**Problem Statement: Banking System using Spring JDBC Template**

**📌 Scenario:**

A bank wants to develop a **simple banking system** that allows customers to create accounts, deposit and withdraw money, check their balance, and transfer funds between accounts. The system should be built using **Spring JDBC Template** for database operations.

**🏗️ Requirements:**

**1️⃣ Functional Requirements:**

✅ Customers should be able to **create** a bank account.  
✅ Customers should be able to **view account details** (account ID, name, balance).  
✅ Customers should be able to **deposit money** into their account.  
✅ Customers should be able to **withdraw money**, ensuring sufficient balance.  
✅ Customers should be able to **transfer money** between two accounts.  
✅ Customers should be able to **view all accounts**.

**2️⃣ Non-Functional Requirements:**

✅ Use **Spring Boot & Spring JDBC Template** for database operations.  
✅ Use **MySQL** as the relational database.  
✅ Implement **transaction management** for deposit, withdrawal, and transfers.  
✅ Handle **exceptions gracefully** (e.g., insufficient funds, invalid accounts).  
✅ Use **REST API** for client interaction.  
✅ Ensure the system is **scalable and maintainable**.

**🏦 Database Schema (schema.sql)**

sql

CopyEdit

CREATE TABLE bank\_accounts (

account\_id BIGINT AUTO\_INCREMENT PRIMARY KEY,

account\_holder VARCHAR(255) NOT NULL,

balance DOUBLE NOT NULL

);

**📌 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 |

**📌 Key Challenges to Solve**

✅ **Data Consistency:** Ensure balance updates are atomic using **transactions**.  
✅ **Concurrency Handling:** Prevent race conditions when multiple users update the same account.  
✅ **Exception Handling:** Handle errors such as **insufficient funds, invalid account ID, and DB failures**.  
✅ **Security:** Prevent SQL injection and unauthorized access.