Experiment No. – 11

(A Role-Based Banking Management System)

Case-study topic : A Bank Database that is used to keep the track of Customers .

```
mysql> create database Bankdb;
Query OK, 1 row affected (0.01 sec)

mysql> use Bankdb;
Database changed

mysql> CREATE TABLE Customer (
   ->     customer_id INT PRIMARY KEY AUTO_INCREMENT,
   ->     name VARCHAR(100) NOT NULL,
   ->     dob DATE NOT NULL,
   ->     gender ENUM('Male', 'Female', 'Other') NOT NULL,
   ->     phone VARCHAR(15) UNIQUE NOT NULL,
   ->     email VARCHAR(100) UNIQUE NOT NULL,
   ->     address TEXT NOT NULL,
   ->     nationality VARCHAR(50) DEFAULT 'Indian'
   -> );
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE TABLE Branch (
   ->     branch_id INT PRIMARY KEY AUTO_INCREMENT,
   ->     name VARCHAR(100) NOT NULL,
   ->     address TEXT NOT NULL,
   ->     city VARCHAR(50) NOT NULL,
   ->     pincode VARCHAR(10) NOT NULL,
   ->     CONSTRAINT chk_pincode CHECK (CHAR_LENGTH(pincode) = 6)
   -> );
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TABLE Employee (
   ->     employee_id INT PRIMARY KEY AUTO_INCREMENT,
   ->     name VARCHAR(100) NOT NULL,
   ->     designation VARCHAR(50) NOT NULL,
   ->     salary DECIMAL(12,2) NOT NULL CHECK (salary >= 10000),
   ->     branch_id INT NOT NULL,
   ->     email VARCHAR(100) UNIQUE NOT NULL,
   ->     FOREIGN KEY (branch_id) REFERENCES Branch(branch_id) ON DELETE CASCADE
   -> );
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> CREATE TABLE Account (
    ->     account_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     customer_id INT NOT NULL,
    ->     branch_id INT NOT NULL,
    ->     account_type ENUM('Saving', 'Current', 'Salary') NOT NULL,
    ->     balance DECIMAL(15,2) NOT NULL DEFAULT 0.00 CHECK (balance >= 0),
    ->     open_date DATE NOT NULL DEFAULT (CURRENT_DATE),
    ->     FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) ON DELETE
CASCADE,
    ->     FOREIGN KEY (branch_id) REFERENCES Branch(branch_id) ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE TABLE BankTransaction (  -- Notice: 'Transaction' is a reserved keyword,
changed name!
    ->     transaction_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     account_id INT NOT NULL,
    ->     transaction_type ENUM('Credit', 'Debit') NOT NULL,
    ->     amount DECIMAL(12,2) NOT NULL CHECK (amount > 0),
    ->     transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ->     FOREIGN KEY (account_id) REFERENCES Account(account_id) ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TABLE Loan (
    ->     loan_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     customer_id INT NOT NULL,
    ->     loan_type ENUM('Home', 'Car', 'Personal', 'Education') NOT NULL,
    ->     amount DECIMAL(15,2) NOT NULL CHECK (amount > 0),
    ->     interest_rate DECIMAL(5,2) NOT NULL CHECK (interest_rate > 0),
    ->     issued_date DATE NOT NULL DEFAULT (CURRENT_DATE),
    ->     due_date DATE NOT NULL,
    ->     status ENUM('Ongoing', 'Completed', 'Defaulted') DEFAULT 'Ongoing',
    ->     FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) ON DELETE
CASCADE
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TABLE Card (
    ->     card_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     customer_id INT NOT NULL,
    ->     card_type ENUM('Debit', 'Credit') NOT NULL,
    ->     card_number CHAR(16) UNIQUE NOT NULL,
```

```
   ->     expiry_date DATE NOT NULL,
   ->     cvv CHAR(3) NOT NULL CHECK (CHAR_LENGTH(cvv) = 3),
   ->     FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) ON DELETE
CASCADE
   -> );
Query OK, 0 rows affected (0.08 sec)

mysql> ALTER TABLE Customer ADD COLUMN password VARCHAR(100) NOT NULL;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Employee ADD COLUMN password VARCHAR(100) NOT NULL;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show tables;
+--------------------------+
| Tables_in_bankdb   |
+--------------------------+
| account                  |
| banktransaction      |
| branch                   |
| card                     |
| customer                 |
| employee                 |
| loan                     |
+--------------------------+
7 rows in set (0.01 sec)

mysql> select * from employee;
Empty set (0.00 sec)

mysql> drop table employee;
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE Employee (
   ->     employee_id INT AUTO_INCREMENT PRIMARY KEY,
   ->     name VARCHAR(100) NOT NULL,
   ->     designation VARCHAR(50) NOT NULL,
   ->     salary DECIMAL(12, 2) NOT NULL,
   ->     branch_id INT NOT NULL,
   ->     email VARCHAR(100) NOT NULL UNIQUE,
   ->     password VARCHAR(100) NOT NULL );
Query OK, 0 rows affected (0.04 sec)
```

mysql> desc account;

Field	Type	Null	Key	Default	Extra
account_id	int	NO	PRI	NULL	auto_increment
customer_id	int	NO	MUL	NULL	
branch_id	int	NO	MUL	NULL	
account_type	enum('Saving','Current','Salary')	NO		NULL	
balance	decimal(15,2)	NO		0.00	
open_date	date	NO		curdate()	DEFAULT_GENERATED

6 rows in set (0.03 sec)

mysql> desc banktransaction;

Field	Type	Null	Key	Default	Extra
transaction_id	int	NO	PRI	NULL	auto_increment
account_id	int	NO	MUL	NULL	
transaction_type	enum('Credit','Debit')	NO		NULL	
amount	decimal(12,2)	NO		NULL	
transaction_date	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

5 rows in set (0.00 sec)

mysql> desc branch;

Field	Type	Null	Key	Default	Extra
branch_id	int	NO	PRI	NULL	auto_increment
name	varchar(100)	NO		NULL	
address	text	NO		NULL	
city	varchar(50)	NO		NULL	
pincode	varchar(10)	NO		NULL	

5 rows in set (0.00 sec)

mysql> desc card;

Field	Type	Null	Key	Default	Extra
card_id	int	NO	PRI	NULL	auto_increment
customer_id	int	NO	MUL	NULL	
card_type	enum('Debit','Credit')	NO		NULL	
card_number	char(16)	NO	UNI	NULL	

```
| expiry_date   | date                      | NO   |         | NULL    |                      |
| cvv           | char(3)                   | NO   |         | NULL    |                      |
+---------------+---------------------------+------+---------+---------+----------------------+
6 rows in set (0.00 sec)

mysql> desc  customer;
+---------------+-------------------------------------+------+-------+-----------+----------------------+
| Field         | Type                                | Null | Key | Default | Extra              |
+---------------+-------------------------------------+------+-------+-----------+----------------------+
|customer_id| int                                 | NO   | PRI | NULL    | auto_increment |
| name          | varchar(100)                        | NO   |       | NULL    |                      |
| dob           | date                                | NO   |       | NULL    |                      |
| gender        |enum('Male','Female','Other')| NO   |       | NULL    |                      |
| phone         | varchar(15)                         | NO   | UNI| NULL    |                     |
| email         | varchar(100)                        | NO   | UNI| NULL    |                      |
| address       | text                                | NO   |       | NULL    |                      |
| nationality   | varchar(50)                         | YES  |       | Indian  |                      |
| password    | varchar(100)                        | NO   |       | NULL    |                      |
+---------------+-------------------------------------+------+-------+-----------+----------------------+
9 rows in set (0.00 sec)

mysql> desc employee;
+------------------+-----------------+-------+------+-----------+--------------------+
| Field            | Type            | Null | Key | Default  | Extra              |
+------------------+-----------------+-------+------+-----------+--------------------+
| employee_id | int             | NO   | PRI | NULL    |auto_increment|
| name           | varchar(100)  | NO   |       | NULL    |                    |
| designation   | varchar(50)    | NO   |       | NULL    |                    |
| salary          | decimal(12,2) | NO   |       | NULL    |                    |
| branch_id     | int               | NO   |       | NULL    |                    |
| email          | varchar(100)   | NO   | UNI | NULL    |                    |
| password     | varchar(100)   | NO   |       | NULL    |                    |
+------------------+-----------------+-------+------+-----------+--------------------+
7 rows in set (0.03 sec)

mysql> desc loan;
+------------------+-----------------------------------------------------+------+-----+-----------+-----------------+
| Field            | Type                                                | Null | Key | Default   | Extra       |
+------------------+-----------------------------------------------------+------+-----+-----------+-----------------+
| loan_id          | int                                                 |NO| PRI | NULL| auto_increment|
| customer_id  | int                                                 |NO|MUL| NULL|                 |
| loan_type     |enum('Home','Car','Personal','Education')|NO|      | NULL  |                |
| amount         | decimal(15,2)                                       |NO|      | NULL |                 |
| interest_rate | decimal(5,2)                                        |NO|      | NULL  |                 |
```

```
| issued_date   | date                                  | NO   |     | curdate()|DEFAULT_GENERATED |
| due_date      | date                                  | NO   |     | NULL     |                  |
| status        |enum('Ongoing','Completed','Defaulted')| YES  |     | Ongoing  |                  |
+---------------+---------------------------------------+------+-----+----------+------------------+
8 rows in set (0.00 sec)

mysql> INSERT INTO Branch (name, address, city, pincode)
   -> VALUES
   -> ('Karve ', ' Karve Naka ,Near  Thorat Hospital ', 'Karad', '415110'),
   -> ('Malkapur', 'Malkapur , Near D-Mart , Krishna Hospital', 'Malkapur', '415124'),
   -> ('Saidapur', 'Saidapur , Canol', 'Karad', '415114'),
   -> ('Karad', 'Karad , near bus stand', ' Karad', '415124');
Query OK, 4 rows affected (0.04 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from branch;
+----------+-------------+-----------------------------------------------------+-------------+------------+
|branch_id| name     | address                                               | city          | pincode  |
+----------+-------------+-----------------------------------------------------+-------------+------------+
|       1   | Karve       |  Karve Naka ,Near  Thorat Hospital            | Karad       | 415110  |
|       2   | Malkapur | Malkapur , Near D-Mart , Krishna Hospital | Malkapur | 415124  |
|       3   | Saidapur  | Saidapur , Canol                                  | Karad       | 415114  |
|       4   | Karad       | Karad , near bus stand                            |  Karad     | 415124  |
+----------+-------------+-----------------------------------------------------+-------------+------------+
4 rows in set (0.00 sec)
```

DBConnection.java

(The DBConnection class manages the process of loading the MySQL JDBC driver and establishing a connection to a MySQL database.)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
   // Database URL, username, and password
   private static final String URL = "jdbc:mysql://localhost:3306/Bankdb";  // Changed to
'Bankdb'
   private static final String USER = "root"; // MySQL username
   private static final String PASSWORD = "Sai@123"; // MySQL password

   // Static method to get a database connection
   public static Connection getConnection() throws SQLException {
      try {
         // Try explicitly loading the MySQL JDBC driver
         System.out.println("Loading MySQL JDBC Driver...");
         Class.forName("com.mysql.cj.jdbc.Driver");  // Ensures that the driver is loaded
         System.out.println("Driver loaded successfully!");
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
         throw new SQLException("MySQL Driver not found", e);
      }

      // Return the connection to the database
      return DriverManager.getConnection(URL, USER, PASSWORD);
   }
}
```

LoginService.java

(handles user login by verifying credentials against the selected role's table (customers, employees, or admins))

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;
```

```
public class LoginService {
    Scanner sc = new Scanner(System.in);

    public String login() {
        System.out.print("Enter Username: ");
        String username = sc.nextLine();

        System.out.print("Enter Password: ");
        String password = sc.nextLine();

        System.out.println("Select Role: 1. Customer 2. Employee 3. Admin");
        int choice = sc.nextInt();
        sc.nextLine(); // consume newline

        String roleTable = "";
        switch (choice) {
            case 1: roleTable = "customers"; break;
            case 2: roleTable = "employees"; break;
            case 3: roleTable = "admins"; break;
            default:
                System.out.println("Invalid role selected.");
                return null;
        }

        try (Connection conn = DBConnection.getConnection()) {
            String sql = "SELECT * FROM " + roleTable + " WHERE username = ? AND password =
?";
            PreparedStatement ps = conn.prepareStatement(sql);
            ps.setString(1, username);
            ps.setString(2, password);

            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                System.out.println("Login Successful as " + roleTable.toUpperCase());
                return roleTable;
            } else {
                System.out.println("Invalid Username or Password.");
                return null;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
```

}

AdminService.java

(to provide administrative functionalities for managing employee records in a bank database, including adding, viewing, searching, updating, deleting employees, and resetting their passwords using JDBC.)

```
import java.sql.*;
import java.util.Scanner;

public class AdminService {
   Scanner sc = new Scanner(System.in);

   // Add a new employee to the database
   public void addEmployee() {
      System.out.print("Enter Employee Name: ");
      String name = sc.nextLine();

      System.out.print("Enter Employee Designation: ");
      String designation = sc.nextLine();

      System.out.print("Enter Employee Salary: ");
      double salary = sc.nextDouble();
      sc.nextLine(); // consume newline

      System.out.print("Enter Employee Email: ");
      String email = sc.nextLine();

      System.out.print("Enter Branch ID: ");
      int branchId = sc.nextInt();
      sc.nextLine(); // consume newline

      System.out.print("Enter Employee Password: ");
      String password = sc.nextLine();

      try (Connection conn = DBConnection.getConnection()) {
         String sql = "INSERT INTO employee (name, designation, salary, branch_id, email,
password) VALUES (?, ?, ?, ?, ?, ?)";
         PreparedStatement ps = conn.prepareStatement(sql);
         ps.setString(1, name);
         ps.setString(2, designation);
         ps.setDouble(3, salary);
         ps.setInt(4, branchId);
         ps.setString(5, email);
```

```
        ps.setString(6, password);

        int rows = ps.executeUpdate();
        if (rows > 0) {
            System.out.println("Employee added successfully!");
        } else {
            System.out.println("Failed to add employee.");
        }
    } catch (SQLException e) {
        System.out.println("Error adding employee: " + e.getMessage());
    }
}

// View all employees
public void viewAllEmployees() {
    try (Connection conn = DBConnection.getConnection()) {
        String sql = "SELECT * FROM employee";
        PreparedStatement ps = conn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();

        while (rs.next()) {
            System.out.println("Employee ID: " + rs.getInt("employee_id"));
            System.out.println("Name: " + rs.getString("name"));
            System.out.println("Designation: " + rs.getString("designation"));
            System.out.println("Salary: " + rs.getDouble("salary"));
            System.out.println("Branch ID: " + rs.getInt("branch_id"));
            System.out.println("Email: " + rs.getString("email"));
            System.out.println("------------------------------");
        }
    } catch (SQLException e) {
        System.out.println("Error viewing all employees: " + e.getMessage());
    }
}

// Search employee by name or email
public void searchEmployee() {
    System.out.print("Enter Employee Name or Email to search: ");
    String searchTerm = sc.nextLine();

    try (Connection conn = DBConnection.getConnection()) {
        String sql = "SELECT * FROM employee WHERE name LIKE ? OR email LIKE ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setString(1, "%" + searchTerm + "%");
        ps.setString(2, "%" + searchTerm + "%");
```

```
        ResultSet rs = ps.executeQuery();
        while (rs.next()) {
            System.out.println("Employee ID: " + rs.getInt("employee_id"));
            System.out.println("Name: " + rs.getString("name"));
            System.out.println("Designation: " + rs.getString("designation"));
            System.out.println("Salary: " + rs.getDouble("salary"));
            System.out.println("Branch ID: " + rs.getInt("branch_id"));
            System.out.println("Email: " + rs.getString("email"));
            System.out.println("-------------------------------");
        }
    } catch (SQLException e) {
        System.out.println("Error searching employee: " + e.getMessage());
    }
}

// Update employee details
public void updateEmployeeDetails() {
    System.out.print("Enter Employee ID to update: ");
    int employeeId = sc.nextInt();
    sc.nextLine(); // consume newline

    System.out.println("Select the detail you want to update:");
    System.out.println("1. Name");
    System.out.println("2. Designation");
    System.out.println("3. Salary");
    System.out.println("4. Email");
    System.out.println("5. Branch ID");
    System.out.println("6. Password");
    System.out.print("Enter choice: ");
    int choice = sc.nextInt();
    sc.nextLine(); // consume newline

    String updateSql = "UPDATE employee SET ";
    try (Connection conn = DBConnection.getConnection()) {
        PreparedStatement psUpdate;

        switch (choice) {
            case 1:
                System.out.print("Enter new Name: ");
                updateSql += "name = ? WHERE employee_id = ?";
                psUpdate = conn.prepareStatement(updateSql);
                psUpdate.setString(1, sc.nextLine());
                psUpdate.setInt(2, employeeId);
```

```
            break;

        case 2:
            System.out.print("Enter new Designation: ");
            updateSql += "designation = ? WHERE employee_id = ?";
            psUpdate = conn.prepareStatement(updateSql);
            psUpdate.setString(1, sc.nextLine());
            psUpdate.setInt(2, employeeId);
            break;

        case 3:
            System.out.print("Enter new Salary: ");
            updateSql += "salary = ? WHERE employee_id = ?";
            psUpdate = conn.prepareStatement(updateSql);
            psUpdate.setDouble(1, sc.nextDouble());
            sc.nextLine();
            psUpdate.setInt(2, employeeId);
            break;

        case 4:
            System.out.print("Enter new Email: ");
            updateSql += "email = ? WHERE employee_id = ?";
            psUpdate = conn.prepareStatement(updateSql);
            psUpdate.setString(1, sc.nextLine());
            psUpdate.setInt(2, employeeId);
            break;

        case 5:
            System.out.print("Enter new Branch ID: ");
            updateSql += "branch_id = ? WHERE employee_id = ?";
            psUpdate = conn.prepareStatement(updateSql);
            psUpdate.setInt(1, sc.nextInt());
            sc.nextLine();
            psUpdate.setInt(2, employeeId);
            break;

        case 6:
            System.out.print("Enter new Password: ");
            updateSql += "password = ? WHERE employee_id = ?";
            psUpdate = conn.prepareStatement(updateSql);
            psUpdate.setString(1, sc.nextLine());
            psUpdate.setInt(2, employeeId);
            break;
```

```
            default:
                System.out.println("Invalid choice.");
                return;
        }

        int rowsUpdated = psUpdate.executeUpdate();
        if (rowsUpdated > 0) {
            System.out.println("Employee details updated successfully!");
        } else {
            System.out.println("Failed to update employee details.");
        }
    } catch (SQLException e) {
        System.out.println("Error updating employee: " + e.getMessage());
    }
}

// Delete employee and reorder IDs (for demo only)
public void deleteEmployee() {
    System.out.print("Enter Employee ID to delete: ");
    int employeeId = sc.nextInt();
    sc.nextLine();

    try (Connection conn = DBConnection.getConnection()) {
        conn.setAutoCommit(false); // Transaction begins

        String sqlDelete = "DELETE FROM employee WHERE employee_id = ?";
        PreparedStatement psDelete = conn.prepareStatement(sqlDelete);
        psDelete.setInt(1, employeeId);

        int rows = psDelete.executeUpdate();
        if (rows > 0) {
            System.out.println("Employee deleted successfully!");

            // Reorder IDs (for demo only, not recommended in production)
            Statement stmt = conn.createStatement();
            stmt.executeUpdate("SET @count = 0");
            stmt.executeUpdate("UPDATE employee SET employee_id = @count:=@count+1");
            stmt.executeUpdate("ALTER TABLE employee AUTO_INCREMENT = 1");

            System.out.println("Employee IDs reordered.");
        } else {
            System.out.println("No such employee found!");
        }
```

```
            conn.commit(); // Transaction ends
        } catch (SQLException e) {
            System.out.println("Error deleting employee: " + e.getMessage());
        }
    }

    // Reset employee password
    public void resetEmployeePassword() {
        System.out.print("Enter Employee ID to reset password: ");
        int employeeId = sc.nextInt();
        sc.nextLine();

        System.out.print("Enter new password: ");
        String newPassword = sc.nextLine();

        try (Connection conn = DBConnection.getConnection()) {
            String sql = "UPDATE employee SET password = ? WHERE employee_id = ?";
            PreparedStatement ps = conn.prepareStatement(sql);
            ps.setString(1, newPassword);
            ps.setInt(2, employeeId);

            int rowsUpdated = ps.executeUpdate();
            if (rowsUpdated > 0) {
                System.out.println("Password reset successfully!");
            } else {
                System.out.println("Employee not found or password not updated.");
            }
        } catch (SQLException e) {
            System.out.println("Error resetting password: " + e.getMessage());
        }
    }
}
```

EmployeeService.java

(provides bank employees with functionalities to manage customers, accounts, loans, and branches through database operations.)

```
import java.sql.*;
import java.util.Scanner;

public class EmployeeService {
    Scanner sc = new Scanner(System.in);

    // Add a new customer to the database
```

```
    public void addCustomer() {
        System.out.print("Enter Customer Name: ");
        String name = sc.nextLine();

        System.out.print("Enter Date of Birth (yyyy-mm-dd): ");
        String dob = sc.nextLine();

        System.out.print("Enter Gender (Male/Female/Other): ");
        String gender = sc.nextLine();

        System.out.print("Enter Customer Phone: ");
        String phone = sc.nextLine();

        System.out.print("Enter Customer Email: ");
        String email = sc.nextLine();

        System.out.print("Enter Customer Address: ");
        String address = sc.nextLine();

        System.out.print("Enter Nationality (default Indian): ");
        String nationality = sc.nextLine();
        if (nationality.isEmpty()) {
            nationality = "Indian";
        }

        System.out.print("Enter Customer Password: ");
        String password = sc.nextLine();

        try (Connection conn = DBConnection.getConnection()) {
            String sql = "INSERT INTO Customer (name, dob, gender, phone, email, address,
nationality, password) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
            PreparedStatement ps = conn.prepareStatement(sql);
            ps.setString(1, name);
            ps.setDate(2, Date.valueOf(dob));
            ps.setString(3, gender);
            ps.setString(4, phone);
            ps.setString(5, email);
            ps.setString(6, address);
            ps.setString(7, nationality);
            ps.setString(8, password);

            int rows = ps.executeUpdate();
            if (rows > 0) {
                System.out.println("Customer added successfully!");
```

```
            } else {
                System.out.println("Failed to add customer.");
            }
        } catch (SQLException e) {
            System.out.println("Error adding customer: " + e.getMessage());
        }
    }

    // Fetch and display all branch details
    public void showBranchDetails() {
        try (Connection conn = DBConnection.getConnection()) {
            String sql = "SELECT branch_id, branch_name FROM Branch"; // Assuming you have a
Branch table
            PreparedStatement ps = conn.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();

            System.out.println("Available Branches:");
            while (rs.next()) {
                int branchId = rs.getInt("branch_id");
                String branchName = rs.getString("branch_name");
                System.out.println("Branch ID: " + branchId + " - Branch Name: " + branchName);
            }
        } catch (SQLException e) {
            System.out.println("Error fetching branches: " + e.getMessage());
        }
    }

    // Add account for a customer with branch selection
    public void addAccountForCustomer() {
        // Show available branches before asking for the branch ID
        showBranchDetails();

        System.out.print("Enter Customer ID: ");
        int customerId = sc.nextInt();
        sc.nextLine(); // consume newline

        System.out.print("Enter Branch ID: ");
        int branchId = sc.nextInt();
        sc.nextLine(); // consume newline

        // Now proceed with the logic for adding an account for the customer
        try (Connection conn = DBConnection.getConnection()) {
            String sql = "INSERT INTO Account (customer_id, branch_id) VALUES (?, ?)";
            PreparedStatement ps = conn.prepareStatement(sql);
```

```
        ps.setInt(1, customerId);
        ps.setInt(2, branchId);

        int rows = ps.executeUpdate();
        if (rows > 0) {
            System.out.println("Account created successfully!");
        } else {
            System.out.println("Failed to create account.");
        }
    } catch (SQLException e) {
        System.out.println("Error adding account: " + e.getMessage());
    }
}

// View all loans
public void viewAllLoans() {
    try (Connection conn = DBConnection.getConnection()) {
        String sql = "SELECT * FROM Loan";
        PreparedStatement ps = conn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();

        while (rs.next()) {
            System.out.println("Loan ID: " + rs.getInt("loan_id"));
            System.out.println("Loan Amount: " + rs.getDouble("amount"));
            System.out.println("Loan Status: " + rs.getString("status"));
            System.out.println("-------------------------------");
        }
    } catch (SQLException e) {
        System.out.println("Error viewing all loans: " + e.getMessage());
    }
}

// Approve a loan
public void approveLoan() {
    System.out.print("Enter Loan ID to approve: ");
    int loanId = sc.nextInt();
    sc.nextLine(); // consume newline

    try (Connection conn = DBConnection.getConnection()) {
        String sql = "UPDATE Loan SET status = 'approved' WHERE loan_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, loanId);

        int rowsUpdated = ps.executeUpdate();
```

```
        if (rowsUpdated > 0) {
            System.out.println("Loan approved successfully!");
        } else {
            System.out.println("Loan not found or approval failed.");
        }
    } catch (SQLException e) {
        System.out.println("Error approving loan: " + e.getMessage());
    }
}

// Reject a loan
public void rejectLoan() {
    System.out.print("Enter Loan ID to reject: ");
    int loanId = sc.nextInt();
    sc.nextLine(); // consume newline

    try (Connection conn = DBConnection.getConnection()) {
        String sql = "UPDATE Loan SET status = 'rejected' WHERE loan_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, loanId);

        int rowsUpdated = ps.executeUpdate();
        if (rowsUpdated > 0) {
            System.out.println("Loan rejected successfully!");
        } else {
            System.out.println("Loan not found or rejection failed.");
        }
    } catch (SQLException e) {
        System.out.println("Error rejecting loan: " + e.getMessage());
    }
}

// View customer details
public void viewCustomerDetails() {
    System.out.print("Enter Customer ID to view details: ");
    int customerId = sc.nextInt();
    sc.nextLine();

    try (Connection conn = DBConnection.getConnection()) {
        String sql = "SELECT * FROM Customer WHERE customer_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, customerId);

        ResultSet rs = ps.executeQuery();
```

```
        if (rs.next()) {
            System.out.println("Customer ID: " + rs.getInt("customer_id"));
            System.out.println("Name: " + rs.getString("name"));
            System.out.println("DOB: " + rs.getDate("dob"));
            System.out.println("Gender: " + rs.getString("gender"));
            System.out.println("Phone: " + rs.getString("phone"));
            System.out.println("Email: " + rs.getString("email"));
            System.out.println("Address: " + rs.getString("address"));
            System.out.println("Nationality: " + rs.getString("nationality"));
            System.out.println("-------------------------------");
        } else {
            System.out.println("No customer found with that ID.");
        }
    } catch (SQLException e) {
        System.out.println("Error viewing customer details: " + e.getMessage());
    }
}

// Update customer details
public void updateCustomerDetails() {
    System.out.print("Enter Customer ID to update: ");
    int customerId = sc.nextInt();
    sc.nextLine();

    System.out.println("Select the detail you want to update:");
    System.out.println("1. Name");
    System.out.println("2. Email");
    System.out.println("3. Phone");
    System.out.println("4. Address");
    System.out.println("5. Password");
    int choice = sc.nextInt();
    sc.nextLine();

    String column = null;
    String newValue;

    switch (choice) {
        case 1:
            column = "name";
            break;
        case 2:
            column = "email";
            break;
        case 3:
```

```
            column = "phone";
            break;
        case 4:
            column = "address";
            break;
        case 5:
            column = "password";
            break;
        default:
            System.out.println("Invalid choice.");
            return;
    }

    System.out.print("Enter new value: ");
    newValue = sc.nextLine();

    try (Connection conn = DBConnection.getConnection()) {
        String sql = "UPDATE Customer SET " + column + " = ? WHERE customer_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setString(1, newValue);
        ps.setInt(2, customerId);

        int rowsUpdated = ps.executeUpdate();
        if (rowsUpdated > 0) {
            System.out.println("Customer " + column + " updated successfully!");
        } else {
            System.out.println("Failed to update customer.");
        }
    } catch (SQLException e) {
        System.out.println("Error updating customer: " + e.getMessage());
    }
}

// Delete customer and resequence customer_id
public void deleteCustomer() {
    System.out.print("Enter Customer ID to delete: ");
    int customerId = sc.nextInt();
    sc.nextLine();

    try (Connection conn = DBConnection.getConnection()) {
        conn.setAutoCommit(false);

        String deleteSql = "DELETE FROM Customer WHERE customer_id = ?";
        PreparedStatement psDelete = conn.prepareStatement(deleteSql);
```

```
        psDelete.setInt(1, customerId);
        int rows = psDelete.executeUpdate();

        if (rows > 0) {
            // Resequence IDs
            String resequence = "SET @count = 0; " +
                    "UPDATE Customer SET customer_id = @count:=@count+1; " +
                    "ALTER TABLE Customer AUTO_INCREMENT = 1;";
            Statement stmt = conn.createStatement();
            for (String sql : resequence.split(";")) {
                if (!sql.trim().isEmpty()) {
                    stmt.execute(sql.trim());
                }
            }

            conn.commit();
            System.out.println("Customer deleted and IDs resequenced.");
        } else {
            conn.rollback();
            System.out.println("No such customer found.");
        }
    } catch (SQLException e) {
        System.out.println("Error deleting customer: " + e.getMessage());
    }
}

// Search customer by name or email
public void searchCustomer() {
    System.out.print("Enter Customer Name or Email to search: ");
    String searchTerm = sc.nextLine();

    try (Connection conn = DBConnection.getConnection()) {
        String sql = "SELECT * FROM Customer WHERE name LIKE ? OR email LIKE ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setString(1, "%" + searchTerm + "%");
        ps.setString(2, "%" + searchTerm + "%");

        ResultSet rs = ps.executeQuery();
        while (rs.next()) {
            System.out.println("Customer ID: " + rs.getInt("customer_id"));
            System.out.println("Name: " + rs.getString("name"));
            System.out.println("Email: " + rs.getString("email"));
            System.out.println("Phone: " + rs.getString("phone"));
            System.out.println("Address: " + rs.getString("address"));
```

```
            System.out.println("-------------------------------");
        }
    } catch (SQLException e) {
        System.out.println("Error searching for customer: " + e.getMessage());
    }
}

// Reset customer password
public void resetCustomerPassword() {
    System.out.print("Enter Customer ID to reset password: ");
    int customerId = sc.nextInt();
    sc.nextLine();

    System.out.print("Enter new password: ");
    String newPassword = sc.nextLine();

    try (Connection conn = DBConnection.getConnection()) {
        String sql = "UPDATE Customer SET password = ? WHERE customer_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setString(1, newPassword);
        ps.setInt(2, customerId);

        int rowsUpdated = ps.executeUpdate();
        if (rowsUpdated > 0) {
            System.out.println("Password reset successfully!");
        } else {
            System.out.println("Customer not found or password not updated.");
        }
    } catch (SQLException e) {
        System.out.println("Error resetting password: " + e.getMessage());
    }
}
}
```

CustomerService.java

(provides banking functionalities for customers, including viewing account details, depositing, withdrawing, transferring money, applying for loans, and recording transactions in a database.)

```
import java.util.Scanner;
import java.sql.*;
import java.time.LocalDate;

public class CustomerService {
    Scanner sc = new Scanner(System.in);
```

```
// View account details
public void viewAccount(int customerId) {
    try (Connection conn = DBConnection.getConnection()) {
        String sql = "SELECT * FROM Account WHERE customer_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, customerId);

        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            System.out.println("Account ID: " + rs.getInt("account_id"));
            System.out.println("Customer ID: " + rs.getInt("customer_id"));
            System.out.println("Account Balance: " + rs.getDouble("balance"));
            System.out.println("Open Date: " + rs.getDate("open_date"));
        } else {
            System.out.println("No account found for this customer.");
        }
    } catch (SQLException e) {
        System.out.println("Error viewing account: " + e.getMessage());
    }
}

// Deposit money
public void depositMoney(double amount, int customerId) {
    System.out.println("Depositing $" + amount + " into your account...");

    try (Connection conn = DBConnection.getConnection()) {
        int accountId = getAccountId(conn, customerId);
        if (accountId == -1) {
            System.out.println("No account found for this customer.");
            return;
        }

        String sql = "UPDATE Account SET balance = balance + ? WHERE account_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setDouble(1, amount);
        ps.setInt(2, accountId);

        int rowsUpdated = ps.executeUpdate();
        if (rowsUpdated > 0) {
            System.out.println("Deposit successful!");
            recordTransaction(conn, accountId, "Credit", amount);
        } else {
            System.out.println("Deposit failed.");
```

```
        }
    } catch (SQLException e) {
        System.out.println("Error depositing money: " + e.getMessage());
    }
}

// Withdraw money
public void withdrawMoney(double amount, int customerId) {
    System.out.println("Withdrawing $" + amount + " from your account...");

    try (Connection conn = DBConnection.getConnection()) {
        int accountId = getAccountId(conn, customerId);
        if (accountId == -1) {
            System.out.println("No account found.");
            return;
        }

        String sql = "SELECT balance FROM Account WHERE account_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, accountId);
        ResultSet rs = ps.executeQuery();

        if (rs.next()) {
            double currentBalance = rs.getDouble("balance");
            if (currentBalance >= amount) {
                sql = "UPDATE Account SET balance = balance - ? WHERE account_id = ?";
                ps = conn.prepareStatement(sql);
                ps.setDouble(1, amount);
                ps.setInt(2, accountId);
                int rowsUpdated = ps.executeUpdate();
                if (rowsUpdated > 0) {
                    System.out.println("Withdrawal successful!");
                    recordTransaction(conn, accountId, "Debit", amount);
                } else {
                    System.out.println("Withdrawal failed.");
                }
            } else {
                System.out.println("Insufficient funds.");
            }
        }
    } catch (SQLException e) {
        System.out.println("Error withdrawing money: " + e.getMessage());
    }
}
```

```
    // Transfer money
    public void transferMoney(double amount, int customerId, int recipientAccountId) {
        System.out.println("Transferring $" + amount + " to account number " +
recipientAccountId);

        try (Connection conn = DBConnection.getConnection()) {
            int senderAccountId = getAccountId(conn, customerId);
            if (senderAccountId == -1) {
                System.out.println("Sender account not found.");
                return;
            }

            // Check if recipient exists
            String sql = "SELECT balance FROM Account WHERE account_id = ?";
            PreparedStatement ps = conn.prepareStatement(sql);
            ps.setInt(1, recipientAccountId);
            ResultSet rs = ps.executeQuery();

            if (!rs.next()) {
                System.out.println("Transfer failed. Recipient not found.");
                return;
            }

            // Check sender balance
            sql = "SELECT balance FROM Account WHERE account_id = ?";
            ps = conn.prepareStatement(sql);
            ps.setInt(1, senderAccountId);
            rs = ps.executeQuery();

            if (rs.next()) {
                double currentBalance = rs.getDouble("balance");
                if (currentBalance >= amount) {
                    // Deduct from sender
                    sql = "UPDATE Account SET balance = balance - ? WHERE account_id = ?";
                    ps = conn.prepareStatement(sql);
                    ps.setDouble(1, amount);
                    ps.setInt(2, senderAccountId);
                    int rowsUpdated = ps.executeUpdate();

                    if (rowsUpdated > 0) {
                        // Credit to recipient
                        sql = "UPDATE Account SET balance = balance + ? WHERE account_id = ?";
                        ps = conn.prepareStatement(sql);
```

```
                ps.setDouble(1, amount);
                ps.setInt(2, recipientAccountId);
                rowsUpdated = ps.executeUpdate();

                if (rowsUpdated > 0) {
                    System.out.println("Transfer successful!");
                    recordTransaction(conn, senderAccountId, "Debit", amount);
                    recordTransaction(conn, recipientAccountId, "Credit", amount);
                } else {
                    System.out.println("Transfer failed. Recipient account update failed.");
                }
            } else {
                System.out.println("Transfer failed. Sender account update failed.");
            }
        } else {
            System.out.println("Insufficient funds.");
        }
    }
  } catch (SQLException e) {
    System.out.println("Error transferring money: " + e.getMessage());
  }
}

// Apply for loan
public void applyLoan(int customerId) {
  System.out.print("Enter loan amount: ");
  double loanAmount = sc.nextDouble();
  sc.nextLine(); // Consume newline

  System.out.print("Enter loan type (Home, Car, Personal, Education): ");
  String loanType = sc.nextLine().trim();

  System.out.print("Enter interest rate (e.g., 7.5): ");
  double interestRate = sc.nextDouble();

  System.out.print("Enter loan term (in months): ");
  int termMonths = sc.nextInt();
  sc.nextLine(); // Consume newline

  LocalDate issuedDate = LocalDate.now();
  LocalDate dueDate = issuedDate.plusMonths(termMonths);

  try (Connection conn = DBConnection.getConnection()) {
```

```
        String sql = "INSERT INTO Loan (customer_id, loan_type, amount, interest_rate,
issued_date, due_date, status) " +
                     "VALUES (?, ?, ?, ?, ?, ?, 'Ongoing')";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, customerId);
        ps.setString(2, loanType);
        ps.setDouble(3, loanAmount);
        ps.setDouble(4, interestRate);
        ps.setDate(5, Date.valueOf(issuedDate));
        ps.setDate(6, Date.valueOf(dueDate));

        int rowsInserted = ps.executeUpdate();
        if (rowsInserted > 0) {
          System.out.println("Loan application submitted successfully!");
        } else {
          System.out.println("Failed to apply for loan.");
        }
      } catch (SQLException e) {
        System.out.println("Error applying for loan: " + e.getMessage());
      }
    }

    // Record transaction in banktransaction table
    private void recordTransaction(Connection conn, int accountId, String transactionType,
double amount) {
      try {
        String sql = "INSERT INTO banktransaction (account_id, transaction_type, amount)
VALUES (?, ?, ?)";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, accountId);
        ps.setString(2, transactionType); // Must be 'Credit' or 'Debit'
        ps.setDouble(3, amount);
        ps.executeUpdate();
      } catch (SQLException e) {
        System.out.println("Error recording transaction: " + e.getMessage());
      }
    }

    // Helper to get account ID from customer ID
    private int getAccountId(Connection conn, int customerId) throws SQLException {
      String sql = "SELECT account_id FROM Account WHERE customer_id = ?";
      PreparedStatement ps = conn.prepareStatement(sql);
      ps.setInt(1, customerId);
      ResultSet rs = ps.executeQuery();
```

```
        if (rs.next()) {
            return rs.getInt("account_id");
        }
        return -1;
    }
}
```

AccountService.java

(manages customer accounts by providing methods to view accounts by customer ID and add new accounts to the database.)

```
import java.sql.*;
public class AccountService {
// Method to view accounts by customer ID
public void viewAccountsByCustomerId(int customerId) {
try (Connection conn = DBConnection.getConnection()) {
String sql = "SELECT * FROM account WHERE customer_id = ?";
PreparedStatement ps = conn.prepareStatement(sql);
ps.setInt(1, customerId);
ResultSet rs = ps.executeQuery();
while (rs.next()) {
System.out.printf("Account ID: %d | Type: %s | Balance: %.2f | Open Date: %s\n",
rs.getInt("account_id"),
rs.getString("account_type"),
rs.getDouble("balance"),
rs.getDate("open_date"));
}
} catch (SQLException e) {
e.printStackTrace();
}
}
// Method to add an account for a customer
public void addAccount(int customerId, int branchId, String accountType) {
String sql = "INSERT INTO account (customer_id, branch_id, account_type, balance,
open_date) VALUES (?, ?, ?, 0.0, CURRENT_DATE)";
try (Connection conn = DBConnection.getConnection()) {
PreparedStatement ps = conn.prepareStatement(sql);
ps.setInt(1, customerId);
ps.setInt(2, branchId);
ps.setString(3, accountType);
int rowsAffected = ps.executeUpdate();
if (rowsAffected > 0) {
System.out.println("Account added successfully for customer ID: " + customerId);
```

```
} else {
System.out.println("Failed to add account.");
}
} catch (SQLException e) {
e.printStackTrace();
}
}
}
```

BankTransaction.java

(The BankTransactionService class retrieves and displays all bank transactions for a specified account, ordered by transaction date.)

```
import java.sql.*;

public class BankTransactionService {
   public void viewTransactions(int accountId) {
      try (Connection conn = DBConnection.getConnection()) {
         String sql = "SELECT * FROM banktransaction WHERE account_id = ? ORDER BY
transaction_date DESC";
         PreparedStatement ps = conn.prepareStatement(sql);
         ps.setInt(1, accountId);
         ResultSet rs = ps.executeQuery();

         while (rs.next()) {
            System.out.printf("Transaction ID: %d | Type: %s | Amount: %.2f | Date: %s\n",
                  rs.getInt("transaction_id"),
                  rs.getString("transaction_type"),
                  rs.getDouble("amount"),
                  rs.getTimestamp("transaction_date"));
         }
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
}
```

CardService.java

(The CardService class manages card issuance (debit/credit), validates card details, and allows viewing of all cards for a specific customer.)

```
import java.sql.*;
import java.util.Random;
```

```
import java.util.Scanner;

public class CardService {
   Scanner sc = new Scanner(System.in);

   // Generate a 16-digit card number
   private String generateCardNumber() {
      StringBuilder cardNumber = new StringBuilder(16);
      Random rand = new Random();
      for (int i = 0; i < 16; i++) {
         cardNumber.append(rand.nextInt(10));
      }
      return cardNumber.toString();
   }

   // Issue a card to a customer
   public void issueCard(int customerId, String cardType) {
      if (!cardType.equalsIgnoreCase("Debit") && !cardType.equalsIgnoreCase("Credit")) {
         System.out.println("Invalid card type. Please choose either 'Debit' or 'Credit'.");
         return;
      }

      String cardNumber = generateCardNumber();

      System.out.print("Enter expiry date (yyyy-mm-dd): ");
      String expiryDateStr = sc.nextLine();
      Date expiryDate;

      try {
         expiryDate = Date.valueOf(expiryDateStr);
      } catch (IllegalArgumentException e) {
         System.out.println("Invalid date format. Use yyyy-mm-dd.");
         return;
      }

      System.out.print("Enter CVV (3 digits): ");
      String cvv = sc.nextLine();

      if (!cvv.matches("\\d{3}")) {
         System.out.println("Invalid CVV. Please enter exactly 3 digits.");
         return;
      }

      try (Connection conn = DBConnection.getConnection()) {
```

```
        String sql = "INSERT INTO Card (customer_id, card_number, card_type, expiry_date,
cvv) VALUES (?, ?, ?, ?, ?)";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, customerId);
        ps.setString(2, cardNumber);
        ps.setString(3, cardType);
        ps.setDate(4, expiryDate);
        ps.setString(5, cvv);

        int rows = ps.executeUpdate();
        if (rows > 0) {
          System.out.println("✅ Card issued successfully with card number: " +
cardNumber);
        } else {
          System.out.println("❌ Failed to issue card.");
        }
      } catch (SQLException e) {
        System.out.println("❌ Error issuing card: " + e.getMessage());
      }
    }

    // Menu to issue card
    public void issueCardMenu() {
      System.out.print("Enter customer ID: ");
      int customerId = sc.nextInt();
      sc.nextLine(); // consume newline
      System.out.print("Enter card type (Debit/Credit): ");
      String cardType = sc.nextLine();
      issueCard(customerId, cardType);
    }

    // ✅ View all cards for a specific customer
    public void viewCardsByCustomerId(int customerId) {
      try (Connection conn = DBConnection.getConnection()) {
        String sql = "SELECT * FROM Card WHERE customer_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, customerId);
        ResultSet rs = ps.executeQuery();

        boolean hasResults = false;
        while (rs.next()) {
          hasResults = true;
          System.out.println("Card ID: " + rs.getInt("card_id"));
```

```
            System.out.println("Card Type: " + rs.getString("card_type"));
            System.out.println("Card Number: " + rs.getString("card_number"));
            System.out.println("Expiry Date: " + rs.getDate("expiry_date"));
            System.out.println("CVV: " + rs.getString("cvv"));
            System.out.println("----------------------------");
        }

        if (!hasResults) {
            System.out.println("No cards found for customer ID: " + customerId);
        }

    } catch (SQLException e) {
        System.out.println("❌ Error retrieving cards: " + e.getMessage());
    }
  }
}
```

LoanService.java

(The LoanService class handles operations related to loans, including viewing loans, repaying loans, and checking loan status for customers.)

```
import java.sql.*;
import java.util.Scanner;

public class LoanService {
  Scanner sc = new Scanner(System.in);

  // View loan details for a customer
  public void viewLoans(int customerId) {
    try (Connection conn = DBConnection.getConnection()) {
      String sql = "SELECT * FROM Loan WHERE customer_id = ?";
      PreparedStatement ps = conn.prepareStatement(sql);
      ps.setInt(1, customerId);

      ResultSet rs = ps.executeQuery();
      boolean found = false;
      while (rs.next()) {
        found = true;
        System.out.println("Loan ID: " + rs.getInt("loan_id"));
        System.out.println("Loan Type: " + rs.getString("loan_type"));
        System.out.println("Amount: " + rs.getDouble("amount"));
        System.out.println("Interest Rate: " + rs.getDouble("interest_rate"));
        System.out.println("Issued Date: " + rs.getDate("issued_date"));
        System.out.println("Due Date: " + rs.getDate("due_date"));
```

```
            System.out.println("Status: " + rs.getString("status"));
            System.out.println("-----------------------------------");
        }

        if (!found) {
            System.out.println("No loans found for this customer.");
        }
    } catch (SQLException e) {
        System.out.println("Error viewing loans: " + e.getMessage());
    }
}

// Repay loan
public void repayLoan(int customerId, int loanId, double amount) {
    try (Connection conn = DBConnection.getConnection()) {
        String sql = "SELECT * FROM Loan WHERE loan_id = ? AND customer_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, loanId);
        ps.setInt(2, customerId);

        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            double loanAmount = rs.getDouble("amount");
            if (loanAmount >= amount) {
                // Update loan amount after repayment
                sql = "UPDATE Loan SET amount = amount - ? WHERE loan_id = ?";
                ps = conn.prepareStatement(sql);
                ps.setDouble(1, amount);
                ps.setInt(2, loanId);
                int rowsUpdated = ps.executeUpdate();

                if (rowsUpdated > 0) {
                    System.out.println("Repayment successful!");
                    // Check if the loan is fully repaid and update status
                    if (loanAmount - amount == 0) {
                        sql = "UPDATE Loan SET status = 'Completed' WHERE loan_id = ?";
                        ps = conn.prepareStatement(sql);
                        ps.setInt(1, loanId);
                        ps.executeUpdate();
                        System.out.println("Loan fully repaid, status updated to 'Completed'.");
                    }
                } else {
                    System.out.println("Error repaying the loan.");
                }
```

```
            } else {
                System.out.println("Repayment amount exceeds the loan amount.");
            }
        } else {
            System.out.println("Loan not found for this customer.");
        }
    } catch (SQLException e) {
        System.out.println("Error repaying loan: " + e.getMessage());
    }
  }

  // Check loan status
  public void checkLoanStatus(int customerId, int loanId) {
    try (Connection conn = DBConnection.getConnection()) {
        String sql = "SELECT status FROM Loan WHERE loan_id = ? AND customer_id = ?";
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setInt(1, loanId);
        ps.setInt(2, customerId);

        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            System.out.println("Loan Status: " + rs.getString("status"));
        } else {
            System.out.println("Loan not found for this customer.");
        }
    } catch (SQLException e) {
        System.out.println("Error checking loan status: " + e.getMessage());
    }
  }
}
```

MenuService.java

(This class manages the different role-based menus (customer, employee, admin) for an online banking system, allowing users to interact with their accounts and perform various financial operations. Each menu offers a set of functionalities specific to the user role.)

```
import java.util.Scanner;

public class MenuService {
  Scanner sc = new Scanner(System.in);

  public void start() {
    LoginService loginService = new LoginService();
    String role = loginService.login();
```

```
    if (role != null) {
        loadMenu(role);
    } else {
        System.out.println("Login failed! Please try again.");
    }
}

public void loadMenu(String role) {
    switch (role.toLowerCase()) {
        case "customer":
            customerMenu();
            break;
        case "employee":
            employeeMenu();
            break;
        case "admin":
            adminMenu();
            break;
        default:
            System.out.println("Invalid role access!");
    }
}

private void customerMenu() {
    CustomerService customerService = new CustomerService();
    AccountService accountService = new AccountService();
    CardService cardService = new CardService();
    LoanService loanService = new LoanService();
    int choice;

    do {
        System.out.println("\n=== Customer Dashboard ===");
        System.out.println("1. View Account");
        System.out.println("2. Deposit Money");
        System.out.println("3. Withdraw Money");
        System.out.println("4. Transfer Money");
        System.out.println("5. Apply for Loan");
        System.out.println("6. View My Cards");
        System.out.println("7. View My Loans");         // New
        System.out.println("8. Repay Loan");            // New
        System.out.println("9. Check Loan Status");     // New
        System.out.println("10. Logout");
        System.out.print("Choose: ");
```

```
choice = sc.nextInt();

switch (choice) {
   case 1 -> {
      System.out.print("Enter your customer ID: ");
      int customerId = sc.nextInt();
      accountService.viewAccountsByCustomerId(customerId);
   }
   case 2 -> {
      System.out.print("Enter customer ID: ");
      int customerId = sc.nextInt();
      System.out.print("Enter amount to deposit: ");
      double depositAmount = sc.nextDouble();
      customerService.depositMoney(depositAmount, customerId);
   }
   case 3 -> {
      System.out.print("Enter customer ID: ");
      int customerId = sc.nextInt();
      System.out.print("Enter amount to withdraw: ");
      double withdrawalAmount = sc.nextDouble();
      customerService.withdrawMoney(withdrawalAmount, customerId);
   }
   case 4 -> {
      System.out.print("Enter customer ID: ");
      int customerId = sc.nextInt();
      System.out.print("Enter amount to transfer: ");
      double transferAmount = sc.nextDouble();
      System.out.print("Enter recipient account number: ");
      int recipientAccount = sc.nextInt();
      customerService.transferMoney(transferAmount, customerId, recipientAccount);
   }
   case 5 -> {
      System.out.print("Enter customer ID: ");
      int customerId = sc.nextInt();
      customerService.applyLoan(customerId);
   }
   case 6 -> {
      System.out.print("Enter your customer ID: ");
      int customerId = sc.nextInt();
      cardService.viewCardsByCustomerId(customerId);
   }
   case 7 -> {
      System.out.print("Enter your customer ID: ");
      int customerId = sc.nextInt();
```

```
                loanService.viewLoans(customerId);
            }
            case 8 -> {
                System.out.print("Enter your customer ID: ");
                int customerId = sc.nextInt();
                System.out.print("Enter loan ID to repay: ");
                int loanId = sc.nextInt();
                System.out.print("Enter amount to repay: ");
                double repayAmount = sc.nextDouble();
                loanService.repayLoan(customerId, loanId, repayAmount);
            }
            case 9 -> {
                System.out.print("Enter your customer ID: ");
                int customerId = sc.nextInt();
                System.out.print("Enter loan ID to check status: ");
                int loanId = sc.nextInt();
                loanService.checkLoanStatus(customerId, loanId);
            }
            case 10 -> System.out.println("Logging out...");
            default -> System.out.println("Invalid choice!");
        }
    } while (choice != 10);
}

private void employeeMenu() {
    EmployeeService employeeService = new EmployeeService();
    AccountService accountService = new AccountService();
    CardService cardService = new CardService();
    int choice;

    do {
        System.out.println("\n=== Employee Dashboard ===");
        System.out.println("1. Add Customer");
        System.out.println("2. View Customer Details");
        System.out.println("3. Update Customer Details");
        System.out.println("4. Delete Customer");
        System.out.println("5. Search Customer");
        System.out.println("6. Reset Customer Password");
        System.out.println("7. View All Loans");
        System.out.println("8. Approve Loan");
        System.out.println("9. Reject Loan");
        System.out.println("10. Add Account for Customer");
        System.out.println("11. Issue Card to Customer");
        System.out.println("12. Logout");
```

```
            System.out.print("Choose: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1 -> employeeService.addCustomer();
                case 2 -> employeeService.viewCustomerDetails();
                case 3 -> employeeService.updateCustomerDetails();
                case 4 -> employeeService.deleteCustomer();
                case 5 -> employeeService.searchCustomer();
                case 6 -> employeeService.resetCustomerPassword();
                case 7 -> employeeService.viewAllLoans();
                case 8 -> employeeService.approveLoan();
                case 9 -> employeeService.rejectLoan();
                case 10 -> {
                    System.out.print("Enter customer ID: ");
                    int customerId = sc.nextInt();
                    System.out.print("Enter branch ID: ");
                    int branchId = sc.nextInt();
                    System.out.print("Enter account type (Saving/Current/Salary): ");
                    String accountType = sc.next();
                    accountService.addAccount(customerId, branchId, accountType);
                }
                case 11 -> {
                    System.out.print("Enter customer ID: ");
                    int customerId = sc.nextInt();
                    System.out.print("Enter card type (Debit/Credit): ");
                    String cardType = sc.next();
                    cardService.issueCard(customerId, cardType);
                }
                case 12 -> System.out.println("Logging out...");
                default -> System.out.println("Invalid choice!");
            }
        } while (choice != 12);
    }

    private void adminMenu() {
        AdminService adminService = new AdminService();
        BankTransactionService bankTransactionService = new BankTransactionService();
        int choice;

        do {
            System.out.println("\n=== Admin Dashboard ===");
            System.out.println("1. Add New Employee");
            System.out.println("2. Delete Employee");
```

```
            System.out.println("3. View All Employees");
            System.out.println("4. Search Employee");
            System.out.println("5. Update Employee Details");
            System.out.println("6. Reset Employee Password");
            System.out.println("7. View Transaction Report by Account");
            System.out.println("8. Logout");
            System.out.print("Choose: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1 -> adminService.addEmployee();
                case 2 -> adminService.deleteEmployee();
                case 3 -> adminService.viewAllEmployees();
                case 4 -> adminService.searchEmployee();
                case 5 -> adminService.updateEmployeeDetails();
                case 6 -> adminService.resetEmployeePassword();
                case 7 -> {
                    System.out.print("Enter account ID: ");
                    int accId = sc.nextInt();
                    bankTransactionService.viewTransactions(accId);
                }
                case 8 -> System.out.println("Logging out...");
                default -> System.out.println("Invalid choice!");
            }
        } while (choice != 8);
    }
}
```

MainApplication.java

(The MainApplication class serves as the entry point for the banking system, prompting the user to enter their role and loading the appropriate menu based on the role.)

```
import java.util.Scanner;

public class MainApplication {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in); // Open the scanner

        try {
            System.out.println("=== Welcome to Bank System ===");
            System.out.print("Enter your role (customer/employee/admin): ");
            String role = sc.nextLine().toLowerCase();

            // Ensure we pass the role correctly and load the menu
```

```
            MenuService menuService = new MenuService();
            menuService.loadMenu(role);
        } finally {
            // Close the scanner in the finally block to ensure it's done at the end
            sc.close();
        }
    }
}
```

Run.bat

(compiles all Java files in the current directory and runs the MainApplication class using the MySQL JDBC driver for database connectivity.)

```
@echo off
echo Compiling all Java files...
javac -cp ".;D:\DBMS_Project\mysql-connector-j-9.3.0\mysql-connector-j-9.3.0\mysql-connector-j-9.3.0.jar" *.java
echo Compilation complete!

echo Running MainApplication...
java -cp ".;D:\DBMS_Project\mysql-connector-j-9.3.0\mysql-connector-j-9.3.0\mysql-connector-j-9.3.0.jar" MainApplication

pause
```

Output :

```
PS D:\DBMS_Project> cmd /c "d:\DBMS_Project\run.bat"
Compiling all Java files...
Compilation complete!
Running MainApplication...
=== Welcome to Bank System ===
Enter your role (customer/employee/admin): employee

=== Employee Dashboard ===
1. Add Customer
2. View Customer Details
3. Update Customer Details
4. Delete Customer
5. Search Customer
6. Reset Customer Password
7. View All Loans
8. Approve Loan
9. Reject Loan
10. Add Account for Customer
11. Issue Card to Customer
12. Logout
Choose: 1
Enter Customer Name: Disha Mane
Enter Date of Birth (yyyy-mm-dd): 2005-05-03
Enter Gender (Male/Female/Other): Female
Enter Customer Phone: 9080706050
Enter Customer Email: disha@gmail.com
Enter Customer Address: Kapil-Goleshwar
Enter Nationality (default Indian): Indian
Enter Customer Password: Disha123
Loading MySQL JDBC Driver...
Driver loaded successfully!
Customer added successfully!


mysql> select * from customer;
+--+-----------------+--------------+----------+----------------+----------------------+---------------------+-------------+-----------+
|customer_id | name   | dob  | gender | phone        | email                | address   | nationality | password     |
+--+-----------------+--------------+----------+----------------+----------------------+---------------------+-------------+-----------+
| 1|Sneha Jadhav | 2005-08-25|Female|7894560987|sneha@gmail.com| Kapil            |Indian | Sneha@123 |
|2 | Disha Mane  |2005-05-03|Female|9080706050|disha@gmail.com|Kapil-Goleshwar|Indian   | Disha123  |
+--+-----------------+--------------+----------+----------------+----------------------+---------------------+-------------+-----------+
2 rows in set (0.00 sec)
```

=== Employee Dashboard ===
1. Add Customer
2. View Customer Details
3. Update Customer Details
4. Delete Customer
5. Search Customer
6. Reset Customer Password
7. View All Loans
8. Approve Loan
9. Reject Loan
10. Add Account for Customer
11. Issue Card to Customer
12. Logout
Choose: 3
Enter Customer ID to update: 2
Select the detail you want to update:
1. Name
2. Email
3. Phone
4. Address
5. Password
4
Enter new value: Karad
Loading MySQL JDBC Driver...
Driver loaded successfully!
Customer address updated successfully!

mysql> select * from customer;

```
+--+-----------------+--------------+----------+----------------+----------------------+----------------------+-------------
|customer_id | name   | dob  | gender | phone         | email      | address   | nationality | password  |
+--+-----------------+--------------+----------+----------------+----------------------+----------------------+-------------
| 1|Sneha Jadhav|2005-08-25|Female|7894560987|sneha@gmail.com|Kapil |Indian|Sneha@123 |
|2 | Disha Mane  |2005-05-03|Female|9080706050|disha@gmail.com|Karad|Indian   | Disha123  |
+--+-----------------+--------------+----------+----------------+----------------------+----------------------+-------------
```

2 rows in set (0.00 sec)

=== Employee Dashboard ===
1. Add Customer
2. View Customer Details
3. Update Customer Details
4. Delete Customer
5. Search Customer
6. Reset Customer Password
7. View All Loans

8. Approve Loan
9. Reject Loan
10. Add Account for Customer
11. Issue Card to Customer
12. Logout
Choose: 2
Enter Customer ID to view details: 1
Loading MySQL JDBC Driver...
Driver loaded successfully!
Customer ID: 1
Name: Sneha Jadhav
DOB: 2005-08-25
Gender: Female
Phone: 7894560987
Email: sneha@gmail.com
Address: Kapil
Nationality: Indian

=== Employee Dashboard ===
1. Add Customer
2. View Customer Details
3. Update Customer Details
4. Delete Customer
5. Search Customer
6. Reset Customer Password
7. View All Loans
8. Approve Loan
9. Reject Loan
10. Add Account for Customer
11. Issue Card to Customer
12. Logout
Choose: 5
Enter Customer Name or Email to search: Disha Mane
Loading MySQL JDBC Driver...
Driver loaded successfully!
Customer ID: 2
Name: Disha Mane
Email: disha@gmail.com
Phone: 9080706050
Address: Karad

=== Employee Dashboard ===

1. Add Customer
2. View Customer Details
3. Update Customer Details
4. Delete Customer
5. Search Customer
6. Reset Customer Password
7. View All Loans
8. Approve Loan
9. Reject Loan
10. Add Account for Customer
11. Issue Card to Customer
12. Logout
Choose: 7
Loading MySQL JDBC Driver...
Driver loaded successfully!
Loan ID: 1
Loan Amount: 100000.0
Loan Status: Ongoing
---
mysql> select * from loan;

loan_id	customer_id	loan_type	amount	interest_rate	issued_date	due_date	status
1	1	Home	98000.00	8.00	2025-04-30	2026-04-30	Ongoing

1 row in set (0.00 sec)

=== Employee Dashboard ===
1. Add Customer
2. View Customer Details
3. Update Customer Details
4. Delete Customer
5. Search Customer
6. Reset Customer Password
7. View All Loans
8. Approve Loan
9. Reject Loan
10. Add Account for Customer
11. Issue Card to Customer
12. Logout
Choose: 10
Enter customer ID: 2
Enter branch ID: 1
Enter account type (Saving/Current/Salary): Saving

Loading MySQL JDBC Driver...
Driver loaded successfully!
Account added successfully for customer ID: 2

mysql> select * from account;

account_id	customer_id	branch_id	account_type	balance	open_date
1	1	2	Saving	2000.00	2025-04-30
2	2	1	Saving	500.00	2025-05-01

2 rows in set (0.00 sec)

=== Employee Dashboard ===
1. Add Customer
2. View Customer Details
3. Update Customer Details
4. Delete Customer
5. Search Customer
6. Reset Customer Password
7. View All Loans
8. Approve Loan
9. Reject Loan
10. Add Account for Customer
11. Issue Card to Customer
12. Logout
Choose: 11
Enter customer ID: 2
Enter card type (Debit/Credit): Credit
Enter expiry date (yyyy-mm-dd): 2028-10-07
Enter CVV (3 digits): 124
Loading MySQL JDBC Driver...
Driver loaded successfully!
? Card issued successfully with card number: 7160263832480214

mysql> select * from card;

card_id	customer_id	card_type	card_number	expiry_date	cvv
1	1	Credit	2873992147642540	2027-08-25	123
2	2	Credit	7160263832480214	2028-10-07	124

2 rows in set (0.00 sec)

=== Employee Dashboard ===
1. Add Customer
2. View Customer Details
3. Update Customer Details
4. Delete Customer
5. Search Customer
6. Reset Customer Password
7. View All Loans
8. Approve Loan
9. Reject Loan
10. Add Account for Customer
11. Issue Card to Customer
12. Logout
Choose: 12
Logging out...
Press any key to continue . . .

PS D:\DBMS_Project> cmd /c "d:\DBMS_Project\run.bat"
Compiling all Java files...
Compilation complete!
Running MainApplication...
=== Welcome to Bank System ===
Enter your role (customer/employee/admin): customer

=== Customer Dashboard ===
1. View Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Apply for Loan
6. View My Cards
7. View My Loans
8. Repay Loan
9. Check Loan Status
10. Logout
Choose: 1
Enter your customer ID: 2
Loading MySQL JDBC Driver...
Driver loaded successfully!
Account ID: 2 | Type: Saving | Balance: 0.00 | Open Date: 2025-05-01

=== Customer Dashboard ===
1. View Account
2. Deposit Money

3. Withdraw Money
4. Transfer Money
5. Apply for Loan
6. View My Cards
7. View My Loans
8. Repay Loan
9. Check Loan Status
10. Logout
Choose: 4
Enter customer ID: 1
Enter amount to transfer: 500
Enter recipient account number: 2
Transferring $500.0 to account number 2
Loading MySQL JDBC Driver...
Driver loaded successfully!
Transfer successful!

mysql> select * from banktransaction;

transaction_id	account_id	transaction_type	amount	transaction_date
1	1	Credit	1000.00	2025-04-30 23:41:52
2	1	Debit	500.00	2025-04-30 23:42:49
3	1	Debit	500.00	2025-05-01 20:55:20
4	2	Credit	500.00	2025-05-01 20:55:20

4 rows in set (0.00 sec)

=== Customer Dashboard ===
1. View Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Apply for Loan
6. View My Cards
7. View My Loans
8. Repay Loan
9. Check Loan Status
10. Logout
Choose: 7
Enter your customer ID: 2
Loading MySQL JDBC Driver...
Driver loaded successfully!
No loans found for this customer.

```
mysql> select * from loan where customer_id=2;
Empty set (0.00 sec)

=== Customer Dashboard ===
1. View Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Apply for Loan
6. View My Cards
7. View My Loans
8. Repay Loan
9. Check Loan Status
10. Logout
Choose: 8
Enter your customer ID: 1
Enter loan ID to repay: 1
Enter amount to repay: 2000
Loading MySQL JDBC Driver...
Driver loaded successfully!
Repayment successful!
mysql> select * from loan;
+----------+-----------------+-------------+----------+----------------+---------------+------------+---------+
| loan_id | customer_id |loan_type |amount|interest_rate|issued_date|due_date|status  |
+----------+-----------------+-------------+----------+----------------+---------------+------------+---------+
|     1    |        1        | Home     | 98000.00 |8.00 | 2025-04-30 | 2026-04-30 | Ongoing |
+----------+-----------------+-------------+----------+----------------+---------------+------------+---------+
1 row in set (0.00 sec)

mysql> select * from banktransaction where account_id =1;
+------------------+---------------+----------------------+------------+--------------------------+
| transaction_id | account_id | transaction_type | amount  | transaction_date     |
+------------------+---------------+----------------------+------------+--------------------------+
|          1       |       1       | Credit               | 1000.00 | 2025-04-30 23:41:52 |
|          2       |       1       | Debit                |  500.00  | 2025-04-30 23:42:49 |
|          3       |       1       | Debit                |  500.00  | 2025-05-01 20:55:20 |
+------------------+---------------+----------------------+------------+--------------------------+
3 rows in set (0.00 sec)

mysql> select * from banktransaction where account_id =2;
+------------------+---------------+----------------------+------------+--------------------------+
| transaction_id | account_id | transaction_type | amount  | transaction_date     |
+------------------+---------------+----------------------+------------+--------------------------+
|          4       |       2       | Credit               |  500.00  | 2025-05-01 20:55:20 |
```

```
+--------------------+---------------+-----------------------+------------+--------------------------+
1 row in set (0.00 sec)

=== Customer Dashboard ===
1. View Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Apply for Loan
6. View My Cards
7. View My Loans
8. Repay Loan
9. Check Loan Status
10. Logout
Choose: 9
Enter your customer ID: 1
Enter loan ID to check status: 1
Loading MySQL JDBC Driver...
Driver loaded successfully!
Loan Status: Ongoing

=== Customer Dashboard ===
1. View Account
2. Deposit Money
3. Withdraw Money
4. Transfer Money
5. Apply for Loan
6. View My Cards
7. View My Loans
8. Repay Loan
9. Check Loan Status
10. Logout
Choose: 10
Logging out...
Press any key to continue . . .
PS D:\DBMS_Project> cmd /c "d:\DBMS_Project\run.bat"
Compiling all Java files...
Compilation complete!
Running MainApplication...
=== Welcome to Bank System ===
Enter your role (customer/employee/admin): admin

=== Admin Dashboard ===
1. Add New Employee
```

2. Delete Employee
3. View All Employees
4. Search Employee
5. Update Employee Details
6. Reset Employee Password
7. View Transaction Report by Account
8. Logout
Choose: 3
Loading MySQL JDBC Driver...
Driver loaded successfully!
Employee ID: 1
Name: Sanika Dange
Designation: Miss.
Salary: 75000.0
Branch ID: 1
Email: sanika@gmail.com

Employee ID: 2
Name: Pranali Jamadade
Designation: Miss
Salary: 55000.0
Branch ID: 2
Email: pranali@gmail.com
---
mysql> select * from employee;

employee_id	name	designation	salary	branch_id	email	password
1	Sanika Dange	Miss.	75000.00	1	sanika@gmail.com	Sanu123
2	Pranali Jamadade	Miss	55000.00	2	pranali@gmail.com	Pranali123

2 rows in set (0.00 sec)

=== Admin Dashboard ===
1. Add New Employee
2. Delete Employee
3. View All Employees
4. Search Employee
5. Update Employee Details
6. Reset Employee Password
7. View Transaction Report by Account
8. Logout
Choose: 7
Enter account ID: 1

Loading MySQL JDBC Driver...
Driver loaded successfully!
Transaction ID: 3 | Type: Debit | Amount: 500.00 | Date: 2025-05-01 20:55:20.0
Transaction ID: 2 | Type: Debit | Amount: 500.00 | Date: 2025-04-30 23:42:49.0
Transaction ID: 1 | Type: Credit | Amount: 1000.00 | Date: 2025-04-30 23:41:52.0

=== Admin Dashboard ===
1. Add New Employee
2. Delete Employee
3. View All Employees
4. Search Employee
5. Update Employee Details
6. Reset Employee Password
7. View Transaction Report by Account
8. Logout
Choose: 7
Enter account ID: 2
Loading MySQL JDBC Driver...
Driver loaded successfully!
Transaction ID: 4 | Type: Credit | Amount: 500.00 | Date: 2025-05-01 20:55:20.0

=== Admin Dashboard ===
1. Add New Employee
2. Delete Employee
3. View All Employees
4. Search Employee
5. Update Employee Details
6. Reset Employee Password
7. View Transaction Report by Account
8. Logout
Choose: 8
Logging out...
Press any key to continue . . .
PS D:\DBMS_Project>