**EXERCISE 3**

**1. Path Variables**

**Fetch a Book by Its ID Using a Path Variable:**

To fetch a book by its ID, you can use a path variable in the URL. Here’s how to implement it in the BookController:

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

return books.stream()

.filter(book -> book.getId().equals(id))

.findFirst()

.orElse(null); // Return null or throw a custom exception if the book is not found

}

* @PathVariable Long id: This annotation maps the id from the URL path to the method parameter.
* URL Example: /books/1 would fetch the book with ID 1.

**2. Query Parameters**

**Filter Books Based on Query Parameters like Title and Author:**

You can use query parameters to filter books by title, author, or both. Here's how to implement it:

@GetMapping("/search")

public List<Book> searchBooks(@RequestParam(required = false) String title,

@RequestParam(required = false) String author) {

return books.stream()

.filter(book -> (title == null || book.getTitle().equalsIgnoreCase(title)) &&

(author == null || book.getAuthor().equalsIgnoreCase(author)))

.collect(Collectors.toList());

}

* @RequestParam(required = false) String title: This annotation maps the title query parameter to the method parameter. The required = false means it's optional.
* Filtering Logic: The method filters books by title and author. If either parameter is not provided, it won't be included in the filtering criteria.