Project Report: RESTful Bookstore API

Author	**Srinivas Jella**
Date	July 2025
Project	RESTful Bookstore API

1. Introduction

The Bookstore API is a RESTful backend service built using the Spring Boot framework. It is designed to provide a comprehensive solution for managing books and authors, featuring full CRUD (Create, Read, Update, Delete) functionality. The API is enhanced with advanced features such as dynamic filtering, sorting, and pagination. To ensure usability and clear documentation, the API is documented using Swagger (OpenAPI). This project simulates the backend system for a simple online bookstore.

2. Abstract

This API manages the data of books and authors stored in a persistent MySQL database, allowing for a complete range of data operations. Key features include filtering book lists by title, genre, and price, and viewing results in a paginated and sorted format. All API endpoints are testable through an interactive Swagger UI and can be consumed by HTTP clients like Postman. The project's clean, layered architecture and use of modern practices make the API easy to maintain and scale.

3. Technology Stack

Tool/Technology	Purpose
Java 17	Core programming language
Spring Boot 3.3.2	Application framework
Spring Data JPA	Data access layer and ORM
MySQL	Relational database
Postman	API testing and validation
Swagger (OpenAPI)	API documentation and UI

Maven	Build and dependency management

4. Architecture & Build Process

1. **Project Setup:** Initialized the project using Spring Initializr with Web, JPA, MySQL, and Swagger dependencies.
2. **Entity & Relationship Mapping:** Defined Book and Author entities using JPA annotations and established the One-to-Many relationship between them.
3. **Repository Layer:** Created JpaRepository interfaces for both entities and extended JpaSpecificationExecutor for the BookRepository to enable dynamic filtering.
4. **Service Layer:** Implemented all business logic, including data validation, safe update patterns ("fetch, then update"), and coordination with the data access layer.
5. **Controller Layer:** Exposed RESTful API endpoints using @RestController, standard URL patterns (e.g., /authors/{id}), and Pageable for pagination.
6. **Advanced Features:**
    - **Filtering:** Built a BookSpecification class for criteria-based dynamic queries.
    - **Error Handling:** Implemented a GlobalExceptionHandler to catch custom exceptions and return structured 404 Not Found responses.
7. **Documentation & Testing:**
    - Configured Swagger to automatically generate interactive API documentation.
    - Tested all endpoints thoroughly with Postman for functionality and edge cases.

5. Conclusion

This project provided significant hands-on experience in building a complete, production-ready REST API using Spring Boot. The key takeaways include a deep understanding of layered architecture, clean REST design, and the practical application of advanced features like pagination and filtering. The process of integrating and troubleshooting tools like Swagger and handling database connections has solidified my backend development skills. The final API is a robust, scalable, and well-documented application that demonstrates a strong command of modern Java development practices.