

## Exercise Lab8 ITCSIU22266

The image shows a screenshot of a web-based Product Management System. At the top, there is a header bar with a logo and the text "Product Management System". Below the header is a search bar with the placeholder "Search products." and a "Search" button. A table displays three products:

ID	Code	Name	Price	Quantity	Category	Actions
1	P001	Laptop Dell XPS 13	\$1299.99	10	Electronics	<span>Edit</span> <span>Delete</span>
2	P002	iPhone 15 Pro	\$999.99	25	Electronics	<span>Edit</span> <span>Delete</span>
3	P003	Office Chair	\$199.99	50	Furniture	<span>Edit</span> <span>Delete</span>

Below the table, there is a modal window titled "+ Add New Product" with the following fields:

- Product Code \***: Input field containing "Enter product code (e.g., P001)".
- Product Name \***: Input field containing "Enter product name".
- Price (\$)**: Input field containing "0.00".
- Quantity \***: Input field containing "0".
- Category \***: A dropdown menu labeled "Select category".
- Description**: A text area with the placeholder "Enter product description (optional)".
- Save Product** button.
- Cancel** button.

## Entity:

Maps Java class to database table.

```
public Product(String productCode, String name, BigDecimal price, Integer quantity, String category, String description) {
    this.productCode = productCode;
    this.name = name;
    this.price = price;
    this.quantity = quantity;
    this.category = category;
    this.description = description;
}
```

## Repository:

```
@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {

    // Spring Data JPA generates implementation automatically!

    // Custom query methods (derived from method names)
    List<Product> findByCategory(String category);

    List<Product> findByNameContaining(String keyword);

    List<Product> findByPriceBetween(BigDecimal minPrice, BigDecimal maxPrice);

    List<Product> findByCategoryOrderByPriceAsc(String category);

    boolean existsByProductCode(String productCode);

    // All basic CRUD methods inherited from JpaRepository:
    // - findAll()
    // - findById(Long id)
    // - save(Product product)
    // - deleteById(Long id)
    // - count()
    // - existsById(Long id)
}
```

Extends JpaRepository<Product, Long> -  
gets free CRUD methods:

- findAll(), findById(), save(), deleteById()

Custom queries derived from method names (Spring Data JPA magic):

- `findByCategory()` -> `SELECT * FROM products WHERE category = ?`
- `findByNameContaining()` -> `SELECT * FROM products WHERE name LIKE %?%`

Service:

```
public interface ProductService {  
  
    List<Product> getAllProducts();  
  
    Optional<Product> getProductById(Long id);  
  
    Product saveProduct(Product product);  
  
    void deleteProduct(Long id);  
  
    List<Product> searchProducts(String keyword);  
  
    List<Product> getProductsByCategory(String category);  
}
```

Business logic layer between controller and repository

DB connection:

```
# 1. Database Configuration (MySQL Changes)  
# Use the MySQL JDBC URL format  
spring.datasource.url=jdbc:mysql://localhost:3306/product_management;encrypt=true;trustServerCertificate=true  
# replace 'root' and 'your_password' with your MySQL credentials  
spring.datasource.username=root  
spring.datasource.password=sa  
# use the MySQL JDBC Driver  
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```