Assignment: RESTful API Development with Java and MySQL

Problem Statement

In this assignment, you will develop a set of RESTful APIs capable of performing CRUD (Create, Read, Update, Delete) operations using HTTP requests. You will set up a MySQL database with specific tables, insert sample data, and implement REST APIs to process orders for an e-commerce system.

Objectives

This assignment tests your ability to:

- **Design and Set Up Database Entities**: Create a relational database schema using MySQL.
- **Write SQL Scripts**: Write SQL scripts to create and populate tables.
- **Implement RESTful APIs in Java**: Develop APIs using the Java programming language.
- **Interact with a Database from Java**: Use JDBC or ORM frameworks to connect your Java application to the MySQL database.

- **Apply HTTP and REST Protocols**: Understand and utilize HTTP methods and RESTful principles in API design.

Prerequisites

Working knowledge of:

- **MySQL Database**: SQL data types, DDL (Data Definition Language), and SQL queries.
- **Java Programming Language**: Basic to intermediate proficiency.
- **HTTP and RESTful APIs**: Understanding of HTTP methods (GET, POST, PUT, DELETE) and REST principles.
- **Java Web Frameworks** (Recommended): Familiarity with frameworks like Spring Boot for easier API development.

Database Schema

Create the required tables in a MySQL database using the naming conventions outlined in this document. Once the tables are set up, populate them with the provided data.

Instructions for Database Creation

- **Define Data Types**: For each column, choose appropriate MySQL data types (e.g., `INT`, `VARCHAR`, `DATE`).
- **Set Constraints**:
  - Primary keys should be set to auto-increment where appropriate.
  - Define foreign key relationships with proper referential integrity.
- **Create Tables**: Write SQL DDL statements (`CREATE TABLE`) to create the tables with the specified columns and relationships.

Below are the tables with columns and data types.

1. Customer

Column	Data Type	Constraints
customer_id	INT	PRIMARY KEY, AUTO_INCREMENT
first_name	VARCHAR(50)	NOT NULL
last_name	VARCHAR(50)	NOT NULL

2. Contact_Mech

Column	Data Type	Constraints
contact_mech_id	INT	PRIMARY KEY, AUTO_INCREMENT

customer_id	INT	NOT NULL, FOREIGN KEY REFERENCES Customer(customer_id)
street_address	VARCHAR(100)	NOT NULL
city	VARCHAR(50)	NOT NULL
state	VARCHAR(50)	NOT NULL
postal_code	VARCHAR(20)	NOT NULL
phone_number	VARCHAR(20)	NULL
email	VARCHAR(100)	NULL

3. Product

Column	Data Type	Constraints
product_id	INT	PRIMARY KEY, AUTO_INCREMENT
product_name	VARCHAR(100)	NOT NULL
color	VARCHAR(30)	NULL
size	VARCHAR(10)	NULL

4. Order_Header

Column	Data Type	Constraints
order_id	INT	PRIMARY KEY, AUTO_INCREMENT
order_date	DATE	NOT NULL
customer_id	INT	NOT NULL, FOREIGN KEY REFERENCES Customer(customer_id)
shipping_contact_mech_id	INT	NOT NULL, FOREIGN KEY REFERENCES Contact_Mech(contact_mech_id)
billing_contact_mech_id	INT	NOT NULL, FOREIGN KEY REFERENCES Contact_Mech(contact_mech_id)

5. Order_Item

Column	Data Type	Constraints
order_item_seq_id	INT	PRIMARY KEY, AUTO_INCREMENT
order_id	INT	PRIMARY KEY, FOREIGN KEY REFERENCES Order_Header(order_id)
product_id	INT	NOT NULL, FOREIGN KEY REFERENCES Product(product_id)
quantity	INT	NOT NULL

status	VARCHAR(20)	NOT NULL

Data Insertion

Insert sample data into the `Customer`, `Contact_Mech`, and `Product` tables to test your application.

Customers

Create **two** customers with associated contact mechanisms.

- **Customer 1**:
  - `first_name`: John
  - `last_name`: Doe
- **Customer 2**:
  - `first_name`: Jane
  - `last_name`: Smith

Contact_Mech

Associate contact mechanisms with the customers using valid USA addresses.

- **For Customer 1 (John Doe)**:
  - **Contact Mechanism 1**:
    - `street_address`: **"1600 Amphitheatre Parkway"**
    - `city`: **"Mountain View"**

    - `state`: **"CA"**
    - `postal_code`: **"94043"**
    - `phone_number`: **"(650) 253-0000"**
    - `email`: **"john.doe@example.com"**
  - **Contact Mechanism 2**:
    - `street_address`: **"1 Infinite Loop"**
    - `city`: **"Cupertino"**
    - `state`: **"CA"**
    - `postal_code`: **"95014"**
    - `phone_number`: **"(408) 996-1010"**
    - `email`: **"john.doe@work.com"**
- **For Customer 2 (Jane Smith)**:
  - **Contact Mechanism**:
    - `street_address`: **"350 Fifth Avenue"**
    - `city`: **"New York"**
    - `state`: **"NY"**
    - `postal_code`: **"10118"**
    - `phone_number`: **"(212) 736-3100"**
    - `email`: **"jane.smith@example.com"**

Products

Insert **five** products into the `Product` table.

1. **Product 1**:
   - `product_name`: T-Shirt
   - `color`: Red
   - `size`: M
2. **Product 2**:
   - `product_name`: Jeans
   - `color`: Blue
   - `size`: 32
3. **Product 3**:
   - `product_name`: Sneakers
   - `color`: White
   - `size`: 9
4. **Product 4**:
   - `product_name`: Jacket
   - `color`: Black
   - `size`: L
5. **Product 5**:
   - `product_name`: Hat
   - `color`: Green
   - `size`: One Size

Instructions for Data Insertion

- Write SQL `INSERT` statements to add the data into the respective tables.
- Ensure that foreign key relationships are maintained (e.g., `customer_id` in `Contact_Mech` matches an existing customer).
- Use the provided USA addresses in the `Contact_Mech` table to ensure data consistency.

API Development Tasks

You are required to develop RESTful APIs to handle order processing based on the database you have created.

Implementation Guidelines

- **Use Java**: Develop your APIs using the Java programming language.
- **Frameworks**: You may use frameworks like Spring Boot to simplify development.
- **Data Format**: Use JSON for request and response bodies.
- **HTTP Methods**: Appropriately use HTTP methods (GET, POST, PUT, DELETE) for each operation.
- **Error Handling**:
  - Implement proper error handling and input validation.
  - Return appropriate HTTP status codes (e.g., 200 OK, 201 Created, 400 Bad Request, 404 Not Found).

API Requirements

Implement the following functionalities:

1. **Create an Order**:
   - **Endpoint**: `POST /orders`
   - **Description**: Create a new order for a customer.
   - **Request Body**: Include order details such as `order_date`, `customer_id`, `shipping_contact_mech_id`, `billing_contact_mech_id`, and a list of `order_items`.
   - **Order Items**: Each order item should include `product_id`, `quantity`, and `status`.
2. **Retrieve Order Details**:
   - **Endpoint**: `GET /orders/{order_id}`
   - **Description**: Retrieve details of a specific order, including customer information and order items.
3. **Update an Order**:
   - **Endpoint**: `PUT /orders/{order_id}`
   - **Description**: Update order information such as shipping or billing addresses.
4. **Delete an Order**:
   - **Endpoint**: `DELETE /orders/{order_id}`
   - **Description**: Delete an existing order from the system.
5. **Add an Order Item**:

- **Endpoint**: `POST /orders/{order_id}/items`
- **Description**: Add a new item to an existing order.

6. **Update an Order Item**:
   - **Endpoint**: `PUT /orders/{order_id}/items/{order_item_seq_id}`
   - **Description**: Update details of a specific order item, such as quantity or status.
7. **Delete an Order Item**:
   - **Endpoint**: `DELETE /orders/{order_id}/items/{order_item_seq_id}`
   - **Description**: Remove an item from an existing order.

Sample Scenarios

Implement the following scenarios to ensure your APIs meet the requirements:

1. **Scenario 1: Create an Order**
   - **Customer**: John Doe (`customer_id` of the first customer you inserted).
   - **Order Details**:
     - `order_date`: Current date.
     - `shipping_contact_mech_id`: One of John Doe's contact mechanisms.
     - `billing_contact_mech_id`: Another of John Doe's contact mechanisms.
     - **Order Items**:
     - 2 units of T-Shirt (`product_id` corresponding to T-Shirt).
     - 1 unit of Jeans (`product_id` corresponding to Jeans).
2. **Scenario 2: Retrieve Order Details**
   - Retrieve the order created in Scenario 1.

- Ensure that the response includes all relevant details.

3. **Scenario 3: Update an Order Item**

- Update the quantity of Jeans in the order to 2 units.

4. **Scenario 4: Add an Order Item**

- Add 1 unit of Sneakers to the existing order.

5. **Scenario 5: Delete an Order Item**

- Remove the T-Shirt item from the order.

6. **Scenario 6: Delete an Order**

- Delete the entire order created in Scenario 1.

Run APIs

Validate the functionality of the implemented REST endpoints by executing requests via Postman or a similar API testing tool.

Bonus Tasks

- Integrate JWT-based authentication to secure all API endpoints.
- Develop user interfaces that interact with the APIs to enhance user experience.

Assignment Submission

- Upload the complete project code to your GitHub repository.
- Include screenshots demonstrating the successful execution of each API endpoint in Postman.