Restful Bookstore API - Project Report

Introduction:

This project demonstrates a RESTful Bookstore API built with Java and Spring Boot. It provides robust endpoints to manage books and authors, including Create, Read, Update, and Delete (CRUD) operations. Additionally, the system supports search, filtering, pagination, and sorting, making it a practical example of real-world backend API development.

Abstract

The Bookstore API simulates a backend system that can be integrated with frontend applications or mobile apps. It uses Spring Boot for rapid development, Spring Data JPA for database interactions, and an in-memory H2 database to keep the project lightweight. Swagger UI integration ensures that the API endpoints are well documented and easy to test. The project is structured into models, repositories, services, controllers, and configuration classes, following clean architecture practices.

Tools Used

The following tools and technologies were used to build this project: Java 17 – Programming language Spring Boot 3 – Backend framework Spring Data JPA – ORM for database operations H2 Database – In-memory database Swagger / OpenAPI – API documentation Postman – API testing tool.

Steps Involved in Building the Project

The step-by-step process of building the application: Designed entity classes: Author and Book with a one-to-many relationship. Created DTOs for request validation using Jakarta Validation. Implemented repositories for data access using Spring Data JPA. Built service

classes with CRUD operations and query specifications. Developed REST controllers exposing endpoints for managing books and authors. Configured Swagger/OpenAPI for automatic documentation. Added a DataLoader class to insert sample records on startup. Tested all endpoints using Postman collection and verified database changes in H2 console.

Conclusion

The RESTful Bookstore API showcases the power of Spring Boot in building modern Java applications. The project highlights the integration of JPA for persistence, validation, exception handling, and API documentation with Swagger. It provides a strong foundation for future extensions, such as integrating with a MySQL/PostgreSQL database, adding user authentication with Spring Security, or deploying as a cloud-based microservice