**Scenario 15: Online Bookstore - API Documentation with Swagger**

Business Scenario:

Document your bookstore's REST APIs using Swagger and Springdoc.

**Add Swagger Dependency**

To include Swagger or Springdoc in your project, you need to add the following dependencies to your pom.xml file:

**Document Endpoints**

Annotate your REST controllers and methods to generate API documentation.

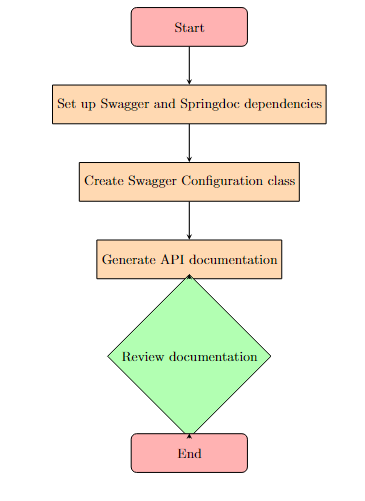
* K**ey Points:**
  + @Tag annotates the class with metadata.
  + @Operation describes the API operation.
  + @ApiResponse specifies the expected response code and description.

**API Documentation**

Run your Spring Boot application and navigate to http://localhost:8080/swagger-ui.html to view the API documentation.

* **Key Points:**
  + Swagger UI is automatically generated based on the annotations and displays interactive API documentation.
  + The Swagger UI page allows you to test the API endpoints directly from the browser.

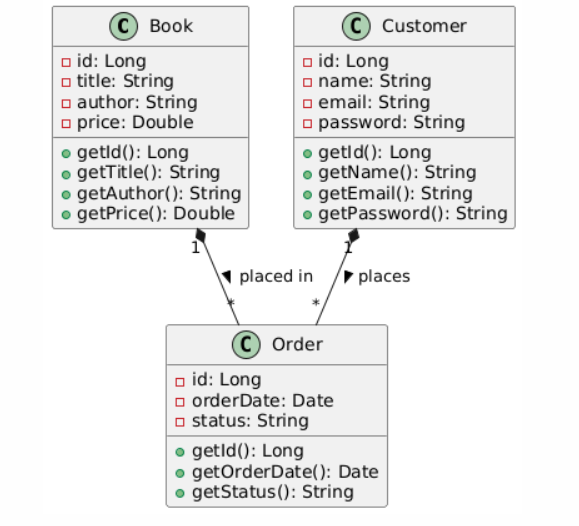
**FLOWCHART :**



**Explanation :**

1. **Start**: The beginning of the process.
2. **Set up Swagger and Springdoc dependencies**: Add the necessary dependencies to your pom.xml or build.gradle file for Swagger and Springdoc to your project.
3. **Create Swagger Configuration class**: Implement a configuration class to set up Swagger, typically by creating a class annotated with @Configuration and using GroupedOpenApi to configure the API documentation.
4. **Generate API documentation**: Use Swagger and Springdoc to generate the API documentation. This usually involves running your Spring Boot application and accessing the Swagger UI to view the documentation.
5. **Review documentation**: Check the generated API documentation for accuracy and completeness.
6. **End**: The completion of the documentation process.

**CLASS DIAGRAM:**



**Explanation :**

1. **Book**:
   * **Attributes**: id, title, author, and price.
   * **Methods**: Getters for each attribute (getId(), getTitle(), getAuthor(), and getPrice()).
2. **Customer**:
   * **Attributes**: id, name, email, and password.
   * **Methods**: Getters for each attribute (getId(), getName(), getEmail(), and getPassword()).
3. **Order**:
   * **Attributes**: id, orderDate, and status.
   * **Methods**: Getters for each attribute (getId(), getOrderDate(), and getStatus()).
4. **Relationships**:
   * **Book to Order**: A one-to-many relationship where one Book can be associated with multiple Order instances.
   * **Customer to Order**: A one-to-many relationship where one Customer can place multiple Order instances.