Lab Exercise 10- Documenting a Spring Boot API Using Swagger (OpenAPI 3.0)

---

1. Create a New Spring Boot Project

Go to: https://start.spring.io/

Choose:

Setting	Value
Project	Maven
Language	Java
Spring Boot	3.x
Group	com.example
Artifact	swaggerapi
Dependencies	Spring Web

Download project → Unzip → Open in IDE.

---

2. Add Swagger Dependency

Edit pom.xml and add:

```
<dependency>

   <groupId>org.springdoc</groupId>

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

   <version>2.3.0</version>

</dependency>
```

This automatically enables:

- Swagger UI
- OpenAPI 3 documentation
- JSON/YAML endpoints

---

3. Create the Model Class

File: src/main/java/com/example/swaggerapi/model/User.java

```
package com.example.swaggerapi.model;

import io.swagger.v3.oas.annotations.media.Schema;

@Schema(description = "User entity representing a system user")
public class User {

   @Schema(description = "Unique ID of the user", example = "1")
   private Long id;

   @Schema(description = "Full name of the user", example = "Alice")
```

```
    private String name;

    @Schema(description = "Email address of the user", example =
"alice@example.com")
    private String email;

    public User() {}

    public User(Long id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }

    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
}
```

4. Create the Service Class

File: src/main/java/com/example/swaggerapi/service/UserService.java

```
package com.example.swaggerapi.service;


import com.example.swaggerapi.model.User;

import org.springframework.stereotype.Service;


import java.util.*;

import java.util.concurrent.atomic.AtomicLong;


@Service

public class UserService {


    private final Map<Long, User> store = new HashMap<>();

    private final AtomicLong idGen = new AtomicLong(1);


    public UserService() {

        save(new User(null, "Alice", "alice@example.com"));

        save(new User(null, "Bob", "bob@example.com"));

    }


    public List<User> findAll() {

        return new ArrayList<>(store.values());

    }
    public User findById(Long id) {
        return store.get(id);
    }

    public User save(User user) {
        if (user.getId() == null) {
            user.setId(idGen.getAndIncrement());
        }
        store.put(user.getId(), user);
        return user;
    }

    public boolean delete(Long id) {
        return store.remove(id) != null;
    }

    public boolean exists(Long id) {
        return store.containsKey(id);
    }
}
```

5. Create the Controller With Full Swagger Documentation

File:

src/main/java/com/example/swaggerapi/controller/UserController.java

```
package com.example.swaggerapi.controller;


import com.example.swaggerapi.model.User;

import com.example.swaggerapi.service.UserService;


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

import io.swagger.v3.oas.annotations.tags.Tag;

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

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


import org.springframework.http.ResponseEntity;

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


import java.net.URI;

import java.util.List;


@Tag(name = "User API", description = "CRUD operations for user management")

@RestController

@RequestMapping("/api/users")

public class UserController {
```

}

```
    private final UserService svc;
    public UserController(UserService svc) { this.svc = svc; }


    @Operation(summary = "Get all users", description = "Retrieves all users from the
system")
    @ApiResponses({
            @ApiResponse(responseCode = "200", description = "Successfully retrieved")
    })
    @GetMapping
    public List<User> list() {
        return svc.findAll();
    }


    @Operation(summary = "Get user by ID")
    @ApiResponses({
            @ApiResponse(responseCode = "200", description = "User found"),
            @ApiResponse(responseCode = "404", description = "User not found")
    })
    @GetMapping("/{id}")
    public ResponseEntity<User> get(@PathVariable Long id) {
        User u = svc.findById(id);
        return (u == null) ? ResponseEntity.notFound().build() : ResponseEntity.ok(u);
    }
```

```
  @Operation(summary = "Create a new user")
  @ApiResponses({
      @ApiResponse(responseCode = "201", description = "User created successfully")
  })
  @PostMapping
  public ResponseEntity<User> create(@RequestBody User user) {
    User created = svc.save(user);
    return ResponseEntity.created(URI.create("/api/users/" +
created.getId())).body(created);
  }

  @Operation(summary = "Update an existing user")
  @ApiResponses({
      @ApiResponse(responseCode = "200", description = "User updated"),
      @ApiResponse(responseCode = "404", description = "User not found")
  })
  @PutMapping("/{id}")
  public ResponseEntity<User> update(@PathVariable Long id, @RequestBody User
user) {
    if (!svc.exists(id)) return ResponseEntity.notFound().build();
    user.setId(id);
    return ResponseEntity.ok(svc.save(user));
  }
```

```
   @Operation(summary = "Delete a user")
   @ApiResponses({
           @ApiResponse(responseCode = "204", description = "User deleted"),
           @ApiResponse(responseCode = "404", description = "User not found")
   })
   @DeleteMapping("/{id}")
   public ResponseEntity<Void> delete(@PathVariable Long id) {
      if (!svc.exists(id)) return ResponseEntity.notFound().build();
      svc.delete(id);
      return ResponseEntity.noContent().build();
   }
}
```

---

6. Add OpenAPI Configuration (Optional but Recommended)

File: src/main/java/com/example/swaggerapi/config/OpenAPIConfig.java

```
package com.example.swaggerapi.config;


import io.swagger.v3.oas.models.info.Info;
import io.swagger.v3.oas.models.OpenAPI;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;


@Configuration
public class OpenAPIConfig {
    @Bean
    public OpenAPI apiDetails() {
        return new OpenAPI()
                .info(new Info()
                        .title("User Management API")
                        .description("API documentation using Swagger (OpenAPI 3.0)")
                        .version("1.0.0"));
    }
}
```

---

7. Run the Application

Use:

```
mvn spring-boot:run
```

---

8. Access Swagger UI

Open browser:

```
http://localhost:8080/swagger-ui.html
```

or

```
http://localhost:8080/swagger-ui/index.html
```

You will see:

- All endpoints
- Model schemas

- Request/Response examples
- Try-it-out buttons

---

9. View OpenAPI JSON/YAML

JSON

```
http://localhost:8080/v3/api-docs
```

YAML

```
http://localhost:8080/v3/api-docs.yaml
```

You can download these for:

- Documentation
- Client code generation
- API validation

---

10. Test API Directly in Swagger UI

Use **Try it out** feature:

- GET /api/users
- POST /api/users
- GET /api/users/{id}
- PUT /api/users/{id}
- DELETE /api/users/{id}