**WEEK 1**

**EXERCISE 15**

**Online Bookstore - API Documentation with Swagger**

**Business Scenario**

The task is to document bookstore's REST APIs using Swagger and Springdoc. This allows developers to easily understand and interact with the API through a user-friendly interface.

**Instructions**

**1. Add Swagger Dependency**

**Task:** Include Swagger or Springdoc dependencies in your project.

**Implementation:**

**For Maven:**

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-ui</artifactId>

<version>1.5.9</version>

</dependency>

**For Gradle:**

implementation 'org.springdoc:springdoc-openapi-ui:1.5.9'

* **Explanation:** The springdoc-openapi-ui dependency integrates Swagger UI with Spring Boot, allowing you to automatically generate interactive API documentation.

**2. Document Endpoints**

**Task:** Annotate the REST controllers and methods to generate API documentation.

**Implementation:**

1. **Annotate Controller Methods:**

* **BookController.java:**

import io.swagger.v3.oas.annotations.Operation;

import io.swagger.v3.oas.annotations.responses.ApiResponse;

import io.swagger.v3.oas.annotations.responses.ApiResponses;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/books")

public class BookController {

@Operation(summary = "Get a book by its ID")

@ApiResponses(value = {

@ApiResponse(responseCode = "200", description = "Found the book"),

@ApiResponse(responseCode = "404", description = "Book not found")

})

@GetMapping("/{id}")

public ResponseEntity<BookDTO> getBookById(@PathVariable Long id) {

// Implementation here...

}

@Operation(summary = "Create a new book")

@PostMapping

public ResponseEntity<BookDTO> createBook(@RequestBody BookDTO bookDTO) {

// Implementation here...

}

@Operation(summary = "Update an existing book")

@PutMapping("/{id}")

public ResponseEntity<BookDTO> updateBook(@PathVariable Long id, @RequestBody BookDTO bookDTO) {

// Implementation here...

}

@Operation(summary = "Delete a book by its ID")

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteBook(@PathVariable Long id) {

// Implementation here...

}

}

**Explanation:**

The @Operation annotation is used to describe the purpose of each API method, while @ApiResponses provides information about the possible HTTP responses. This helps in generating comprehensive API documentation.

**3. API Documentation**

**Task:** Generate and review the API documentation using Swagger UI or Springdoc UI.

**Implementation:**

1. **Access Swagger UI:**

After starting the Spring Boot application, navigate to http://localhost:8080/swagger-ui.html to view the automatically generated API documentation.

1. **Review Documentation:**

Swagger UI will display all your REST endpoints, complete with descriptions, parameters, and response types. we can interact with the API directly from this interface.

**Conclusion:**

By documenting the APIs with Swagger and Springdoc, we make the bookstore's REST services easier to understand and use. Swagger UI provides a powerful tool for developers to explore and test the API in a user-friendly manner.