REPORT EXERICSE LAB 7

Student Name: Nguyễn Tiến Anh

Student ID: ITITDK22128

Course: Web Application Development

Lab 7: SPRING BOOT & JPA CRUD

Github: https://github.com/TienAnh0108/Lab7_Exercise.git

Ex 1:

ProductServiceImpl.java 1 | application.properties | ProductController.java 1

product-management > product-management > src > main > resources > application.properties

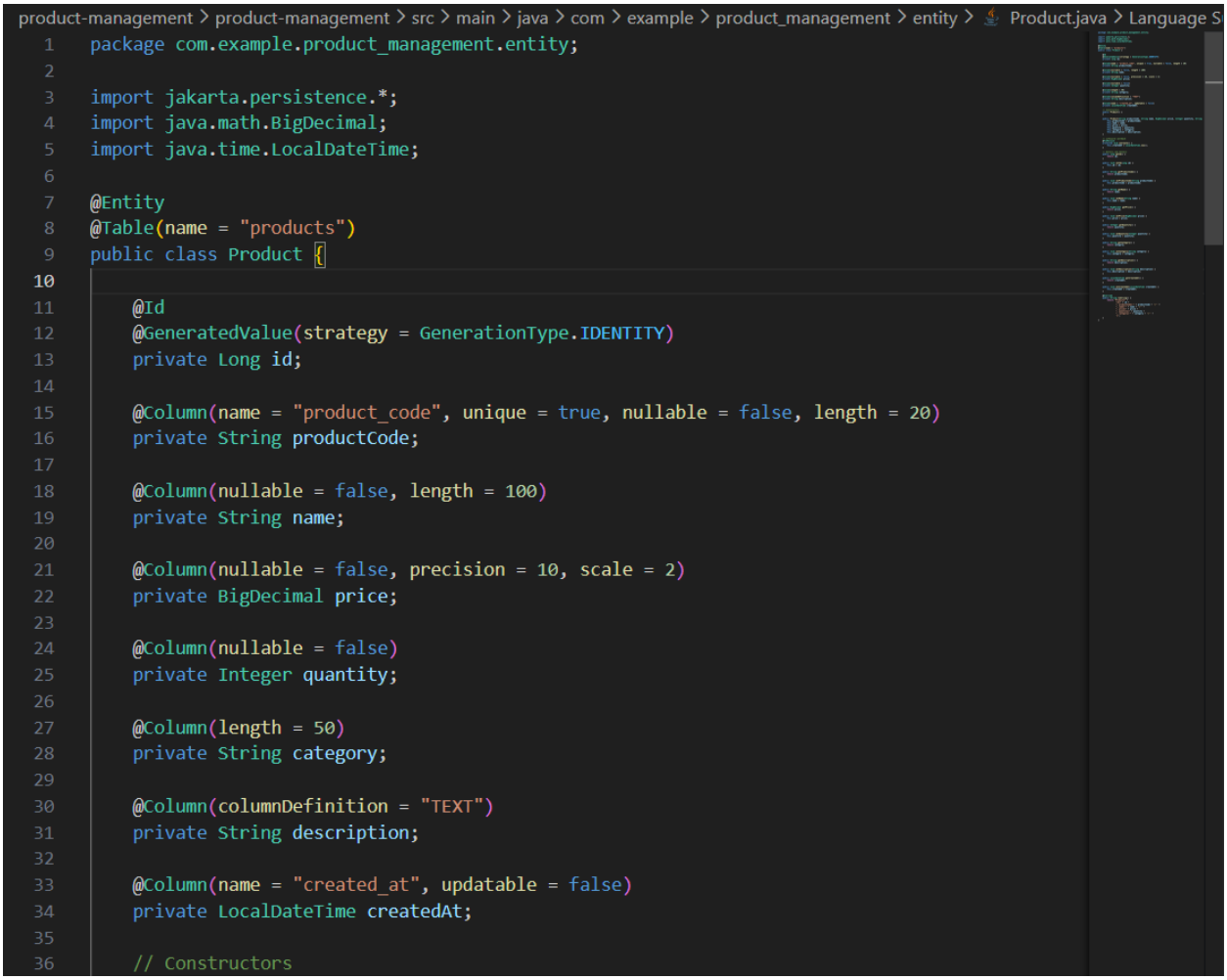
```
# Application Name
spring.application.name=product-management

# Server Port
server.port=8080

# Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/product_management?useSSL=false&serverTimezo
spring.datasource.username=root
spring.datasource.password=Tienanh0108!
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA/Hibernate Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

# Thymeleaf Configuration
spring.thymeleaf.cache=false
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html

# Logging
logging.level.org.springframework=INFO
logging.level.com.example.productmanagement=DEBUG

```

File: application.properties **Role:** Defines the runtime environment settings.

- **Database:** Configures the MySQL connection details (url, username, password) used by the Hikari connection pool.

- **JPA/Hibernate:** spring.jpa.hibernate.ddl-auto=update allows Hibernate to automatically modify the table structure based on entity changes, simplifying development.
- **Thymeleaf:** Disables caching (spring.thymeleaf.cache=false) for fast template reloading during development.

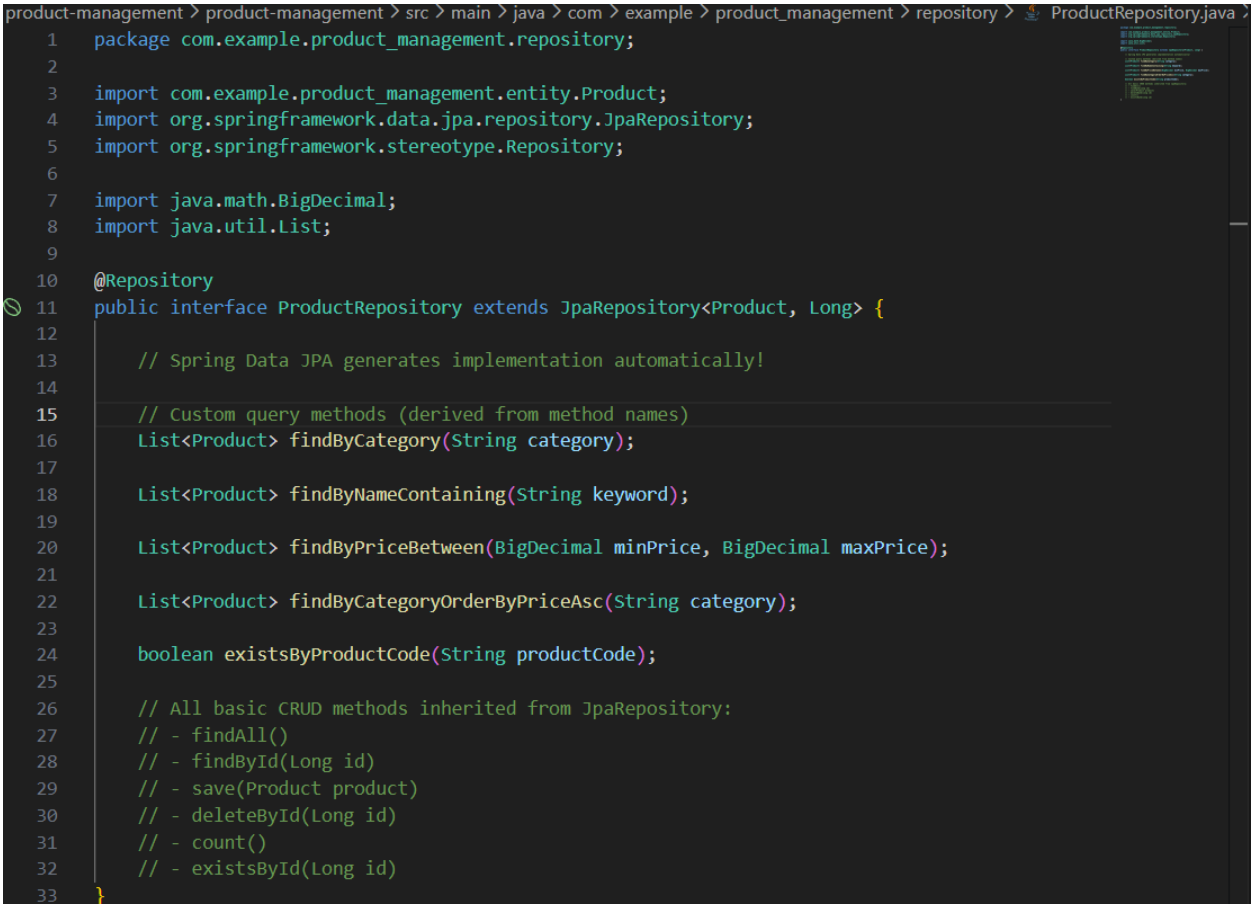
Ex 2:



File: Product.java **Role:** Represents the data structure and the corresponding database table.

- **Mapping:**
    - @Entity, @Table(name = "products"): Maps this class to the products table.
    - @Id, @GeneratedValue(strategy = GenerationType.IDENTITY): Defines the primary key with auto-increment generation.

- **Fields:** Contains essential product fields: productCode, name, price, quantity, category, and description.
- **Timestamp:** Uses @PrePersist and @Column(name = "created_at", updatable = false) to automatically record the creation time when the entity is saved for the first time.

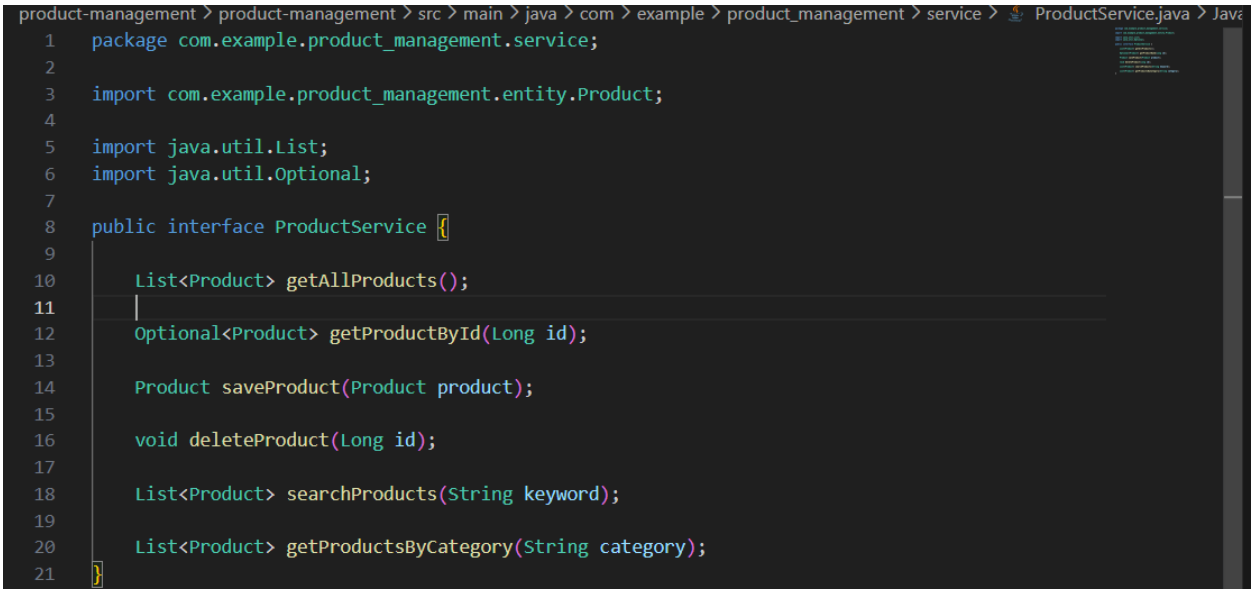


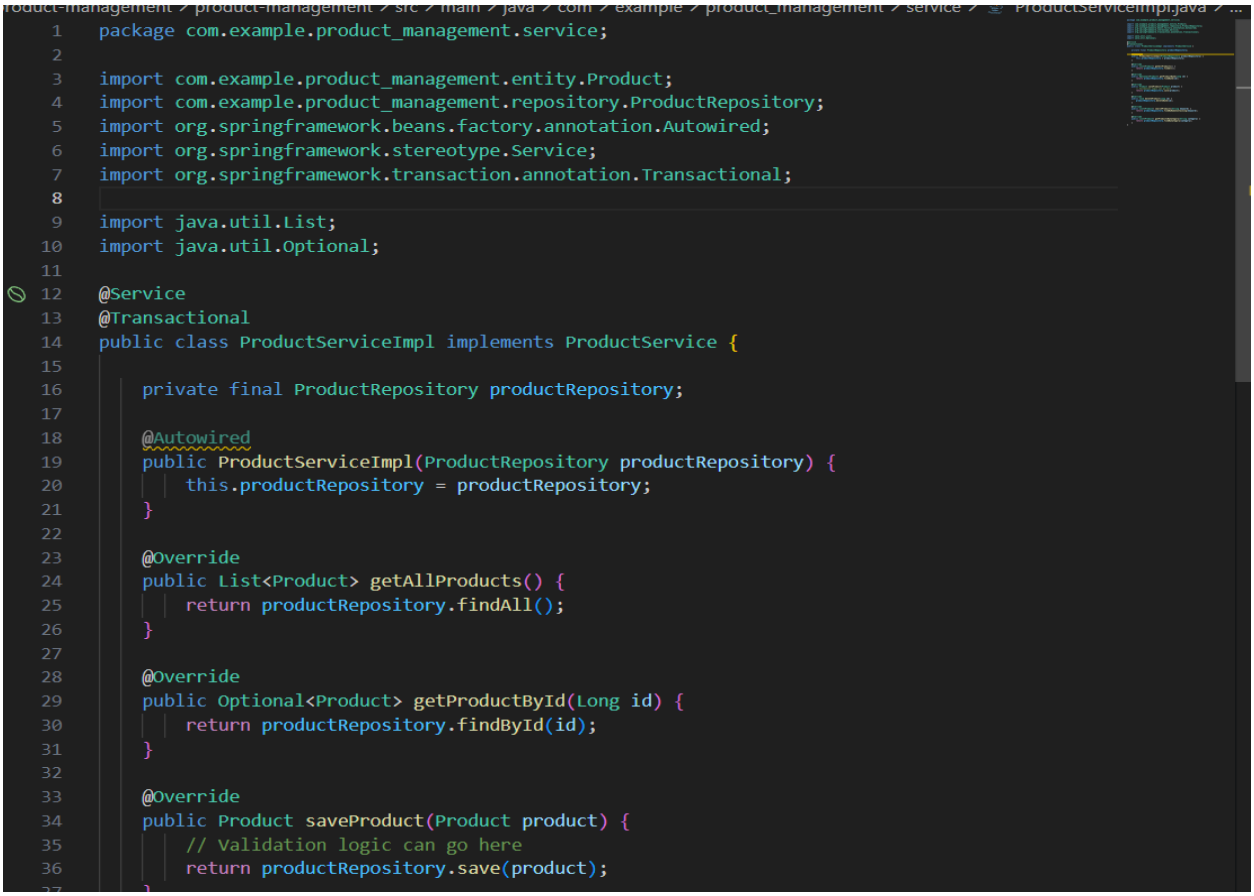
File: ProductRepository.java **Role:** Directly interfaces with the database.

- **Inheritance:** Extends **JpaRepository<Product, Long>**, which automatically provides basic CRUD methods (findAll, save, findById, deleteById) without requiring boilerplate code.
- **Custom Queries (Derived Queries):**
    - findByCategory(String category): Searches products based on their category.
    - findByNameContaining(String keyword): Searches by a substring within the product name (used by the Controller for searching).

- existsByProductCode(String productCode): Checks if a product code already exists

Ex 3:



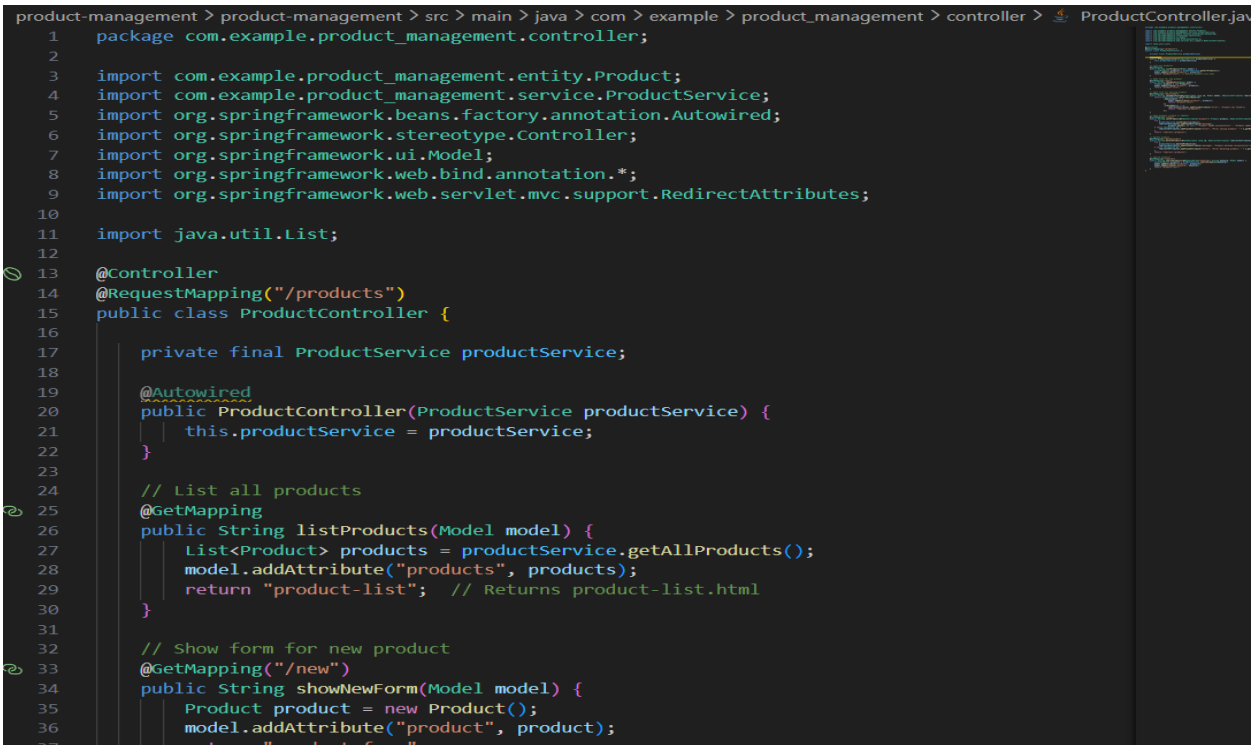


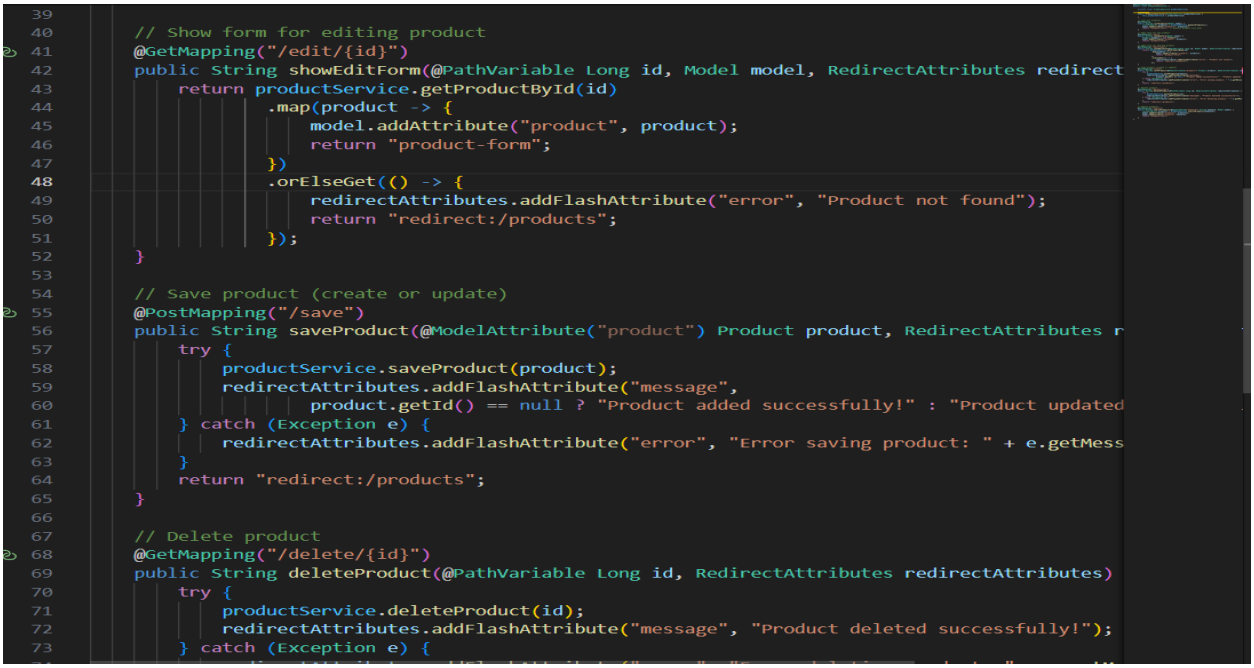
```
    @Override
    public void deleteProduct(Long id) {
        productRepository.deleteById(id);
    }

    @Override
    public List<Product> searchProducts(String keyword) {
        return productRepository.findByNameContaining(keyword);
    }

    @Override
    public List<Product> getProductsByCategory(String category) {
        return productRepository.findByCategory(category);
    }
}
```

Files: ProductService.java (Interface) & ProductServiceImpl.java (Implementation) **Role:** Holds the core business logic, acting as the intermediary between the Controller and the Repository.

- **Core Functionality:**
    - getAllProducts(): Retrieves all products.
    - getProductById(Long id): Retrieves a product by ID (returning an Optional).
    - saveProduct(Product product): Executes the insert or update operation.
    - deleteProduct(Long id): Executes the deletion operation.
    - searchProducts(String keyword): Searches products by name.
- **Data Integrity:** The ProductServiceImpl class uses **@Transactional** to ensure that database operations are executed safely and atomically (all or nothing)

Ex 4:


product-management > product-management > src > main > java > com > example > product_management > controller > ProductController.jav
1 package com.example.product_management.controller;
2
3 import com.example.product_management.entity.Product;
4 import com.example.product_management.service.ProductService;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.ui.Model;
8 import org.springframework.web.bind.annotation.*;
9 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
10
11 import java.util.List;
12
13 @Controller
14 @RequestMapping("/products")
15 public class ProductController {
16
17 private final ProductService productService;
18
19 @Autowired
20 public ProductController(ProductService productService) {
21 this.productService = productService;
22 }
23
24 // List all products
25 @GetMapping
26 public String listProducts(Model model) {
27 List<Product> products = productService.getAllProducts();
28 model.addAttribute("products", products);
29 return "product-list"; // Returns product-list.html
30 }
31
32 // Show form for new product
33 @GetMapping("/new")
34 public String showNewForm(Model model) {
35 Product product = new Product();
36 model.addAttribute("product", product);

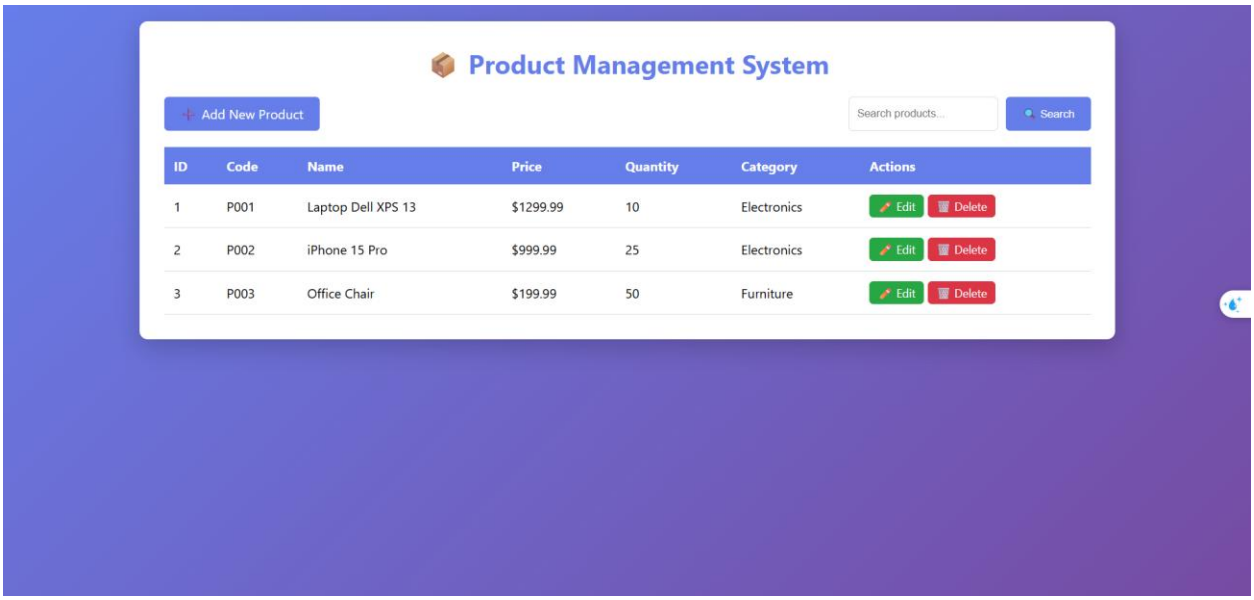

39
40 // Show form for editing product
41 @GetMapping("/edit/{id}")
42 public String showEditForm(@PathVariable Long id, Model model, RedirectAttributes redirect
43 return productService.getProductById(id)
44 .map(product -> {
45 model.addAttribute("product", product);
46 return "product-form";
47 })
48 .orElseGet(() -> {
49 redirectAttributes.addFlashAttribute("error", "Product not found");
50 return "redirect:/products";
51 });
52 }
53
54 // Save product (create or update)
55 @PostMapping("/save")
56 public String saveProduct(@ModelAttribute("product") Product product, RedirectAttributes r
57 try {
58 productService.saveProduct(product);
59 redirectAttributes.addFlashAttribute("message",
60 product.getId() == null ? "Product added successfully!" : "Product updated
61 } catch (Exception e) {
62 redirectAttributes.addFlashAttribute("error", "Error saving product: " + e.getMess
63 }
64 return "redirect:/products";
65 }
66
67 // Delete product
68 @GetMapping("/delete/{id}")
69 public String deleteProduct(@PathVariable Long id, RedirectAttributes redirectAttributes)
70 try {
71 productService.deleteProduct(id);
72 redirectAttributes.addFlashAttribute("message", "Product deleted successfully!");
73 } catch (Exception e) {

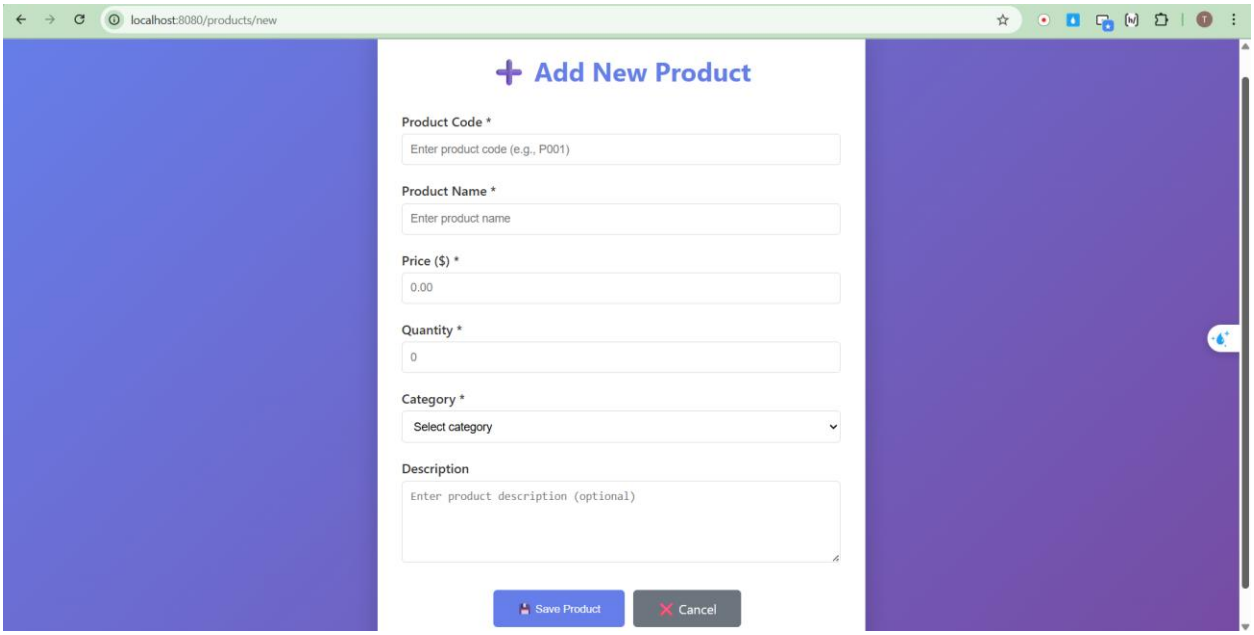
```java
    // Delete product
    @GetMapping("/delete/{id}")
    public String deleteProduct(@PathVariable Long id, RedirectAttributes redirectAttributes)
        try {
            productService.deleteProduct(id);
            redirectAttributes.addFlashAttribute("message", "Product deleted successfully!");
        } catch (Exception e) {
            redirectAttributes.addFlashAttribute("error", "Error deleting product: " + e.getMe
        }
        return "redirect:/products";
    }

    // Search products
    @GetMapping("/search")
    public String searchProducts(@RequestParam("keyword") String keyword, Model model) {
        List<Product> products = productService.searchProducts(keyword);
        model.addAttribute("products", products);
        model.addAttribute("keyword", keyword);
        return "product-list";
    }
}
```

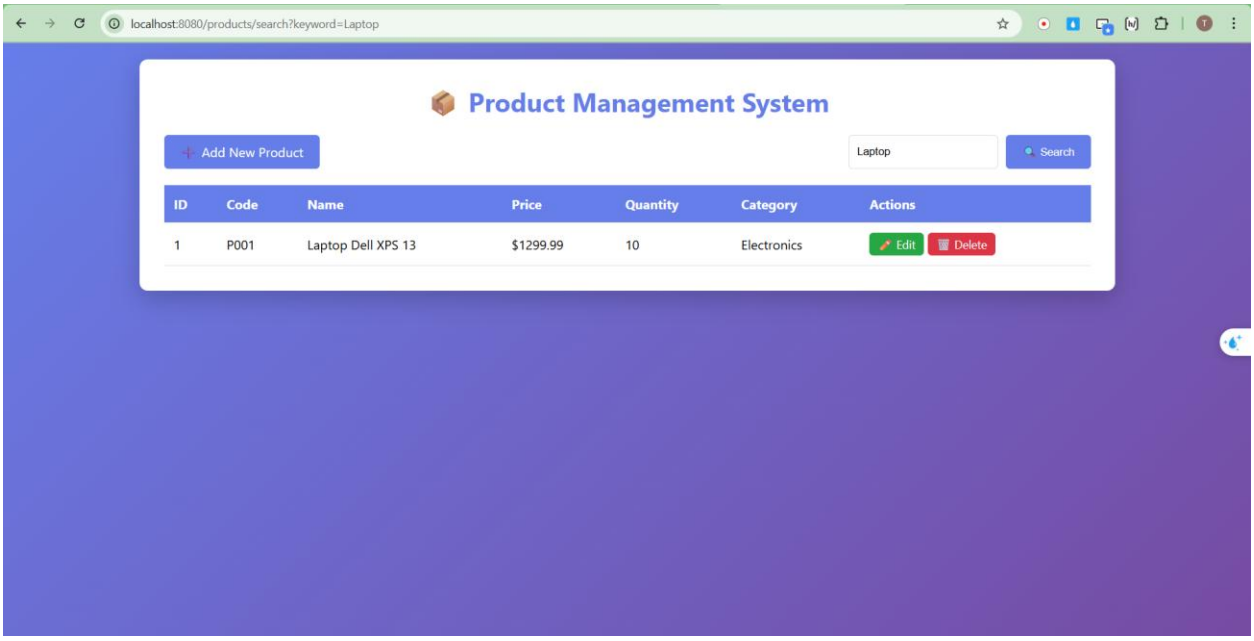
File: ProductController.java **Role:** Handles HTTP requests and manages the application flow.

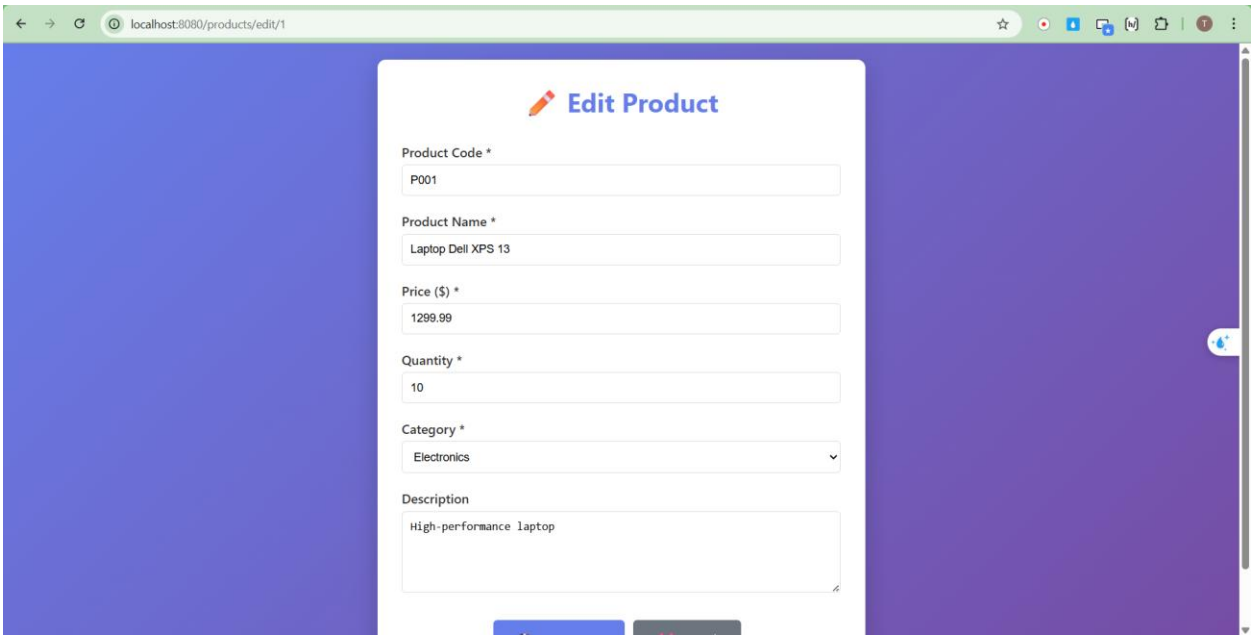
- **Class Mapping:** @RequestMapping("/products") – All URLs starting with /products are handled here.
- **List Products:** @GetMapping (/products) – Fetches all products, adds them to the Model, and returns the **product-list** view.
- **New Form:** @GetMapping("/new") – Prepares an empty Product object and returns the **product-form** view.
- **Edit Form:** @GetMapping("/edit/{id}") – Finds the product by ID; if found, it loads the product into the **product-form**; otherwise, it redirects with an error message.
- **Save (Create/Update):** @PostMapping("/save") – Receives the submitted Product object (@ModelAttribute), calls productService.saveProduct(), and redirects to /products with a success message.
- **Delete:** @GetMapping("/delete/{id}") – Calls productService.deleteProduct(), then redirects to /products with a success message.
- **Search:** @GetMapping("/search") – Receives a keyword (@RequestParam("keyword")), calls productService.searchProducts(), and returns the filtered results in the **product-list** view.

RESULT:


Product Management System
Add New Product
Search products...
Search
ID
Code
Name
Price
Quantity
Category
Actions
1
P001
Laptop Dell XPS 13
$1299.99
10
Electronics
Edit
Delete
2
P002
iPhone 15 Pro
$999.99
25
Electronics
Edit
Delete
3
P003
Office Chair
$199.99
50
Furniture
Edit
Delete


localhost:8080/products/new
Add New Product
Product Code *
Enter product code (e.g., P001)
Product Name *
Enter product name
Price ($) *
0.00
Quantity *
0
Category *
Select category
Description
Enter product description (optional)
Save Product
Cancel


localhost:8080/products/search?keyword=Laptop
Product Management System
Add New Product
Laptop
Search
ID
Code
Name
Price
Quantity
Category
Actions
1
P001
Laptop Dell XPS 13
$1299.99
10
Electronics
Edit
Delete


localhost:8080/products/edit/1
Edit Product
Product Code *
P001
Product Name *
Laptop Dell XPS 13
Price ($) *
1299.99
Quantity *
10
Category *
Electronics
Description
High-performance laptop