**Lab 7**

**Name:** Trần Gia Khôi

**ID:** ITITIU21231

**Exercise 5: Advanced Search**

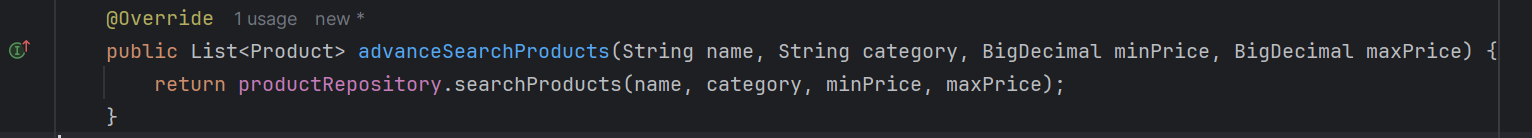
**Task 5.1: Multi-Criteria Search**

A screenshot of a computer

AI-generated content may be incorrect.

Add advanced search method to ProductService and ProductServiceImpl





Add advanced search in ProductController

A screen shot of a computer program

AI-generated content may be incorrect.

**Workflow:**

1. Browser → GET /products /advanced-search

2. ProductController receives request (@GetMapping)

3. Controller calls productService.advanceSearchProducts()

4. Service calls productRepository.searchProducts()

5. JPA automatically queries database

6. Repository returns List<Product>

7. Controller adds data to Model

8. Returns view name "product-list"

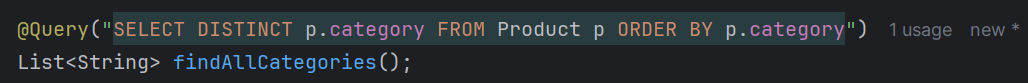
9. Thymeleaf renders HTML

10. Response sent to browser

**Task 5.2: Category Filter**

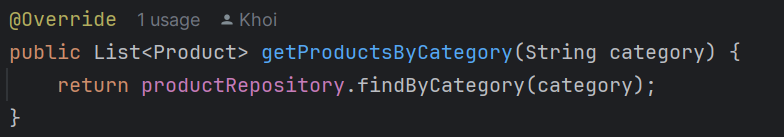
For filtering categories, we need to have two methods findByCategory() and findAllCategories() in ProductRepository





Define the getProductsByCategory() and categoryFilter() in ProductService and implement that method in ProductServiceImpl







A screenshot of a computer screen

AI-generated content may be incorrect.

Then called them in ProductController

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Workflow:**

**User selects “Electronics” from the dropdown**

1. Browser → GET /products /filter?category=Electronics

2. ProductController receives request (@GetMapping)

3. Controller calls productService.categoryFilter()

4. Controller evaluate the **if** condition

5. Controller calls productService.getProductsByCategory(“Electronics”)

6. Service calls productRepository.findByCategory(“Electronics”)

7. JPA automatically queries database

8. Repository returns List<Product>

9. Controller adds data to Model

10. Returns view name "product-list"

11. Thymeleaf renders HTML

12 . Response sent to browser

**Task 5.3: Search with Pagination**

Modify method in ProductRepository to use Pageable



Also modifying the Service



A screen shot of a computer

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

**Workflow:**

**User types “iPhone” in the search box and click “Search”, or click “Page 2” at the bottom of the search result**

1. Browser 🡪 GET /search?keyword=iPhone&page=1&size=10

2. ProductController receives request (searchProducts() method)

3. Controller creates Pageable object

4. Controller calls Service

5. Service calls Repository

6. JPA executes Database query

7. Repository returns Page<Product>

8. Controller extracts data for the View

9. Controller adds attributes to Model

10. Thymeleaf renders product-list.html

**Exercise 6: Validation**

**Task 6.1: Add Validation Annotations**

Validation annotations added

A computer screen shot of code

AI-generated content may be incorrect.

**Task 6.2: Add Validation in Controller**

Controller updated, which is saveProduct() method (POST request)

A screenshot of a computer program

AI-generated content may be incorrect.

**Task 6.3: Display Validation Errors**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 7: Sorting & Filtering**

**Task 7.1: Add Sorting**

Sorting name in ascending order

A screenshot of a computer

AI-generated content may be incorrect.

Sorting name in descending order

A screenshot of a computer

AI-generated content may be incorrect.

**Workflow:**

**User sorting products by name (ascending)**

1. Browser 🡪 GET /products?sortBy=name&sortDir=asc

2. ProductController receives request

3. Controller creates Sort object

4. Controller calls productService.getAllProducts(sort)

5. Service calls productRepository.findAll(sort)

6. JPA generates SQL

7. Database execute query

8. Repository returns List<Product> (Sorted) 🡪 Service 🡪 Controller

9. Controller adds data to Model

10. Thymeleaf renders product-list.htlm

**Task 7.2: Filter by Category**

Define a findByCategory() method that contain the Sort object as parameter:

****

Implement that logic in ProductService and ProductServiceImpl

****

**A screenshot of a computer

AI-generated content may be incorrect.**

Then call it in ProductController:

A screenshot of a computer program

AI-generated content may be incorrect.

By doing this, we make sure that sorting and filtering always doing together, not separating as before.

A screenshot of a computer

AI-generated content may be incorrect.

Filtering for “Electronics” and the result is as follows:

A screenshot of a computer

AI-generated content may be incorrect.

Then sorting for its name by ascending order

A screenshot of a computer

AI-generated content may be incorrect.

**Workflow:**

**User is viewing “Electronics” category and clicks the “Name” column header**

1. Browser 🡪 GET /products? sortBy=name&sortDir=asc &category=Electronics

2. ProductController receives request

3. Controller creates Sort object

4. Controller evaluates Filter condition

5. Controller calls Service

6. Service calls Repository

7. JPA executes Database query

8. Repository returns List<Product>

9. Controller adds data to Model

10. Thymeleaf renders product-list.html

**Exercise 8: Statistics Dashboard**

**Task 8.1: Add statistics methods**

A screenshot of a computer program

AI-generated content may be incorrect.

**Task 8.2: Create Dashboard Controller**

DashboardController created

A screen shot of a computer

AI-generated content may be incorrect.

**Task 8.3: Create Dashboard View**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Display:**

1. Total products count

For this feature, we can simply define a method in ProductService and implement this method in ProductServiceImpl



A screenshot of a computer

AI-generated content may be incorrect.

The count() method is a built-in method in JPA

Then call it in controller



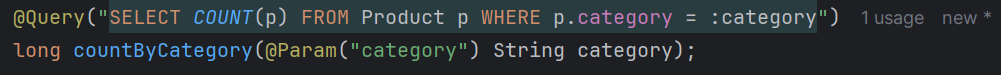
Result:

A blue rectangular object with a white border

AI-generated content may be incorrect.

2. Products by category

For this feature, we use the method countByCategory() in ProductRepository



ProductService and ProductServiceImpl



A computer screen with text

AI-generated content may be incorrect.

By using HashMap data structure, we can easily mapping the number of products for each category

DashboardController



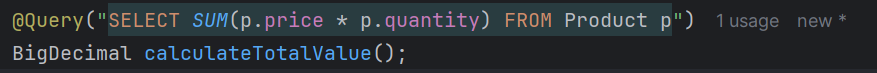
Result:

A screenshot of a phone

AI-generated content may be incorrect.

3. Total inventory value

To display inventory value, the custom method in ProductRepository is calculateTotalValue()



Method in ProductService and implementation in ProductServiceImpl



A screenshot of a computer

AI-generated content may be incorrect.

Call method in controller



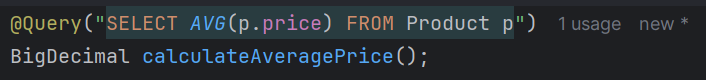
Result:

A green rectangular object with white text

AI-generated content may be incorrect.

4. Average product price

Custom method in ProductRepository



Method in ProductService and ProductServiceImpl



A screen shot of a computer

AI-generated content may be incorrect.

Call in controller



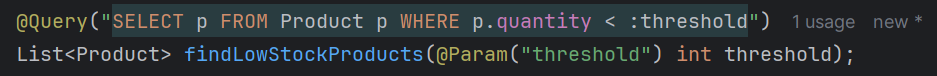
Result:

A blue rectangle with white text

AI-generated content may be incorrect.

5. Low stock alerts

Custom method in ProductRepository



Define method in ProductService and ProductServiceImpl



A screen shot of a computer

AI-generated content may be incorrect.

Call method in DashboardController



Result:

A screenshot of a computer

AI-generated content may be incorrect.

6. Recent products

Custom method in ProductRepository



Method defined in ProductService and implementation in ProductServiceImpl



A screenshot of a computer

AI-generated content may be incorrect.

Call method in controller



Result:

A screenshot of a computer

AI-generated content may be incorrect.

**Workflow:**

**User visit the dashboard to view “Product Dashboard”**

1. Browser 🡪 GET /dashboard

2. DashboardController receives request (@GetMapping)

3. Controller calls productService.getTotalCount()

4. Service calls productRepository.count()

5. Controller calls productService.getTotalValue()

6. Service calls productRepository.calculateTotalValue()

7. Controller calls productService.getAveragePrice()

8. Service calls productRepository.calculateAveragePrice()

9. Controller calls productService.getLowStockProducts()

10. Service calls productRepository.findLowStockProducts()

11. Controller calls productService.getRecentProducts()

12. Service calls productRepository.findTop5ByOrderByIdDesc()

13. Controller calls productService.getCategoryDistribution()

14. Service calls productRepository.findAllCategories()

15. Service calls productRepository.countByCategory()

16. JPA automatically queries database

17. Repository returns number of products to display “Total Product”

18. Repository returns total value in number to display “Total Inventory Value”

19. Repository returns average price in number to display “Average Price”

20. Repository returns List<Product> to display product that has low quantity

21. Repository returns List<Product> to display product that has been added recently

22. Repository returns a Map that contains *key:value* pair to display the categories of product as well as quantity of each category

23. Controller adds data to Model

24. Thymeleaf renders dashboard.html