Assignment 2 - RESTful E-Commerce

Building a Spring Boot Product API with Relational Model Classes

Objective Design, implement, and test a robust RESTful API using Spring Boot to serve as the backend for a product catalog in an e-commerce application. This assignment will solidify your understanding of:

- **CRUD Operations:** Create, retrieve, update, and delete product data in a structured, scalable manner.
- **Database Integration:** Ensure data persistence with a relational database using Spring Data JPA.
- **JSON and XML Support:** Enable seamless data interchange in both JSON and XML formats.
- **RESTful Error Handling:** Provide meaningful feedback on API operations through appropriate HTTP status codes.

Scenario Imagine you are part of a team developing a backend API for an e-commerce site's product catalog. Your API will manage products and their associated categories, requiring you to define two model classes: `Product` and `Category`. These classes should have a one-to-one relationship, representing each product's unique category for better data organization and retrieval. Your assignment involves designing an API that supports full CRUD operations for both products and categories, handles complex data relationships, and ensures data persistence in a relational database.

Core Functionality

- **Product Management:**
  - **POST:** Add new products, including details such as title, description, price, inventory quantity, image reference, and category.
  - **GET:** Retrieve individual products by ID or fetch lists with optional filtering by category or keywords. Calculate and display a discounted price (10% off the original price).
  - **PUT:** Update existing product details.
  - **DELETE:** Remove products.
- **Category Management:**
  - Implement CRUD operations for product categories, where each `Product` is assigned a `Category` via a one-to-one relationship.

Database Integration

- **Database Selection:** Use a relational database (e.g., MySQL, PostgreSQL) to maintain data integrity and structure.
- **Database Table Design:**

    - Create a `products` table with columns such as `product_id`, `title`, `description`, `price`, `inventory_quantity`, `image_url`, and `category_id`.
    - Create a `categories` table with columns like `category_id` and `category_name` to establish a one-to-one link with the `products` table.
- **Spring Data JPA:** Leverage Spring Data JPA for entity mapping, database interactions, and establishing the relationship between `Product` and `Category`.

Technical Requirements

1. **Spring Boot Framework:** Use Spring Boot as the foundational framework for your application.
2. **Database Configuration:** Set up and configure your database with Spring Data JPA for seamless integration.
3. **Data Models:**
    - Define `Product` and `Category` model classes with a one-to-one relationship.
4. **REST Controllers:** Develop controllers to expose RESTful endpoints for CRUD operations on products and categories.
5. **Content Negotiation:** Use Jackson and Spring's support to enable XML and JSON responses.
6. **Error Handling:** Implement clear error handling for operations to guide users on issues.

Testing, Documentation, and Presentation

- **Testing:** Develop test cases using Postman or curl for all endpoints.
- **Submission:** Package your project as a ZIP or Git repository, with complete code and documentation.
- **Presentation (Required):** Be prepared to demonstrate the application to the instructor in a live, synchronous meeting, explain your design choices, and discuss any challenges. No written report or recorded video is required for this assignment.

Getting Started

1. **Project Setup:** Start a new Spring Boot project using your IDE or the Spring Initializr (https://start.spring.io/).
2. **Database Configuration:** Set up your database and configure the connection in the application properties.
3. **Implement Core Functionality:** Focus on building product CRUD operations, then expand with category management and additional features.