Project Design Overview: Transaction App

Project: Transaction App

This app implements a **JWT-based Login** with **Role-Based Authorization**. The system allows users to either **post transactions** or **view transaction history** based on their roles (Admin or Client).

Versions:

- **Spring Boot**: 2.5.4
- **Java**: 11
- **MySQL**: 8

Short Design Summary

1. **User Authentication with JWT**:
   When a user successfully logs in with their credentials (username and password), a **JWT token** is generated and returned to the client. This token is used for authentication and authorization in subsequent requests.

2. **Role-Based Access Control**:
   The system supports two roles:
   - **Client**:
     - Can **post transactions** to the system.
   - **Admin**:
     - Can **view transaction history** with filters (date and status).
     - Can **initiate the refund process**.
     -

Project Architecture

1. **Client Layer**:
   - **Tools**: POSTMAN

2. **API Layer (Controller Package)**:
    - **Responsibilities**:
        - Receives incoming HTTP requests.
        - Maps the request body (like JSON) to Java objects.
        - Extracts request parameters (e.g., JWT token from Authorization header).
        - Returns responses to the client (e.g., CommonResponse with status, message, and data).

3. **Service Layer (Business Logic)**:
    - **Responsibilities**:
        - Contains business logic and processes.
        - Validates incoming requests (DTOs) and interacts with the data layer (e.g., saving transactions, validating user roles).

4. **Data Access Layer (Repository)**:
    - **Responsibilities**:
        - Handles CRUD operations for entities (e.g., AppUser, TransactionDetails).
    - **Implementation**:
        - Uses **Spring Data JPA** interfaces (e.g., JpaRepository) to interact with the database.
5. **Security Layer**:
    - **Responsibilities**:
        - Manages authentication and authorization.
        - Verifies JWT tokens and roles.
        - Ensures that only authorized users (Admin or Client) can access their permitted resources.
        - Handles secure data access and authorization for API requests.

Flow Overview

1. **Frontend Interaction (Postman, React, Angular)**:
    - The frontend sends a request to a Controller endpoint (e.g., POST /txn-app/sign-in) with user credentials

2. **Controller**:
    - Receives and processes incoming requests.
    - Validates the request and maps data to Java objects.
    - Calls the appropriate service to handle the business logic (e.g., user login, posting transactions).

3. **Service Layer**:
    - Contains the core business logic, such as validating the user, posting transactions, filtering transaction data, etc.
    - Communicates with the **Repository** layer to fetch or save data in the database.

4. **Repository (Data Layer)**:
    - Interacts with the database through **Spring Data JPA** to persist and retrieve entities like AppUser and Transaction.

5. **Security Layer**:
    - Ensures that every API request has a valid JWT token.
    - Checks user roles (Admin or Client) and ensures the user has the necessary permissions.
    - If the user's JWT is valid and their role allows access, the request is processed; otherwise, the request is denied

6. **Response Handling**:
    - After the request is processed, the **Controller** returns a response to the frontend, typically in a **standard response structure** (e.g., CommonResponse), including status, message, and data.

7. **Logging & Exception Handling**:
    - Throughout the system, **logging** is used for tracking events, errors, and important actions (like login attempts), and **Exception handling**