**Software Requirements Specification (SRS)**

**Advanced Web Sorting Algorithms for Books**

**1. Introduction**

**1.1 Purpose**

This document outlines the software requirements for the Advanced Web Sorting Algorithms project, a RESTful web application that implements various sorting algorithms using Spring Framework, provides CRUD operations for books, and deploys on Apache Tomcat.

**1.2 Scope**

The system will provide a web-based interface for librarian to manage a collection of books and apply sorting algorithms to this collection through a RESTful API.

**1.3 Definitions, Acronyms, and Abbreviations**

* API: Application Programming Interface
* CRUD: Create, Read, Update, Delete
* HATEOAS: Hypermedia as the Engine of Application State
* HTTP: Hypertext Transfer Protocol
* REST: Representational State Transfer
* SRS: Software Requirements Specification

**2. System Overview**

The Advanced Web Sorting Algorithms project is a Spring-based web application that implements CRUD operations for books and sorting algorithms. It shows functionality through RESTful API deployed on an Apache Tomcat web server

**3. Functional Requirements**

3.1 Book Management:

* The system shall allow creating new book
* The system shall allow getting books
* The system shall allow updating books
* The system shall allow deleting book

3.2 The system shall provide a RESTful API for CRUD operations on the books

3.3 The system shall allow librarian to select and execute sorting algorithm on the book dataset.

**4. Non-Functional Requirements**

* Performance
* Scalability
* Security
* Compatibility

**5. System Architecture**

The system will follow a typical Spring MVC architecture:

Client -> Controller -> Service -> Repository -> Database

**6. API Design**

The RESTful API will follow HATEOAS principles and include the following endpoints:

**6.1 Book CRUD Operations**

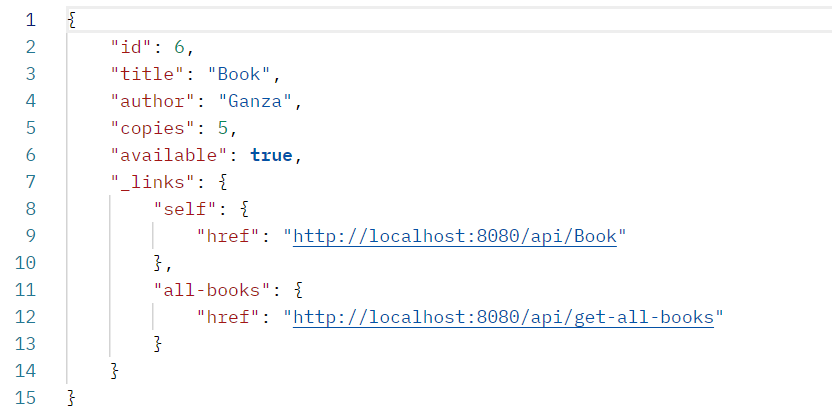
* GET /api/get-all-books: Retrieve all books
* POST /api/create-book: Create a new book
* GET /api/{id}: Retrieve a specific book
* GET /api/{title}: Retrieve a specific book
* PUT /api/{id}: Update a specific book
* DELETE /api/{id}: Delete a specific book

**6.2 Sorting Operations**

* POST /api/sort/quick: Sort the book collection using a specified algorithm

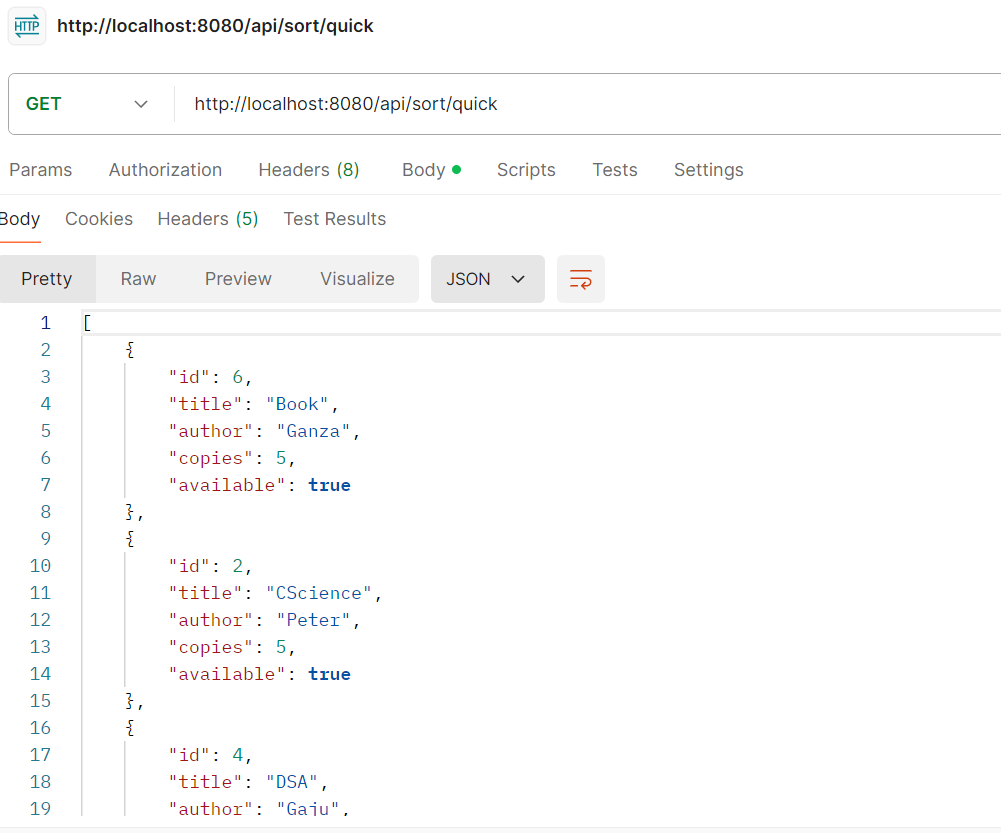
**6.3 Example HATEOAS Response**

Json



**6.4 Sorting Request Example**

Json



**7. Data Flow**

Libririan -> HTTP Request -> Spring Controller -> Book Service

**8. Deployment**

The application was deployed on Apache Tomcat web server. The deployment process include:

Configuring Tomcat for the application.

Setting up the necessary environment variables.

Deploying to the Tomcat webapps directory.

Ensuring proper handling of HTTP protocols.

**9. Data Model**

The Book entity will include the following attributes:

* id (Long): Unique identifier for the book
* title (String): Title of the book
* author (String): Author of the book
* copies (Integer): copies of books
* available (Boolean): Availability of the book

**10. Error Handling**

The API shall return appropriate HTTP status codes for different scenarios ( 200 for success, 404 for not found).

This SRS document provides a comprehensive overview of my Advanced Web Sorting Algorithms project for book management. It covers the main requirements, API design, and other crucial aspects of the system. The document is structured to give a clear understanding of what the system do and how it should be built.