Plot No.2, Sector 17-A, Yamuna Expressway, Greater Noida, Gautam Buddha Nagar, U.P., India

Bank Management System using Java and MySQL

Project Report

Team: Vintage Saga

B.Tech. (***Third Sem.2024-2025***)

Submitted by:

Kartikey Tiwari
Garvit Tyagi
Ravikant Tyagi
Aseem Shukla

Report on Online Banking System in Java

1. Introduction

The online banking system is a web-based application that enables users to perform banking operations through the internet. It eliminates the need for customers to physically visit a bank and allows them to manage their accounts, transfer funds, view transaction history, and perform other financial activities from anywhere. This project, implemented in Java, demonstrates core banking functionalities with security and ease of access.

2. Objective

The primary objective of this project is to provide users with a platform for convenient and secure management of their bank accounts. The system includes features such as account management, fund transfers, transaction history, balance inquiries, and user authentication, with a focus on providing a user-friendly interface and data security.

3. Technologies Used

- Java (JDK): Core programming language for developing the system's logic.
- JSP/Servlets: For dynamic web pages and server-side processing.
- HTML/CSS: To create the front-end user interface.
- MySQL: To store user data, account details, and transaction logs.
- Apache Tomcat: As the web server to deploy and run the application.
- JDBC: For database connectivity between the Java application and MySQL.
- Spring/Spring Boot (optional): For implementing modern Java web frameworks and enhancing the system's scalability and security.

4. System Architecture

The system follows a three-tier architecture:

1. Presentation Layer: Provides the user interface, where customers interact with the system via forms and buttons to perform banking operations.

2. Business Logic Layer: Contains the core Java code that processes requests from the user, performs validations, and sends the required data to the database.

3. Data Layer: Interacts with the database to retrieve, update, or store information, such as account details and transaction logs.

5. Features and Functionalities

The main features of the online banking system:

5.1 User Authentication

- Users must log in with their unique credentials (username and password).

- Password encryption ensures secure login.

5.2 Account Management

- Account balance and status are displayed on the dashboard.

5.3 Fund Transfers

- Users can transfer funds between their own accounts or to third-party accounts.

- The system checks for sufficient balance before processing transfers.

- Transaction details are recorded in the database for future reference.

5.4 Transaction History

- Users can view their transaction history, including deposits, withdrawals, and transfers.

- The system provides filtering options, such as date ranges or transaction types, for easy search.

5.5 Balance Inquiry

- Users can check their current balance, recent transactions, and account status on the dashboard.

5.6 Security Features

- SSL encryption to protect sensitive data during transmission.

- Role-based access control to prevent unauthorized actions (e.g., admin vs. User roles).

- Secure password storage using hashing techniques.

6. Database Schema

The database consists of the following tables:

- **Users**: Stores user details, login credentials, and personal information.

- **Accounts**: Contains account-related information such as account number, balance, and account type.

- **Transactions**: Records every transaction, including the sender’s and recipient’s details, amount, and timestamp.

7. Login Module

This module handles user authentication. It uses the Java `JDBC` library to connect to the MySQL database and verify login credentials. Successful login grants access to the user’s dashboard, while incorrect credentials prompt an error message.

7.1 Account Management Module

Users can update and view their account balance. This module retrieves and displays data from the `Accounts` table and ensures secure updates.

7.2 Transaction Module

The transaction module allows users to transfer funds. It checks for sufficient balance and updates the `Transactions` table upon successful transfer. It also handles error cases, such as invalid recipient account numbers or insufficient funds.

7.3 Transaction History Module

This module fetches transaction data from the `Transactions` table based on user queries (e.g., recent transactions, date range filters). It displays the transaction type, date, and amount.

8. Challenges and Solutions

8.1 Security

One of the major challenges was ensuring data security. To tackle this, encryption techniques (e.g., password hashing, SSL) were employed. Additionally, access control and input validation helped mitigate common security risks such as SQL injection.

8.2 Concurrency

Handling simultaneous transactions without data conflicts was essential. Implementing **transaction management** using Java’s `synchronized` keyword and MySQL transaction isolation levels ensured data consistency.

9. Testing

The system underwent thorough testing, including:

- **Unit Testing:** Individual components like login, fund transfers, and database connectivity were tested using Junit.

- **Integration Testing:** Ensured that all modules work together smoothly.

10. Future Enhancements

- **Mobile App Integration**: A mobile version for Android or iOS to increase accessibility.

- **AI-based Fraud Detection**: Implementing machine learning algorithms to detect unusual activity and prevent fraud.

- **Advanced Reporting Tools**: Adding more detailed financial reports and budgeting tools for users.

11. Conclusion

The online banking system in Java provides users with a secure, reliable, and user-friendly platform to manage their bank accounts. By utilizing Java's robust framework, combined with modern security practices and database management, this system offers a complete solution for digital banking. Future enhancements could make it even more versatile and user-centric, catering to a broader audience.

12. References

- Java documentation:https://docs.oracle.com/javase/

- MySQL documentation: https://dev.mysql.com/doc/

- Spring Framework:https://spring.io/projects/spring-boot