International University

School of Computer Science and Engineering

Web Application Development

Laboratory

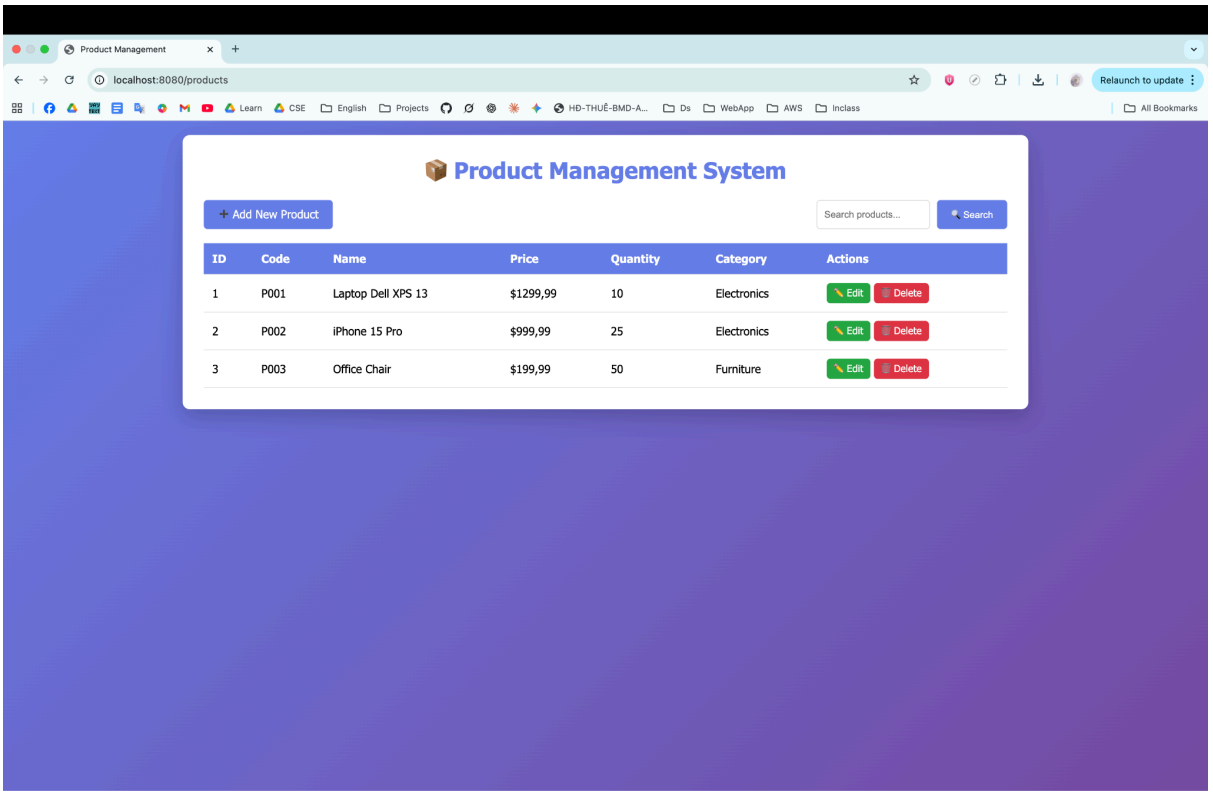
IT093IU

Lab #7 Practice

Submitted by

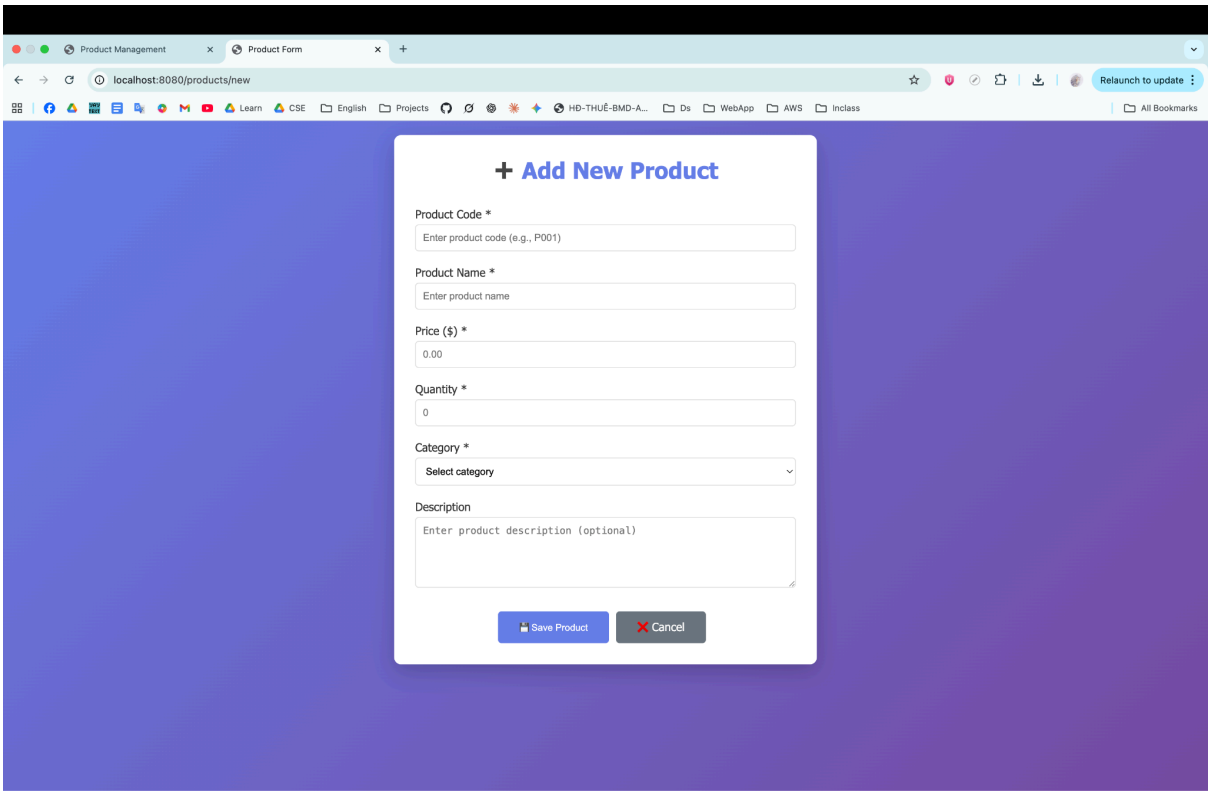
Nguyễn Hồng Ngọc Hân - ITCSIU22229

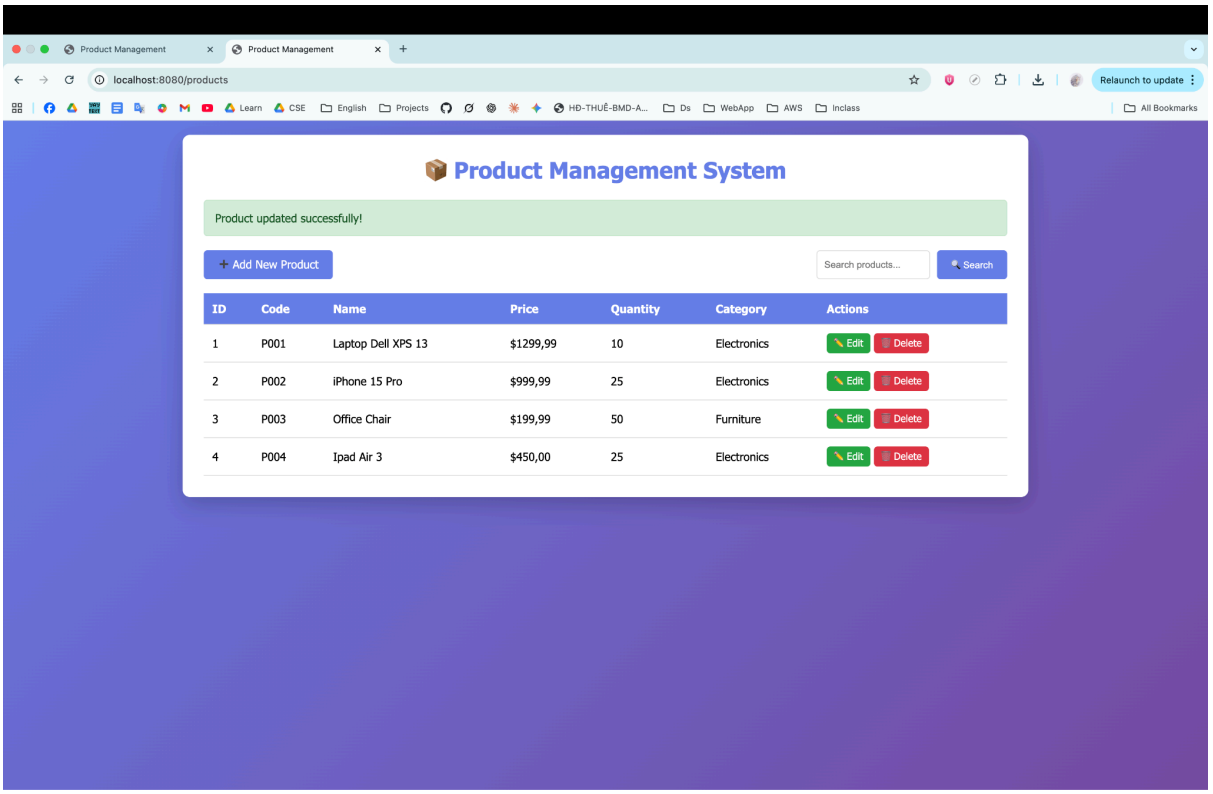
List all products:
http://localhost:8080/products



1. Browser → GET /products
2. ProductController receives request (@GetMapping)
3. Controller calls productService.getAllProducts()
4. Service calls productRepository.findAll()
5. JPA automatically queries the database.
6. Repository returns List<Product>
7. Controller adds data to Model
8. Returns view name "product-list"
9. Thymeleaf renders HTML
10. Response sent to browser

Add new product:
http://localhost:8080/products/new





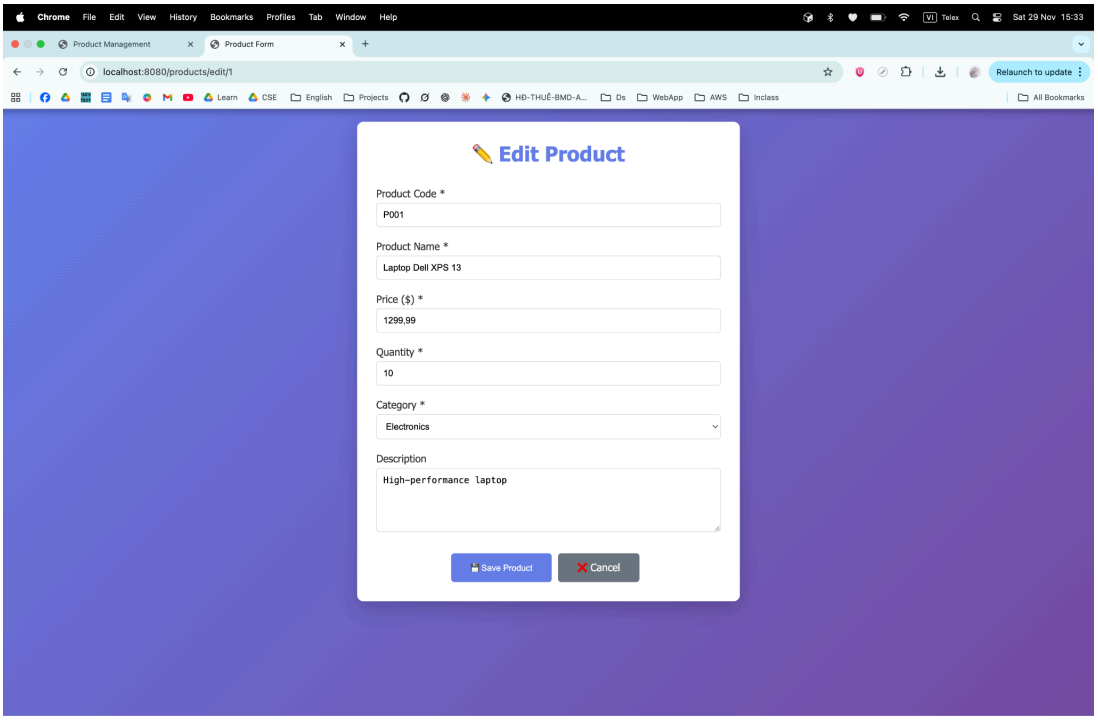
1. Browser → GET /products/new
2. ProductController receives request (@GetMapping("/new"))
3. Create new product (Product product = new Product())

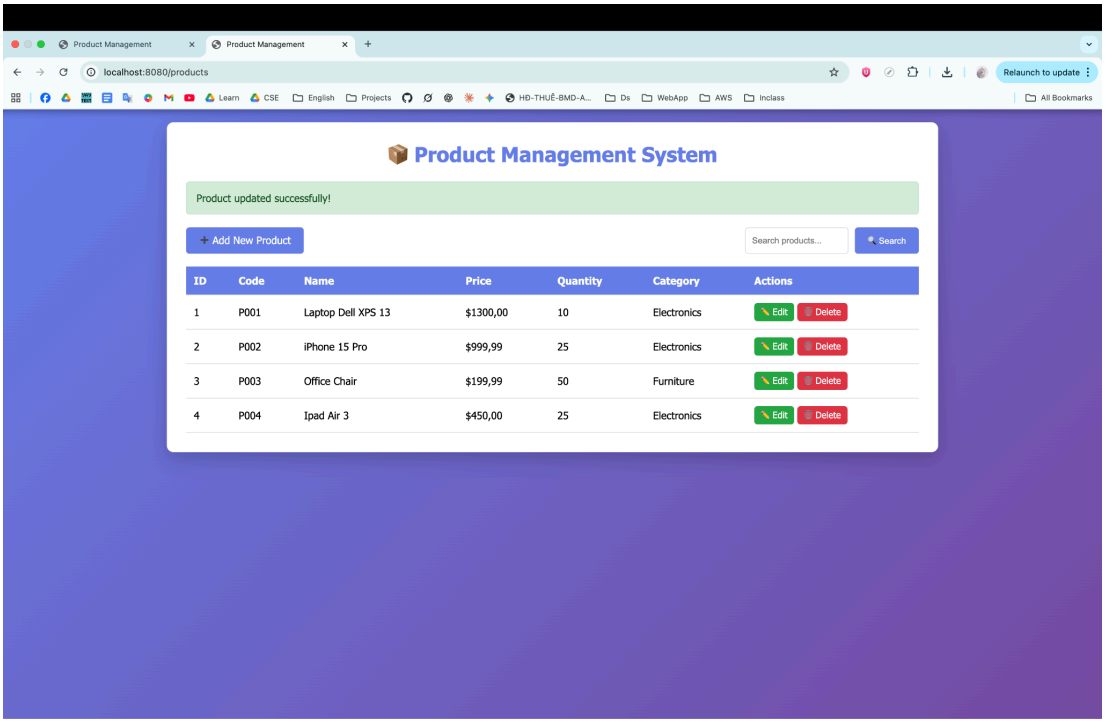
4. Repository returns Product
5. Controller adds data to Model (model.addAttribute("product", product))
6. Returns view name "product-form"
7. Thymeleaf renders HTML
8. Response sent to browser

Update or edit a product:
The edit logic operates as a retrieval-and-display cycle that precedes the actual saving process. When a user clicks the "Edit" button in the product list, the browser sends a GET request to a URL containing the specific product's ID (e.g., /products/edit/5). The ProductController captures this ID from the path and instructs the service layer to fetch the corresponding record from the database.
If the product is found, the controller injects this specific object into the model and renders the product-form.html template. Because the HTML form uses Thymeleaf binding (attributes like th:field="*{name}"), the page loads with all input boxes pre-filled with the existing database values rather than being empty. This allows the user to see the current data, make modifications, and submit the form, which carries the original ID in the hidden field to ensure the changes overwrite the correct row in the database.
1. Browser → Post /products/save
2. ProductController receives request (@PostMapping("/save"))
3. Controller calls productService.saveProduct(product)
4. Service calls productRepository.save(product)
5. JPA automatically save(Product product).
6. Repository returns "redirect:/products"
7. Controller adds data to
case 1: redirectAttributes.addFlashAttribute("message",
product.getId() == null ? "Product added successfully!" : "Product updated successfully!");
case 2: redirectAttributes.addFlashAttribute("error", "Error saving product: " + e.getMessage());
8. Returns view name "product-list"
9. Thymeleaf renders HTML
10. Response sent to browser

Edit product (ID=1):

http://localhost:8080/products/edit/1

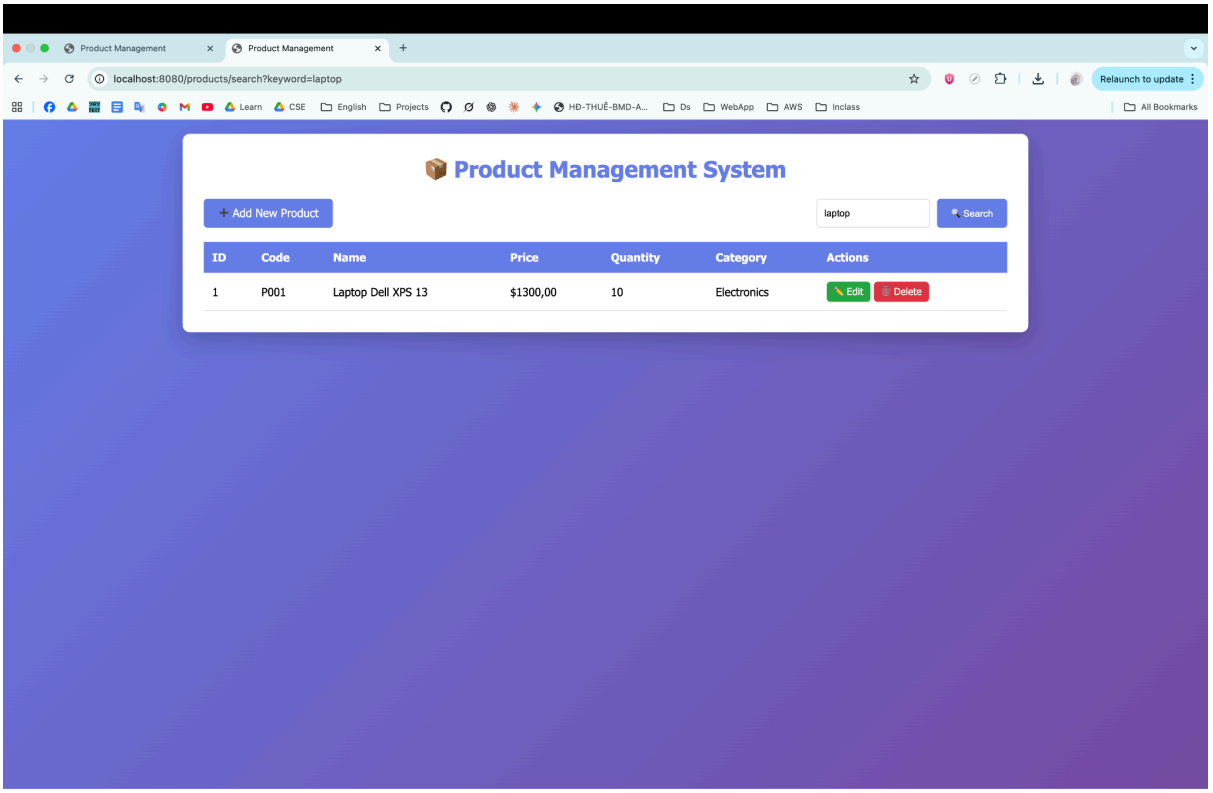




1. Browser → GET /products/edit/1
2. ProductController receives request (@GetMapping("/edit/{id}"))
3. Controller calls productService.getProductById(id)
4. Service calls productRepository.findById(id)
5. JPA automatically findById(Long id).
6. Repository returns Product
7. Controller adds data to Model

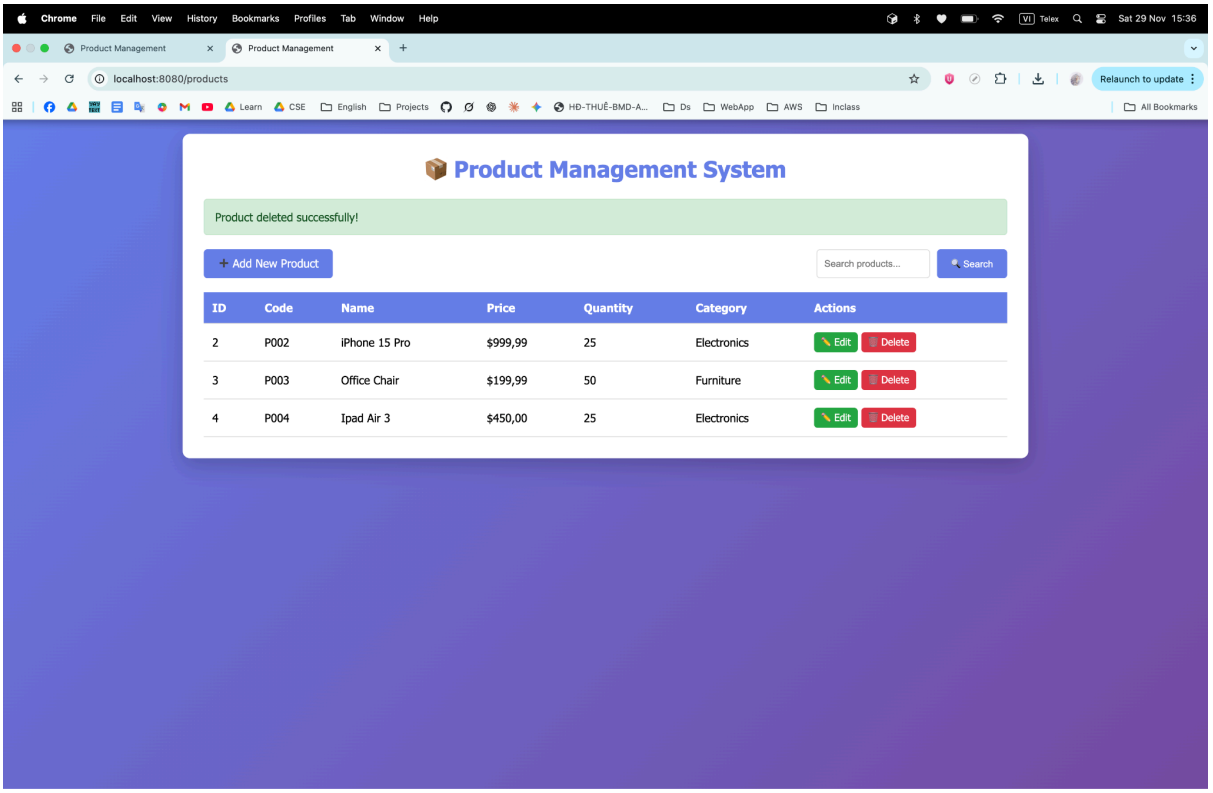
8. Returns view name "product-form"
9. Thymeleaf renders HTML
10. Response sent to browser

Search products:
http://localhost:8080/products/search?keyword=laptop



1. Browser → GET /products/search?keyword=laptop
2. ProductController receives request (@GetMapping("/search"))
3. Controller calls productService.searchProducts(keyword)
4. Service calls productRepository.findByNameContaining(keyword)
5. Custom query methods List<Product> findByNameContaining(String keyword)
6. Repository returns List<Product>
7. Controller adds data to Model
   model.addAttribute("products", products);
   model.addAttribute("keyword", keyword);
8. Returns view name "product-list"
9. Thymeleaf renders HTML
10. Response sent to browser

Delete product (ID=1):
http://localhost:8080/products/delete/1



1. Browser → GET /products/delete/1
2. ProductController receives request (@GetMapping("/delete/{id}"))
3. Controller calls productService.deleteProduct(id)
4. Service calls productRepository.deleteById(id);
5. JPA automatically deleteById(Long id)
6. Repository returns "redirect:/products"
7. Controller
   case 1: redirectAttributes.addFlashAttribute("message", "Product deleted successfully!");
  case 2: redirectAttributes.addFlashAttribute("error", "Error deleting product: " + e.getMessage())
8. Returns view name "product-list"
9. Thymeleaf renders HTML
10. Response sent to browser