

BANK DATABASE MANAGEMENT SYSTEM

SUBMITTED BY

NAME

USN

Mihika Dhariwal

1MS21CS075

As part of the Course **Database Systems – CS53**

SUPERVISED BY

Faculty

Dr. Dayananda R.B.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute, Affiliated to VTU)

Bangalore – 560054

Department of Computer Science and Engineering

Ramaiah Institute of Technology

(Autonomous Institute, Affiliated to VTU)

Bangalore – 560054



CERTIFICATE

This is to certify that **Mihika Dhariwal** has completed the Database system Project report on **Bank Database Management System**. We declare that the entire content embodied in this B.E. 5th Semester report contents are not plagiarized.

Submitted by

Mihika Dhariwal

1MS21CS075

Guided by

Dr. Dayananda R.B.

Dept of CSE, RIT

Evaluation Sheet

SI. No.	Name	USN	Research Content Understanding and Coding (10)	Demo and Report Submission (10)	Total Marks (20)
1	Mihika Dhariwal	1MS21CS075			

Evaluated By:

Dr. Dayananda R.B.

Faculty
Computer Science and Engineering, RIT

TABLE OF CONTENTS

		Page No
1.	Abstract	4
2.	Introduction	
	2.1 Problem statement	5
	2.2 Objectives	5
	2.3 Proposed Solution	5
3.	Methodology	6
4.	Requirements	
	4.1 Software requirements	8
	4.2 Hardware Requirements	8
	4.3 Functional Requirements	8
	4.4 Non-functional Requirements	10
5.	Relational Database Design	11
	5.1 ER Diagrams	12
6.	Data Normalization	13
7.	Data Directory	14
8.	Sample Source Code	15
9.	Graphical User Interface	22
10.	Conclusion	28
11.	References	29

1. ABSTRACT

The Bank Database Management System (BDMS) is a comprehensive software solution designed to cater to the diverse needs of both customers and administrators within the banking domain. Customers can efficiently create accounts, access account information including balances and transaction histories, conduct transactions such as transfers, deposits, and withdrawals, and apply for loans with the added benefit of a simple interest calculator for informed financial planning. On the administrative front, the system allows administrators to oversee the entire database, access specific account details, and efficiently manage accounts, ensuring a secure and streamlined banking experience for all stakeholders involved. BDMS serves as an integrated solution, offering a user-friendly interface for customers to manage their accounts and conduct financial transactions, and providing administrators with robust tools for database oversight and account management, thereby contributing to an efficient and secure banking environment.

2. INTRODUCTION

2.1 Problem Statement

The current banking infrastructure faces difficulties in managing and integrating diverse banking operations while providing excellent customer service. Existing systems do not integrate seamlessly, resulting in inefficiencies in transaction processing, customer data management, and timely decision-making. Furthermore, the lack of a unified platform makes it difficult to provide personalized services to customers. This Bank Management System seeks to address these issues by providing a strong, integrated solution that streamlines banking processes, centralizes data management, and improves customer experiences.

2.2 Objectives

1. **Efficient Account Management:** Streamline account creation, maintenance, and user authentication processes for customers.
2. **Enhanced User Experience:** Provide a user-friendly interface for customers to access and manage their accounts seamlessly.
3. **Comprehensive Financial Management:** Enable customers to perform various financial transactions such as transfers, deposits, withdrawals, and loan applications.
4. **Financial Planning Tools:** Offer simple interest calculators to aid customers in making informed financial decisions.
5. **Administrative Oversight:** Grant administrators' comprehensive oversight of the database, facilitating efficient account management and data control.
6. **Operational Efficiency:** Improve operational processes for both customers and administrators within the banking environment.

2.3 Proposed Solution

The proposed solution is a Java and MySQL-based banking platform that encompasses features to improve user experiences for both customers and administrators. For the customer end, the system will introduce an intuitive and user-friendly interface. Customers can seamlessly execute deposit and withdrawal transactions, apply for loans, perform transfers between accounts, and leverage an interest calculator to make informed financial decisions. The user interface will prioritize simplicity and responsiveness to ensure a positive and efficient customer experience. On the admin end, the platform will introduce a comprehensive dashboard displaying all registered accounts, including relevant details such as account balances. Admins will have the ability to efficiently review, approve, or reject loan applications through a streamlined workflow, reducing manual intervention and processing times. To improve overall system performance and scalability, the solution will leverage advanced database optimization techniques and potentially explore the integration of cloud-based services.

3. METHODOLOGY

The development of the Bank Database Management System (BDMS) adopts a systematic methodology, integrating Java Servlets, JSP, and SQL within the XAMPP environment. The outlined methodology delineates key steps in designing, implementing, and testing the system:

Requirements Analysis:

- Engaged in exhaustive discussions with stakeholders to define functional and non-functional requirements.
- Identified user stories and use cases to capture specific functionalities and interactions within the banking context.

System Design:

- Developed a high-level system architecture, illustrating main components and their interactions within the Java Servlets and JSP framework.
- Designed the SQL database schema, identifying entities, relationships, and attributes relevant to banking operations.
- Created wireframes and mockups for the user interface, ensuring an intuitive design for both customers and administrators.

Technology Stack Selection:

- Selected Java Servlets and JSP for web application development due to their compatibility and efficiency.
- Utilized XAMPP as the server environment, incorporating Apache Tomcat for Servlet execution, and MySQL for database management.

Implementation:

- Adhered to the Model-View-Controller (MVC) architecture within the Java Servlets and JSP framework for organized and maintainable code structure.
- Implemented user authentication and authorization features using Java Servlets and JSP.
- Developed servlets and JSP pages for account management, transaction processes, and user interactions.
- Implemented CRUD operations for the database tables

Account and Transaction Management:

- Implemented logic for adding, updating, and managing account information.
- Integrated functionalities for customers to perform transactions (Deposit, Withdraw, Transfer, Request Loans), check balances, and manage their accounts.

User Interface Development:

- Leveraged JSP pages for creating dynamic and responsive user interfaces.

Security Implementation:

- Implemented input validation within servlets and JSP to prevent SQL injection and other security vulnerabilities.

Testing:

- Conducted unit tests for each component of the system to ensure functionality and identify potential bugs.
- Performed integration testing to evaluate the interactions between different modules.

Documentation:

- Created comprehensive documentation, including a project overview, database schema documentation, and a guide for setting up and running the application.
- Documented the codebase to facilitate future maintenance and development.

Scalability Considerations:

- Designed the system with scalability in mind, considering potential areas for optimization and expansion.
- Evaluated the system's performance under different load conditions when hosted on XAMPP with Apache Tomcat.

This methodology ensures a structured and systematic approach to the development of the Bank Database Management System, emphasizing best practices and established frameworks within the Java Servlets, JSP, and SQL environment.

4. REQUIREMENTS

4.1 Software Requirements:

Frontend – HTML, CSS, JavaScript

Backend – Java Servlets, MySQL

- Operating System: Windows 10
- Google Chrome/Internet Explorer
- XAMPP (Version-3.7)
- MySQL (Version 8.0)
- Java editor: Eclipse EE

4.2 Hardware Requirements:

- Computer with a 1.1 GHz or faster processor
- Minimum 2GB of RAM or more
- 2.5 GB of available hard-disk space
- 5400 RPM hard drive
- 1366 × 768 or higher-resolution display
- DVD-ROM drive

4.3 Functional Requirements

GUI:

The GUI designed for the Bank Management System ensures an intuitive and accessible experience catering to both customers and administrators. It provides two distinct modes - a customer-oriented view tailored for regular users and an administrative view dedicated to system management. The GUI has a user-friendly layout enabling easy navigation and interaction. For customers, it offers a dashboard facilitating essential banking operations such as deposit, withdrawal, and funds transfer, alongside account management tools, loan request features, and comprehensive access to personal and account details. Administrators benefit from access to detailed customer information, account data, and authorization capabilities for loan requests. Moreover, the interface incorporates robust error handling mechanisms, promptly alerting users to incorrect inputs or system errors, thus guiding them through seamless interactions.

Customer/User Management:

User Registration: Enables new users to sign up by providing necessary personal details and creating login credentials.

User Login: Allows registered users to access the system by entering their username and password.

User Authentication: Verifies user identity through secure authentication methods to prevent unauthorized access.

User Roles and Permissions: Assigns specific roles (e.g., customer, admin) and defines access levels for different system modules.

Account Management:

Account Creation: Facilitates the setup of new bank accounts, assigning unique account numbers for each user.

Transaction Processing:

Deposit: Enables users to add funds to their accounts through various channels (e.g., cash, check, online transfer).

Withdrawal: Allows users to withdraw funds from their accounts, specifying the amount and withdrawal method.

Funds Transfer: Permits the transfer of funds between accounts, either internally or to external accounts.

Transaction History: Records and displays a chronological list of all transactions.

Loan Management:

Loan Application Submission: Allows customers to request for loans by specifying the loan amount, loan type and duration.

Loan Approval Process: Facilitates the review and approval of loan applications by administrators, considering eligibility criteria and risk assessment.

Administrative Functions:

Admin Registration: Enables system administrators to create their accounts with appropriate privileges.

Admin Login: Grants access to the administrative dashboard

Customer Details and Account Information View: Allows the administrator to view the personal details and account information of all registered users.

Loan approval: Allows the approval or rejection of loan requests sent by users

Error Handling:

Error Messages: System-generated, descriptive prompts conveying encountered issues, guiding users on rectification steps.

Handling Invalid Inputs or Incorrect Actions: Validation checks prompt users to rectify errors or guide correct data entry, maintaining system integrity.

4.4 Non-Functional Requirements

Response Time: Upon initiating a transaction or service request, users should receive acknowledgment or necessary follow-up within a specified time frame, ensuring prompt response and service completion.

Throughput: The system's transactional capacity should scale with the number of concurrent users, accommodating various user types, including customers and bank employees. It should efficiently handle the diverse range of transactions and user interactions within the system.

Capacity: The system must support a large user base, capable of concurrently serving more than 10,000 users to ensure smooth and uninterrupted operations during peak periods without performance degradation.

Resource Utilization: The system's resources, including computational power, storage, and network bandwidth, should be optimized based on user demands and service requirements. Adjustments in resources must align with varying user needs and service requests to maintain efficiency and effectiveness.

Reliability: The system emphasizes the application of advanced engineering practices and specialized techniques to prevent or mitigate system failures. A dedicated team is in place to identify, rectify, and analyze causes of system failures, aiming to ensure high reliability, minimize downtime, and optimize system performance.

Security: As a banking system, robust security measures are paramount. The system ensures strict access control, encryption standards, and secure authentication protocols to safeguard sensitive financial data and transactions, maintaining confidentiality and integrity.

Maintainability: While the software itself requires no maintenance, the system allows for seamless integration with the end user's-maintained database, ensuring compatibility, data integrity, and efficient management by the user.

Portability: The developed software should be compatible across various platforms, enabling smooth operations on different systems without compatibility issues. Additionally, the system code will be designed to be easily translatable into other programming languages, ensuring adaptability and portability for future enhancements or system migrations.

5. RELATIONAL DATABASE DESIGN

Relational database design refers to the process of structuring data in a way that facilitates efficient storage, retrieval, and management of information in a relational database management system (RDBMS). The goal of relational database design is to organize data into tables and establish relationships between these tables to represent the logical structure of the information. Relational database design follows principles known as the normalization rules, which help create well-structured databases that minimize data redundancy and dependency. It aims to ensure data consistency, integrity, and ease of maintenance.

In the Bank Database Management System, the relational database design follows normalization principles to ensure a well-structured database with minimized data redundancy and dependency. The relational model, based on the principles introduced by E.F. Codd, forms the theoretical foundation for this database design. Here are examples of foreign key relationships in your database tables:

Customer Table:

- Email references cemail in Account table
- Email references User_email in Loans table

Account Table

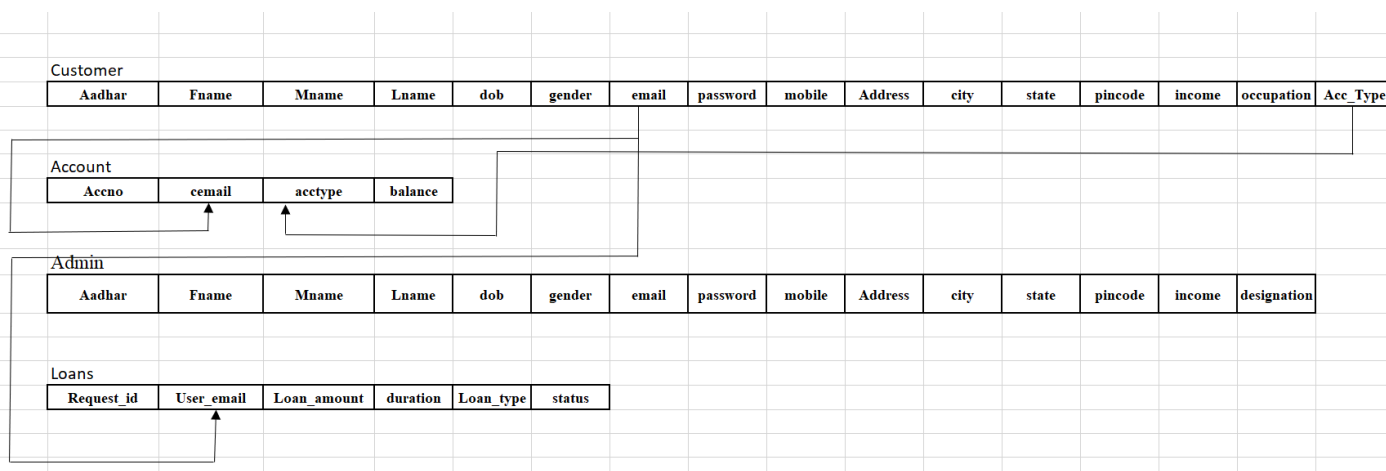
- No explicit foreign keys in the provided schema.

Admin Table

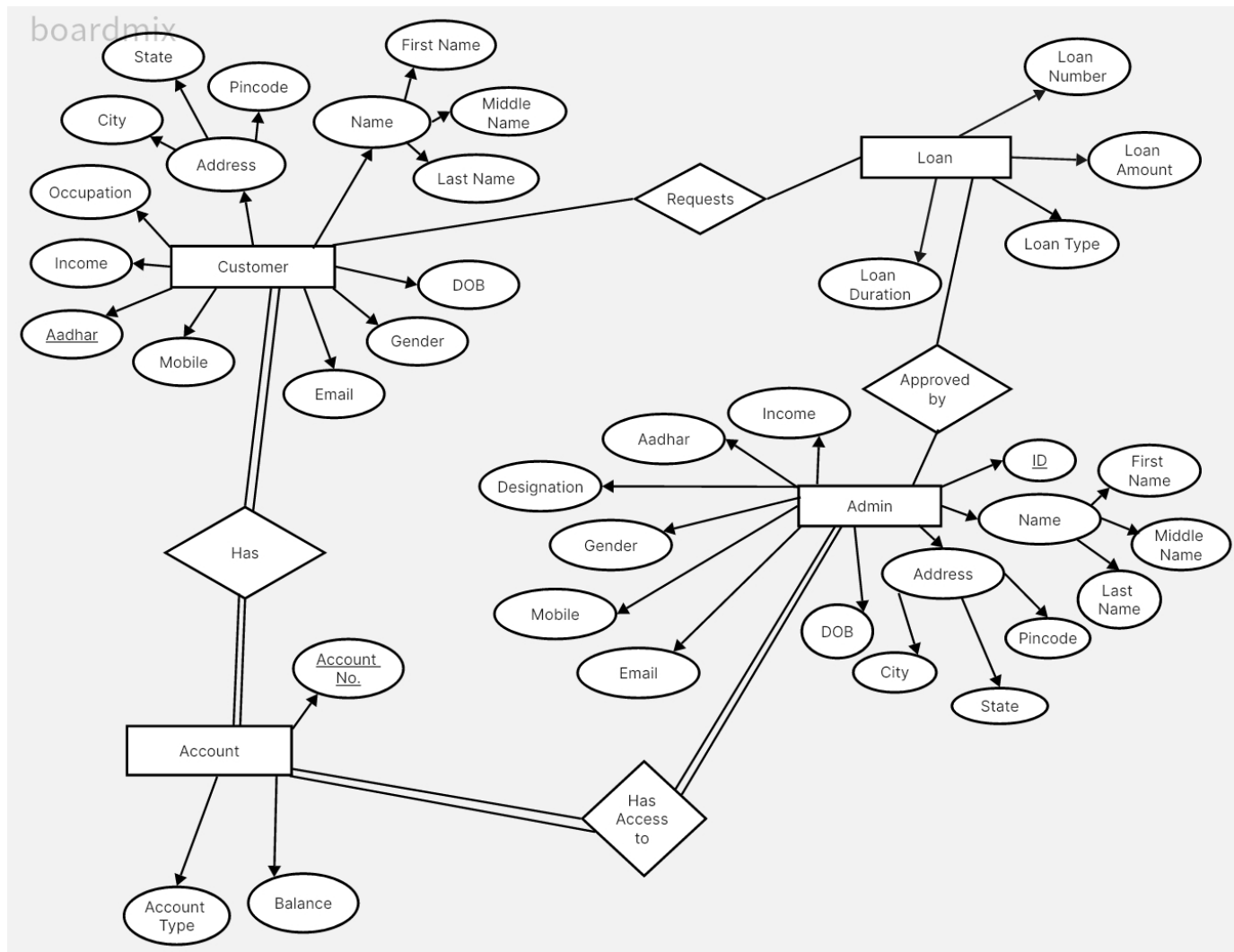
- No explicit foreign keys in the provided schema

Loan Table

- No explicit foreign keys in the provided schema.



5.1 ER Diagram



6. DATA NORMALISATION

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. The normalization process typically involves dividing large tables into smaller, more manageable tables and defining relationships between them.

A possible normalization for Bank Database Management System:

First Normal Form (1NF): Ensure all columns contain atomic values (no repeating groups or arrays). Eliminate any redundant data.

Adjustment: No adjustment needed as the initial tables are already in 1NF.

Second Normal Form (2NF): Meet the requirements of 1NF. Remove partial dependencies by creating separate tables for sets of attributes that functionally depend on a part of the primary key.

In the account table, candidate key is (Accno, cemail). Both Accno and cemail determines non prime attribute acctype, making it a partial dependancy.

Adjustment: Split the account table into 2 relations, the first one containing attributes (Accno, acctype, balance) and the second one containing attributes (cemail, acctype and balance).

Third Normal Form (3NF): Meet the requirements of 2NF. Eliminate transitive dependencies.

Adjustment: No adjustment needed as the tables are already in 3NF.

7. DATA DIRECTORY

In the context of database systems and management, a data directory refers to a directory or folder that stores metadata and control information about the database. This directory is crucial for the proper functioning and organization of the database system. It involves:

- **Metadata Storage:** The data directory contains metadata, which includes information about the structure of the database, such as tables, indexes, views, and other database objects.
- **Control Files:** Database systems often use control files that store configuration settings and parameters. These files are part of the data directory and help manage aspects such as transaction logs, data file locations, and other system-related configurations.
- **Database Configuration:** The data directory may include files that store configuration information for the entire database system, including details about users, access permissions, and other global settings.
- **Logging and Recovery Information:** In some database systems, the data directory may also include logs and information needed for recovery in case of system failures. This ensures data consistency and durability.
- **Security Information:** Database security information, such as user credentials and access control lists, may be stored within the data directory to manage and enforce security policies.

The Data Directory is as follows:

```
├── main
│   ├── java
│   │   ├── adminloginservlet.java
│   │   ├── depositServlet.java
│   │   ├── loanServlet.java
│   │   ├── loginServlet.java
│   │   ├── transferServlet.java
│   │   ├── withdrawServlet.java
│   │   ├── lrServlet.java
│   │   ├── regadminservlet.java
│   │   └── registerservlet.java
│   ├── webapp
│   │   ├── adminlogin.jsp
│   │   ├── adminwelcome.jsp
│   │   ├── index.jsp
│   │   ├── login.jsp
│   │   ├── regadmin.jsp
│   │   └── images
│   │       ├── pic1.jpg
│   │       ├── pic2.jpg
│   │       ├── pic3.jpg
│   │       └── pic4.jpg
│   ├── register.jsp
│   ├── welcome.jsp
│   ├── adminlogin.jsp.css
│   ├── adminwelcome.css
│   ├── index.css
│   ├── login.css
│   ├── regadmin.css
│   ├── register.css
│   ├── welcome.css
│   └── WEB-INF
│       └── web.xml
```

8. SAMPLE SOURCE CODE

Register Servlet

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

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class registerservlet
 */
@WebServlet("/registerservlet")
public class registerservlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        String fname = request.getParameter("fname");
        String mname = request.getParameter("mname");
        String lname = request.getParameter("lname");
        String dob = request.getParameter("dob");
        String gender = request.getParameter("gender");
        String email = request.getParameter("email");
        String password = request.getParameter("password");
        String mobile = request.getParameter("mobile");
        String address = request.getParameter("address");
        String city = request.getParameter("city");
        String state = request.getParameter("state");
        String pin = request.getParameter("pin");
        String income = request.getParameter("income");
        String occupation = request.getParameter("occupation");
        String aadhar = request.getParameter("aadhar");
        RequestDispatcher dispatcher = null;
        Connection con=null;
        Random random = new Random();
        int accountNumber = 100000 + random.nextInt(900000);

        request.setAttribute("accountNumber", String.valueOf(accountNumber));

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            con=DriverManager.getConnection("jdbc:mysql://localhost:3307/banking?useSSL=false", "root", "mihika123");
            PreparedStatement pst = con.prepareStatement("insert into customer values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
            pst.setString(1, aadhar);
            pst.setString(2, fname);
            pst.setString(3, mname);
            pst.setString(4, lname);
            pst.setString(5, dob);
            pst.setString(6, gender);
            pst.setString(7, email);
            pst.setString(8, password);
            pst.setString(9, mobile);
            pst.setString(10, address);
            pst.setString(11, city);
            pst.setString(12, state);
            pst.setString(13, pin);
            pst.setString(14, income);
            pst.setString(15, occupation);
```



```

int rowCount = pst.executeUpdate();
dispatcher = request.getRequestDispatcher("register.jsp");
if(rowCount > 0) {
    request.setAttribute("status", "success");
}
else {
    request.setAttribute("status", "failed");
}
dispatcher.forward(request, response);

```

```

PreparedStatement accountPst = con.prepareStatement("INSERT INTO account (accno, cemail, acctype, balance) VALUES (?, ?, ?, ?)");

```

```

    accountPst.setInt(1, accountNumber);
    accountPst.setString(2, email);
    accountPst.setString(3, "Savings Account");
    accountPst.setDouble(4, 0.0);

```

```

    accountPst.executeUpdate();

```

```

} catch (ClassNotFoundException e) {
    e.printStackTrace();

```

```

} catch (SQLException e) {
    e.printStackTrace();

```

```

}finally {

```

```

try {

```

```

    if (con != null) {
        con.close();
    }

```

```

} catch (SQLException e) {
    e.printStackTrace();
}

```

```

}

```

```

}

```

Login Servlet

```

import java.io.IOException;
import java.sql.DriverManager;
import java.sql.*;

```

```

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

```

```

/**

```

```

 * Servlet implementation class loginservlet

```

```

 */

```

```

@WebServlet("/loginservlet")

```

```

public class loginservlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

```

```

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String email = request.getParameter("email");
        String pwd = request.getParameter("password");

```

```

    try {

```

```

        Class.forName("com.mysql.cj.jdbc.Driver");

```

```

        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3307/banking?useSSL=false", "root", "mihika123");

```

```

        PreparedStatement pst = con.prepareStatement("select * from customer where email = ? and password=?");

```

```

pst.setString(1, email);
pst.setString(2, pwd);

ResultSet rs = pst.executeQuery();

if(rs.next()) {

String userEmail = rs.getString("email");
request.setAttribute("email", userEmail );
String firstName = rs.getString("fname");
request.setAttribute("firstName", firstName);

double accountBalance = getBalancefromDatabase(email);

request.setAttribute("balance", accountBalance);

RequestDispatcher dispatcher = request.getRequestDispatcher("welcome.jsp");
dispatcher.forward(request, response);

}
else {
request.setAttribute("errorMessage", "Invalid credentials. Please try again.");
RequestDispatcher dispatcher = request.getRequestDispatcher("login.jsp");
dispatcher.forward(request, response);
}
} catch(Exception e) {
e.printStackTrace();
}
}

protected double getBalancefromDatabase(String userEmail) {
double balance = 0.0;

try {
Class.forName("com.mysql.cj.jdbc.Driver");
Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3307/banking?useSSL=false", "root", "mihika123");
PreparedStatement pst = con.prepareStatement("select * from account where cemail = ?");
pst.setString(1, userEmail);
ResultSet resultSet = pst.executeQuery();
    if(resultSet.next()) {
        balance = resultSet.getDouble("balance");
    }
    resultSet.close();
    pst.close();
    con.close();

}
catch(Exception e) {
e.printStackTrace();
}
return balance;
}

}

```

Register JSP

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>
<html>
<head>
    <meta charset="ISO-8859-1">
    <title>register</title>
    <link rel="stylesheet" href="reg.css">
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Comfortaa">
    <script src="https://kit.fontawesome.com/b59c55b5a0.js" crossorigin="anonymous"></script>
    <script
src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
    <link rel="stylesheet"
href="https://unpkg.com/sweetalert/dist/sweetalert.css">
    <script>
        function validateForm() {

            var dob = new Date(document.getElementById("dob").value);
            var currentDate = new Date();

            if (dob >= currentDate) {
                alert("Date of Birth should be before the current date.");
                return false;
            }

            var mobile = document.getElementById("mobile").value;
            if (!/^\\d{10}$/.test(mobile)) {
                alert("Please enter a valid 10-digit mobile number.");
                return false;
            }

            var pin = document.getElementById("pin").value;
            if (!/^\\d{6}$/.test(pin)) {
                alert("Please enter a valid 6-digit PIN.");
                return false;
            }

            var city = document.getElementById("city").value;
            var state = document.getElementById("state").value;
            if (!/^\\[A-Za-z]+$/\\.test(city) || !/^\\[A-Za-z]+$/\\.test(state)) {
                alert("Invalid Entry");
                return false;
            }

            var password = document.getElementById("password").value;
            if (password.length < 8 || !/^\\[a-z]/\\.test(password) || !/^\\[A-Z]/\\.test(password)) {
                alert("Password must contain at least 8 characters, including one lowercase and one uppercase letter.");
                return false;
            }

            var pin = document.getElementById("aadhar").value;
            if (!/^\\d{12}$/.test(pin)) {
                alert("Please enter a valid 12-digit Aadhar number.");
                return false;
            }

            return true;
        }
    </script>
</script>
```

```

document.addEventListener("DOMContentLoaded", function() {
var status = document.getElementById("status").value;
var accountNumber = document.getElementById("accountNumber").value;

if (status === "success" && accountNumber !== null && accountNumber !== "") {
swal("Congrats", "Account created successfully. Your account number is: " + accountNumber, "success");
} else if (status === "failed") {
swal("Oops", "Registration failed", "error");
}
});

</script>

</head>
<body>

<input type="hidden" id="status" value="<%= request.getAttribute("status") %>">
<input type="hidden" id="accountNumber" value="<%= request.getAttribute("accountNumber") %>">

<div class="container">
<h1>Registration</h1>
<form action="registerservlet" method="post" onsubmit="return validateForm()">
<!-- Replace 'submit_registration.php' with your form processing script -->
<div class="form-group name">
<div class="fname">
<label for="name">First Name:</label>
<input type="text" id="fname" name="fname" required>
</div>
<div class="mname">
<label for="name">Middle Name:</label>
<input type="text" id="mname" name="mname">
</div>
<div class="lname">
<label for="name">Last Name:</label>
<input type="text" id="lname" name="lname" required>
</div>
</div>
<div class="form-group">
<label for="dob">Date of Birth:</label>
<input type="date" id="dob" name="dob" required>
</div>
<div class="form-group">
<label>Gender:</label>
<div class="gender-options">
<input type="radio" id="male" name="gender" value="male">Male
<input type="radio" id="female" name="gender" value="female">Female

</div>
</div>
<div class="form-group">
<label for="email">Email:</label>
<input type="email" id="email" name="email" required>
</div>
<div class="form-group">
<label for="password">Password:</label>
<input type="password" id="password" name="password" required>
</div>

<div class="form-group">
<label for="mobile">Mobile:</label>
<input type="tel" id="mobile" name="mobile" required>
</div>

<div class="form-group">
<label for="address">Address:</label>
<textarea id="address" name="address" required></textarea>
</div>

```

```

<div class="form-group addr">
  <div class="city">
    <label for="city">City</label>
    <input type="text" id="city" name="city" required>
  </div>
  <div class="state">
    <label for="name">State</label>
    <input type="text" id="state" name="state" required>
  </div>
  <div class="pin">
    <label for="name">Pin</label>
    <input type="text" id="pin" name="pin" required>
  </div>
</div>

<div class="form-group">
  <label for="income">Income:</label>
  <select id="income" name="income" required style="width: 100%; padding: 8px; border-radius: 5px;
border: 1px solid #ccc;">
    <option value="">Select Income</option>
    <option value="0-50000">0 - 50,000</option>
    <option value="50001-100000">50,001 - 100,000</option>
    <option value="100001-150000">100,001 - 150,000</option>
    <option value="150001-200000">150,001 - 200,000</option>
    <option value="200001+">200,001 and above</option>
  </select>
</div>
<div class="form-group">
  <label for="occupation">Occupation:</label>
  <input type="text" id="occupation" name="occupation" required>
</div>
<div class="form-group">
  <label for="aadhar">Aadhar Number:</label>
  <input type="text" id="aadhar" name="aadhar" required>
</div>
<div class="buttons">
  <button class="cust" type="submit">Register</button>
  <button class="admin"><a href="regadmin.jsp" style="text-decoration: none; color: white;">Register as Admin</a> </button>
</div>
</form>
</div>

</body>
</html>

```

Login JSP

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>

<head>
    <title>Login Page</title>
    <link rel="stylesheet" href="login.css">
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Comfortaa">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">

</head>

<body>
    <div class="container">
        <div class="content">
            <h1>PayCom.</h1>
            <h3>Hello! Let's get started</h3>
            <p class="login">Login to your account</p>
            <form id="loginForm" method="post" action = "loginservlet">
                <div class="form-group">
                    <i class="fas fa-envelope"></i>
                    <input type="email" id="email" name="email" placeholder="Email" required>
                </div>
                <div class="form-group">
                    <i class="fas fa-lock"></i>
                    <input type="password" id="password" name="password" placeholder="Password" required>
                </div>

                <button type="submit">Login</button>
            </form>

            <!-- Display error message here -->
            <% String errorMessage = (String) request.getAttribute("errorMessage");
            if (errorMessage != null && !errorMessage.isEmpty()) { %>
                <p style="color: red;"><%= errorMessage %></p>
            <% } %>

            <p class="question">Don't have an account? <span> <a href="register.jsp">SIGN UP</a></span> here</p>
            <p class="admin">Login to <span><a href="adminlogin.jsp">ADMIN</a></span> account</p>
        </div>
        <div class="image">
            
        </div>
    </div>

</body>

</html>
```

9. GRAPHICAL USER INTERFACE

Customer Login Page

PayCom.


Hello! Let's get started

Login to your account

Login

Don't have an account? [SIGN UP](#) here

Login to **ADMIN** account



Admin Login Page

Admin Login


Hello! Let's get started

Login to your account

Login

Don't have an account? [SIGN UP](#) here

Login to **CUSTOMER** account



Customer Registration Page

Registration

First Name:

Middle Name:

Last Name:

dd-mm-yyyy

Gender:

Male

Female

Email:

Password:

Mobile:

Address:

City

State

Pin

Income:

Select Income

Occupation:

Aadhar Number:

Register

Register as Admin

Admin Registration Page

Register as Admin

First Name:

Middle Name:

Last Name:

dd-mm-yyyy

Gender:

Male

Female

Email:

Password:

Mobile:

Address:

City

State

Pin

Income:

Select Income

Designation:


Select Designation

Aadhar Number:


Register

Customer Dashboard


Welcome, Mihika!




Deposit




Withdraw



Transfer



Loan



Balance:

Rs 11000.0

Simple Interest Calculator

Principal amount:

Annual interest rate (%):


Time period (years):

Calculate

Simple Interest: 80

Deposit Window

Deposit



Deposit Amount:

Deposit

Deposit successful!

Withdraw Window

Withdraw

Withdraw Amount:

Withdraw

Transfer Window

Transfer

Transfer Amount:

Account No:

Transfer

Loan Request Window

Loan

Loan Amount:

Enter amount to be loaned

Duration

Enter duration (in years)

Select loan type:


Student Loan

Request Loan


Admin Dashboard

Welcome, zz!


Welcome Admin!




View Customer Details



View Account Details



View Loan Requests



Customer Details Window

Customer Details														
Aadhar	First Name	Middle Name	Last Name	Date of Birth	Gender	Email	Password	Mobile	Address	City	State	PIN	Income	Occupation
666666666666	Mihika		Dhariwal	2023-12-16	female	m@gmail.com	Mihika1234	1111111111	xxx	x	x	777777	150001-200000	Student

Account Details Window

Account Details			
Account No.	Email	Account Type	Balance
909550	m@gmail.com	Savings Account	12000.00

Loan Requests Window

Loan Requests						
Request ID	Email	Loan Amount	Duration	Loan Type	Status	Action
23	m@gmail.com	1000.00	3	student	Accepted	<div>AcceptReject</div>

10. CONCLUSION

The Bank Management System has been successfully implemented, leveraging Java Servlets, JSP, and SQL within the XAMPP environment. This robust web application empowers users to register, log in, perform transactions, and access personalized banking services, while administrators can efficiently manage customer accounts and financial information.

The system revolves around four core tables - customers, accounts, admin and loan requests - establishing a well-structured relational database with a focus on data consistency. The choice of MySQL as the database management system was driven by its widespread usage and adaptability, ensuring streamlined data management capabilities.

This project has been instrumental in providing practical insights, underscoring the significance of meticulous planning, structured methodologies, and collaborative teamwork. It has significantly enhanced our proficiency in both theoretical concepts and their practical application, contributing to a holistic comprehension of project development principles in the context of a bank management system.

11. REFERENCES

<https://www.w3schools.com/MySQL/default.asp>

<https://www.javatpoint.com/jsp-tutorial>

<https://www.javatpoint.com/servlet-tutorial>

<https://www.scaler.com/topics/er-diagram-for-bank-database/>

<https://www.geeksforgeeks.org/er-diagram-of-bank-management-system/>