# Banking Application with Spring Boot - Documentation

## Table of Contents

1. [Introduction](#introduction)
2. [Technologies Used](#technologies-used)
3. [Features](#features)
   * [User Management](#user-management)
   * [Account Management](#account-management)
   * [Transactions](#transactions)
   * [Notifications](#notifications)
4. [Architecture](#architecture)
5. [Database Schema](#database-schema)
6. [Endpoints](#endpoints)
7. [Security](#security)
8. [Getting Started](#getting-started)
9. [Additional Features](#additional-features)

## Introduction

This documentation provides a comprehensive guide for a Banking Application built using Spring Boot. The application covers major functionalities like user management, account management, transactions, and notifications.

## Technologies Used

* **Spring Boot**
* **Spring Data JPA**
* **Spring Security**
* **Hibernate**
* **MySQL/PostgreSQL**
* **Thymeleaf (for front-end)**
* **RESTful APIs**
* **Maven/Gradle**

## Features

### 1. User Management

#### a. User Registration (Account Creation)

Users can create a new account by providing necessary details such as CustomerID, Account Number, First Name, Last Name, Phone Number, Email Address, and password. The UserAccounts table will be created in the database with the following schema:

* **CustomerID**: Alphanumeric (0-9, A-Z) with a fixed length of 9 characters, ensuring uniqueness and allowing multiple account numbers.
* **Account Number**: Numeric only, with a length of 14 characters.
* **Password**: Combination of a-z, A-Z, 0-7, and symbols.
* **First Name, Last Name, Phone Number, Email Address**: Will have default values.

#### b. User Login

Users can log in using either:

* Customer ID and password + Email OTP
* Email ID and password + Registered Phone OTP

#### c. Dashboard

After login, users will be directed to a dashboard displaying:

* Available Balance
* Options to update personal details
* Transaction history
* Account settings where users can:
  + Link other accounts (e.g., Business Account, Loan Account)
  + Remove other accounts
  + Delete Customer ID (requires OTP verification for deletion)

#### d. User Delete (Account Deletion)

Users can delete their accounts permanently from the application after OTP verification.

### 2. Account Management

#### a. View Account

Users can view their account details such as account number, account type, balance, and other relevant information.

#### b. Account Deletion

Users can delete an account. This requires proper authorization and may include security checks.

#### c. Account Creation

Users can create multiple accounts of different types (savings, checking, etc.).

### 3. Transactions

#### a. Create Transaction

Users can create transactions such as deposits, withdrawals, and payments.

#### b. View Transaction History

Users can view their transaction history with details such as date, amount, and transaction type.

#### c. Transfer Funds

Users can transfer funds between their own accounts or to other users' accounts within the application.

### 4. Notifications

#### a. SMS Notifications

Users receive SMS notifications for critical activities such as account creation, transactions, and account deletion.

#### b. Email Notifications

Users receive email notifications for activities such as registration, transactions, and profile updates.

## Architecture

The application follows a multi-tier architecture:

1. **Presentation Layer**: Handles the user interface (UI) using Thymeleaf.
2. **Business Layer**: Contains the business logic implemented using Spring services.
3. **Persistence Layer**: Manages database interactions using Spring Data JPA and Hibernate.
4. **Database Layer**: Stores user and transaction data in a relational database (MySQL/PostgreSQL).

## Database Schema

### Tables

* **UserAccounts**
  + customer\_id (PRIMARY KEY)
  + account\_number
  + password
  + first\_name
  + last\_name
  + phone\_number
  + email\_address
* **Accounts**
  + id (PRIMARY KEY)
  + user\_id (FOREIGN KEY)
  + account\_number
  + account\_type
  + balance
* **Transactions**
  + id (PRIMARY KEY)
  + account\_id (FOREIGN KEY)
  + transaction\_date
  + transaction\_type
  + amount
* **Notifications**
  + id (PRIMARY KEY)
  + user\_id (FOREIGN KEY)
  + type (SMS/Email)
  + message
  + timestamp

## Endpoints

### User Management

* POST /register: Register a new user.
* POST /login: User login.
* GET /profile: View user profile.
* PUT /profile: Update user profile.
* DELETE /profile: Delete user account.

### Account Management

* GET /accounts: View all user accounts.
* POST /accounts: Create a new account.
* DELETE /accounts/{id}: Delete an account.

### Transactions

* POST /transactions: Create a new transaction.
* GET /transactions: View transaction history.
* POST /transactions/transfer: Transfer funds.

### Notifications

* GET /notifications: View all notifications.
* POST /notifications: Create a notification.

## Security

* **Authentication**: Implemented using Spring Security with JWT tokens.
* **Authorization**: Role-based access control (RBAC) to secure endpoints.
* **Encryption**: Passwords are stored encrypted using BCrypt.

## Getting Started

### Prerequisites

* Java 11+
* Maven/Gradle
* MySQL/PostgreSQL