the BMS can gracefully manage unexpected situations, ensuring that the system remains stable and secure. This aspect of the project highlights the practical considerations necessary for developing resilient software applications.

Furthermore, the BMS project provides a deep understanding of the complexities involved in managing banking operations. It demonstrates how software systems can be designed to address the specific needs of the banking sector, such as data security, transaction integrity, and efficient customer service. The project showcases how modern technologies can be applied to create efficient, reliable, and scalable solutions that meet the high standards required in the financial industry.

In conclusion, the Banking Management System (BMS) project demonstrates the effective integration of Java and Oracle to create a robust management system for banking operations. It serves as both an educational tool, bridging theoretical knowledge with practical implementation, and a testament to the power of well-designed software in addressing real-world challenges.