Project Documentation

Project title : Online Banking System

Submitted By : Mokshada Gupta
Kriti Rastogi
Aakriti Mishra
Jansi Verma

Contents

Introduction

- The **Banking System Project** is designed to simplify banking operations, enabling secure account management, transactions, and reporting.
- Traditional banking systems are often inefficient, insecure, and lack user-friendly interfaces.
- The project aims to create a robust platform to streamline banking processes for customers and administrators.

Objectives

- Enable secure and efficient account and transaction management.
- Provide role-based access control for customers and administrators.
- Generate detailed transaction and account reports.

System Design

Architectural Overview

- The system employs a 3-tier architecture, ensuring modularity, scalability, and maintainability. The **Presentation Layer** serves as the front-end interface, allowing users to interact with the system through web or desktop applications. It handles input collection and output display. The **Service Layer** contains the core business logic, acting as a mediator between the presentation and data layers. It processes user requests, enforces business rules, and orchestrates operations. Finally, the **Data Layer** manages database interactions, including CRUD operations and secure data storage. This separation of concerns allows independent development, testing, and scaling of each layer for better performance and flexibility.

UML Diagrams

- Visualizes user interactions with the system, highlighting actions like account creation and transactions.
- Represents the database schema, defining entities, attributes, and relationships.
- Shows system structure, detailing classes(User, Account) with attributes and methods, and their relationships.

Implementation Details

Technologies Used:

- Java
- Spring Boot
- MySqL
- Junit
- Mokito

- **Features:**
- **User Registration**: Create user accounts with validation.
- **Account Management**: Add, update, and delete accounts.
- **Transactions**: Supports deposits, withdrawals, and transfers.
- **Reporting**: Generates transaction and balance reports.

Testing

Test Case ID	Module	Input	Expected Output	Status
TC001	Login Module	Valid credentials	Login successful	Passed
TC002	Transactions	Deposit $500	Balance updated	Passed

- **Tools Used :**
- JUnit for unit testing.
- Apache JMeter for performance testing.
- Mockito for test of DAO classes

User Manual

- **Install Java and MySQL**:

  Ensure Java (JDK 8 or later) and MySQL are installed on your system. Configure the JAVA_HOME environment variable for Java and set up a MySQL root user with a secure password.

- **Clone the Repository** :

  Use Git to clone the project repository from the version control system (e.g., GitHub) to your local machine.

- **Configure the Database :**

  Import the provided SQL script into MySQL to create the database and tables. Update the database credentials in the application's application.properties file .

- **Deploy the Application:**

  Use Apache Tomcat to deploy the WAR file generated during project build, then access the application through the specified URL.

Usage

- **Login**: Enter your credentials (username and password) on the login screen to access the system. The system verifies your details and provides role-based access to features (e.g., customer or admin).
- **Account Management:** Navigate to the "Accounts" section to create, update, or delete bank accounts. View account details and manage linked customer information securely.
- **Transactions**: Access the "Transactions" section to perform deposits, withdrawals, or fund transfers. Enter the required details, and the system updates account balances in real time while maintaining a transaction log.

Conclusion

- The Banking System project successfully delivers a robust, secure, and user-friendly solution to streamline banking operations. By implementing essential features such as user registration, account management, transactions, and reporting, the system ensures efficiency and accuracy for both customers and administrators. The project’s use of a 3-tier architecture enhances scalability and maintainability, making it suitable for future enhancements.
- Extensive testing, including unit and integration testing, ensures reliability, while security measures like encryption and role-based access control safeguard user data.