Java Spring Boot Backend Coding Challenge: Product Management System

Scenario

You are tasked with developing a **Product Management System** where users can manage the inventory of products. You need to implement a RESTful API for **CRUD operations** on products, as well as searching and filtering functionalities.

Objective

Build a Spring Boot application with the following layers:

1. **Controller**
2. **Service**
3. **Repository**
4. **Model**

Use **PostgreSQL** as the database. Your application should allow the following operations:

- Create a new product.
- Retrieve all products.
- Retrieve a product by its ID.
- Update product details.
- Delete a product by its ID.
- Search for products by name or category (case-insensitive).
- Filter products by a price range.

Instructions

1. Create the Product Model

Define an entity class Product with the following fields:

- id (Long): Primary key, auto-generated.
- name (String): Name of the product (non-null).
- description (String): Short description of the product.
- price (Double): Price of the product (non-null).
- category (String): Category of the product (e.g., Electronics, Furniture, etc.).
- availableStock (Integer): Quantity available in stock.

Ensure this class is annotated with **@Entity** and contains proper JPA annotations for table mapping.

---

2. Implement the Repository Layer

Create a repository interface ProductRepository that extends **JpaRepository<Product, Long>**. Add the following custom query methods:

- List<Product> findByNameContainingIgnoreCase(String name);
- List<Product> findByCategoryIgnoreCase(String category);
- List<Product> findByPriceBetween(Double minPrice, Double maxPrice);

---

3. Implement the Service Layer

Create a service interface ProductService with the following methods:

- List<Product> getAllProducts();
- Product getProductById(Long id);
- Product createProduct(Product product);
- Product updateProduct(Long id, Product productDetails);
- void deleteProductById(Long id);
- List<Product> searchProductsByName(String name);
- List<Product> filterProductsByCategory(String category);
- List<Product> filterProductsByPriceRange(Double minPrice, Double maxPrice);

Create a class ProductServiceImpl that implements this interface. Use the ProductRepository for database operations.

---

4. Implement the Controller Layer

Create a controller class ProductController with the following endpoints:

- **GET /api/products**: Retrieve all products.
- **GET /api/products/{id}**: Retrieve a product by its ID.
- **POST /api/products**: Create a new product.
- **PUT /api/products/{id}**: Update a product's details.
- **DELETE /api/products/{id}**: Delete a product by its ID.
- **GET /api/products/search?name={name}**: Search for products by name (partial match).
- **GET /api/products/filter/category?category={category}**: Filter products by category.
- **GET /api/products/filter/price?minPrice={minPrice}&maxPrice={maxPrice}**: Filter products by price range.

Use @RestController and @RequestMapping("/api/products") to define the controller.