**Problem Statement: Banking System using Spring Data JPA**

**📌 Scenario:**

A bank wants to develop a **secure and efficient banking system** that allows customers to manage their accounts. The system should support **account creation, balance inquiry, deposits, withdrawals, and money transfers**. The bank also requires **transaction history tracking** and efficient data management using **Spring Data JPA** with **Hibernate and MySQL**.

**🏗️ Requirements:**

**1️⃣ Functional Requirements:**

✅ **Account Management:**

* Customers should be able to **create an account**.
* Customers should be able to **view account details**.

✅ **Transactions:**

* Customers should be able to **deposit money** into their account.
* Customers should be able to **withdraw money** (with balance validation).
* Customers should be able to **transfer money** between accounts.
* Customers should be able to **view transaction history**.

✅ **Admin Operations:**

* Admin should be able to **view all accounts**.
* Admin should be able to **view all transactions**.

**2️⃣ Non-Functional Requirements:**

✅ **Use Spring Boot & Spring Data JPA** for database operations.  
✅ **Use MySQL** as the relational database.  
✅ Implement **transactions** using @Transactional to ensure atomicity.  
✅ Handle **exceptions gracefully** (e.g., insufficient funds, account not found).  
✅ Implement **RESTful APIs** for seamless integration with UI or third-party apps.  
✅ Use **pagination and sorting** for fetching transaction history.  
✅ Implement **security** using Spring Security and JWT authentication.

**🏦 Database Schema (MySQL)**

**1️⃣ bank\_accounts Table (Stores customer account details)**

sql

CopyEdit

CREATE TABLE bank\_accounts (

account\_id BIGINT AUTO\_INCREMENT PRIMARY KEY,

account\_holder VARCHAR(255) NOT NULL,

balance DOUBLE NOT NULL

);

**2️⃣ transactions Table (Stores transaction history)**

sql

CopyEdit

CREATE TABLE transactions (

transaction\_id BIGINT AUTO\_INCREMENT PRIMARY KEY,

account\_id BIGINT,

transaction\_type ENUM('DEPOSIT', 'WITHDRAW', 'TRANSFER'),

amount DOUBLE NOT NULL,

transaction\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (account\_id) REFERENCES bank\_accounts(account\_id) ON DELETE CASCADE

);

**📌 Expected APIs**

| **HTTP Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| **POST** | /accounts | Create a new bank account |
| **GET** | /accounts/{id} | Get account details by ID |
| **GET** | /accounts | Get all bank accounts |
| **PUT** | /accounts/deposit/{id} | Deposit money into an account |
| **PUT** | /accounts/withdraw/{id} | Withdraw money from an account |
| **PUT** | /accounts/transfer | Transfer money between accounts |
| **GET** | /transactions/{id} | Get transaction history by account ID |
| **GET** | /transactions | Get all transactions (admin) |

**📌 Key Challenges to Solve**

✅ **Data Consistency:** Ensure transaction integrity using **@Transactional**.  
✅ **Concurrency Handling:** Prevent race conditions during balance updates.  
✅ **Performance Optimization:** Use **pagination** for large transaction histories.  
✅ **Security:** Implement **JWT authentication** and **role-based access control (RBAC)**.