

# Project Report: Spring Boot Application

## Overview

This project involves the development of a Spring Boot application featuring two entities, **Books** and **Authors**. The application implements key CRUD operations—Create, Read, and Update—allowing users to manage book and author records. Additionally, JSP pages facilitate user interaction, providing a visually appealing interface.

## 1. Entity Design

### 1.1. Entities

- **Book Entity:**
  - Attributes: `id`, `title`, `author` (Many-to-One relationship with Author).
- **Author Entity:**
  - Attributes: `id`, `name`, `books` (One-to-Many relationship with Book).

### 1.2. JPA Annotations

Entities are annotated with JPA annotations to define relationships:

- `@Entity` for entity classes.
- `@Id` for primary keys.
- `@ManyToOne` and `@OneToMany` for defining relationships.

### 1.3. Database Mapping

Entities are accurately mapped to the database, ensuring data integrity and proper relationship handling.

```

@Entity
@Table(name = "authors")
public class Author {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "name", nullable = false)
    private String name;

    @OneToMany(mappedBy = "author", cascade = CascadeType.ALL)
    @JsonManagedReference
    private List<Book> books;

```

```

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "title", nullable = false)
    private String title;

    @ManyToOne
    @JoinColumn(name = "author_id", nullable = false)
    @JsonBackReference
    private Author author;

```

## 2. CRUD Functionality

### 2.1. Implementation

The application allows users to:

- **Create:** Add new Books and Authors.
- **Read:** Display a list of Books and Authors with an option to view details.
- **Update:** Modify existing records for Books and Authors.

```

17 // Create
18 public Author save(Author author) {
19     return authorRepository.save(author);
20 }
21
22 // Read
23 public List<Author> findAll() {
24     return authorRepository.findAll();
25 }
26
27 public Author findById(Long id) {
28     Optional<Author> author = authorRepository.findById(id);
29     return author.orElse(null); // Return null or throw an exception if not found
30 }
31
32 // Update
33 public Author update(Long id, Author authorDetails) {
34     Author author = findById(id);
35     if (author != null) {
36         author.setName(authorDetails.getName());
37         return authorRepository.save(author);
38     }
39     return null; // Or throw an exception
40 }
41
42 // Delete
43 public void delete(Long id) {
44     authorRepository.deleteById(id);
45 }

```

## 2.2. Custom Queries

A custom query method is implemented to perform an inner join between Books and Authors, enabling efficient data retrieval.

```

// Read
public List<BookDto> findAll() {
    List<Book> books = bookRepository.findAll();
    return books.stream()
        .map(book -> new BookDto(book.getId(), book.getTitle(), book.getAuthor().getName()))
        .collect(Collectors.toList());
}

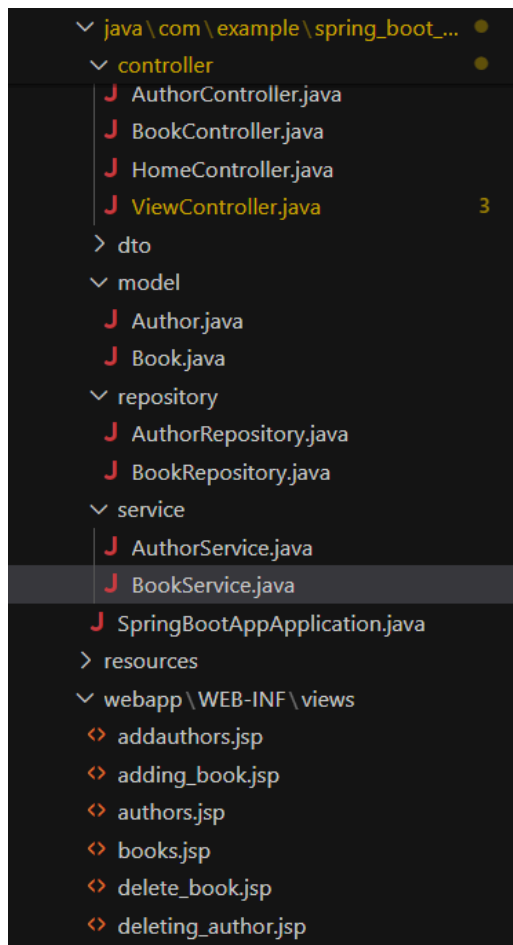
```

## 3. Spring Boot Component Integration

### 3.1. Layer Organization

The application follows a clear separation of concerns:

- **Repository Layer:** Handles database interactions.
- **Service Layer:** Implements business logic and integrates repository calls.
- **Controller Layer:** Manages HTTP requests and responses, binding data to JSP views.



## 3.2. HTTP Request Handling

Controller methods effectively route user requests to the appropriate service methods and prepare data for display on JSP pages.

```
<div class="authors-grid">
  <%
    // Initialize an empty string to hold authors' data
    String apiUrl = "http://localhost:8080/api/authors";
    StringBuilder result = new StringBuilder();

    try {
      // Create a URL object
      URL url = new URL(apiUrl);
      HttpURLConnection conn = (HttpURLConnection) url.openConnection();
      conn.setRequestMethod("GET");
      conn.setRequestProperty("Accept", "application/json");

      // Read the response
      BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
      String inputLine;
      while ((inputLine = in.readLine()) != null) {
        result.append(inputLine);
      }
      in.close();
    }
```

# 4. User Interface

## 4.1. JSP Design

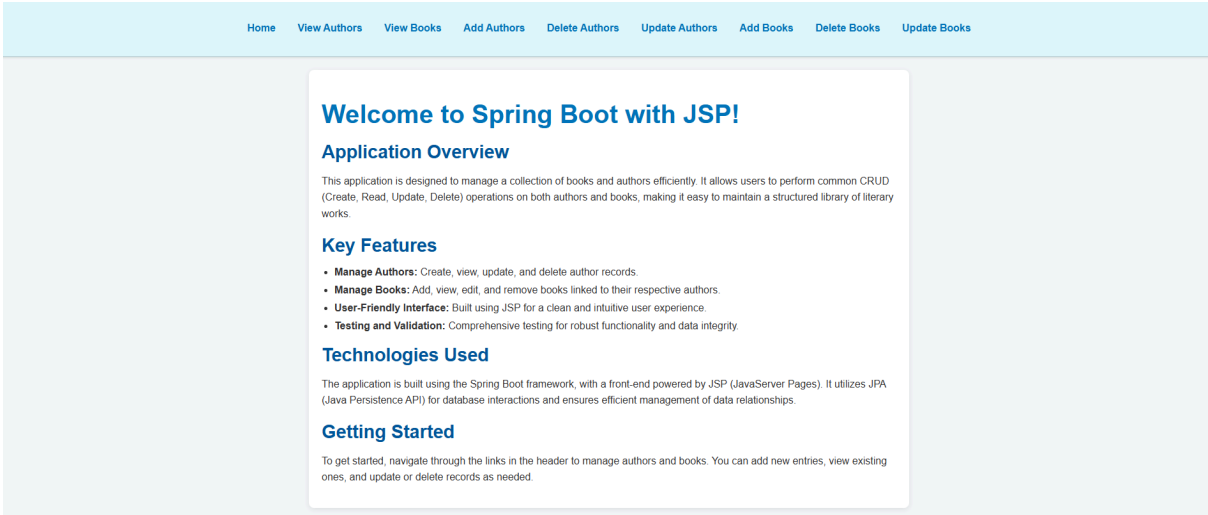
The JSP pages are designed to be user-friendly, featuring:

- Intuitive forms for data entry.
- Clear and organized displays of data.
- CSS styling to enhance visual appeal.

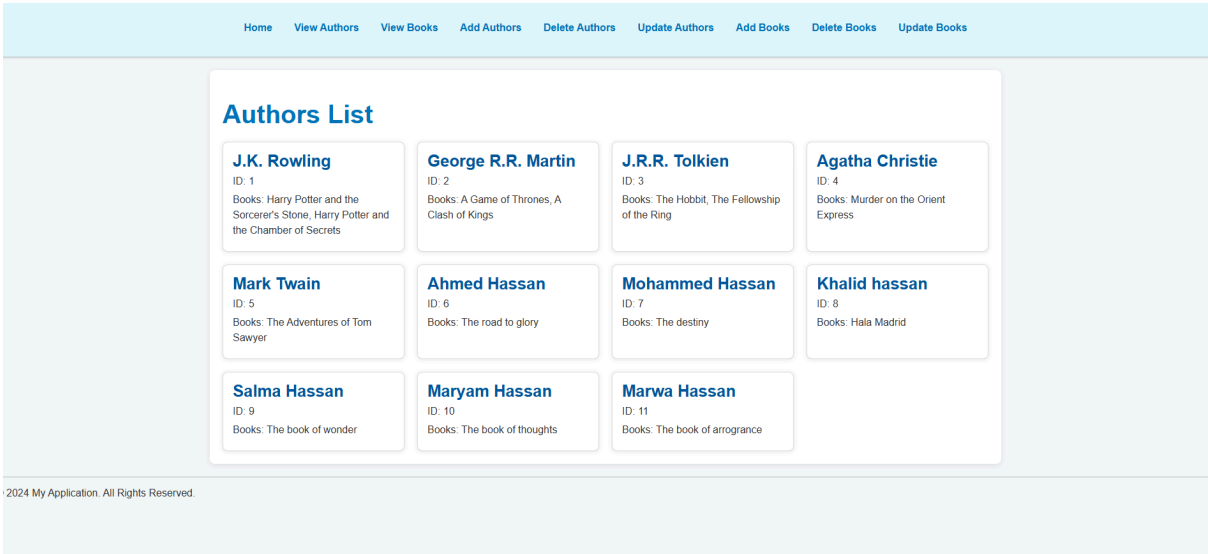
## 4.2. User Experience

The application is evaluated based on ease of navigation and overall user experience, ensuring that users can interact seamlessly with the system.

### 4.2.3 Homepage



### 4.2.3 Authors Page



### 4.2.3 Books Page

HomeView AuthorsView BooksAdd AuthorsDelete AuthorsUpdate AuthorsAdd BooksDelete BooksUpdate Books

Book List

Harry Potter and the Sorcerer's Stone

ID: 1

Author: J.K. Rowling

A Game of Thrones

ID: 2

Author: George R.R. Martin

The Hobbit

ID: 3

Author: J.R.R. Tolkien

Murder on the Orient Express

ID: 4

Author: Agatha Christie

The Adventures of Tom Sawyer

ID: 5

Author: Mark Twain

Harry Potter and the Chamber of Secrets

ID: 6

Author: J.K. Rowling

A Clash of Kings

ID: 7

Author: George R.R. Martin

The Fellowship of the Ring

ID: 8

Author: J.R.R. Tolkien

The road to glory

ID: 9

Author: Ahmed Hassan

The destiny

ID: 10

Author: Mohammed Hassan

Hala Madrid

ID: 11

Author: Khalid hassan

The book of wonder

ID: 12

Author: Salma Hassan

The book of thoughts

ID: 13

Author: Maryam Hassan

The book of arrogance

ID: 14

Author: Manwa Hassan

### 4.2.3 Add Authors Page

HomeView AuthorsView BooksAdd AuthorsDelete AuthorsUpdate AuthorsAdd BooksDelete BooksUpdate Books

Add Author

Author Name:

Enter author's name

Books (comma separated):

Enter book titles separated by commas

Add Author

2024 My Application. All Rights Reserved.

### 4.2.3 Delete Authors Page

HomeView AuthorsView BooksAdd AuthorsDelete AuthorsUpdate AuthorsAdd BooksDelete BooksUpdate Books

Delete Author

Author ID:

Enter author's ID

Delete Author

2024 My Application. All Rights Reserved.

### 4.2.3 Update Authors Page

HomeView AuthorsView BooksAdd AuthorsDelete AuthorsUpdate AuthorsAdd BooksDelete BooksUpdate Books

Update Author

Author ID:

Enter author's ID

Author Name:

Enter author's name

Books (comma separated):

Enter book titles separated by commas

Update Author

2024 My Application. All Rights Reserved.

### 4.2.3 Add Books Page

HomeView AuthorsView BooksAdd AuthorsDelete AuthorsUpdate AuthorsAdd BooksDelete BooksUpdate Books

Add Book

Book Title:

Author Name:

Enter author's name

Add Book

My Application. All Rights Reserved.



### 4.2.3 Delete Books Page

HomeView AuthorsView BooksAdd AuthorsDelete AuthorsUpdate AuthorsAdd BooksDelete BooksUpdate Books

Delete Book

Book ID:

Enter book's ID

Delete Book

© 2024 My Application. All Rights Reserved.

### 4.2.3 Update Books Page

HomeView AuthorsView BooksAdd AuthorsDelete AuthorsUpdate AuthorsAdd BooksDelete BooksUpdate Books

Update Book

Book ID:

Enter book's ID

Book Title:

Author Name:

Enter author's name

Update Book

© 2024 My Application. All Rights Reserved.