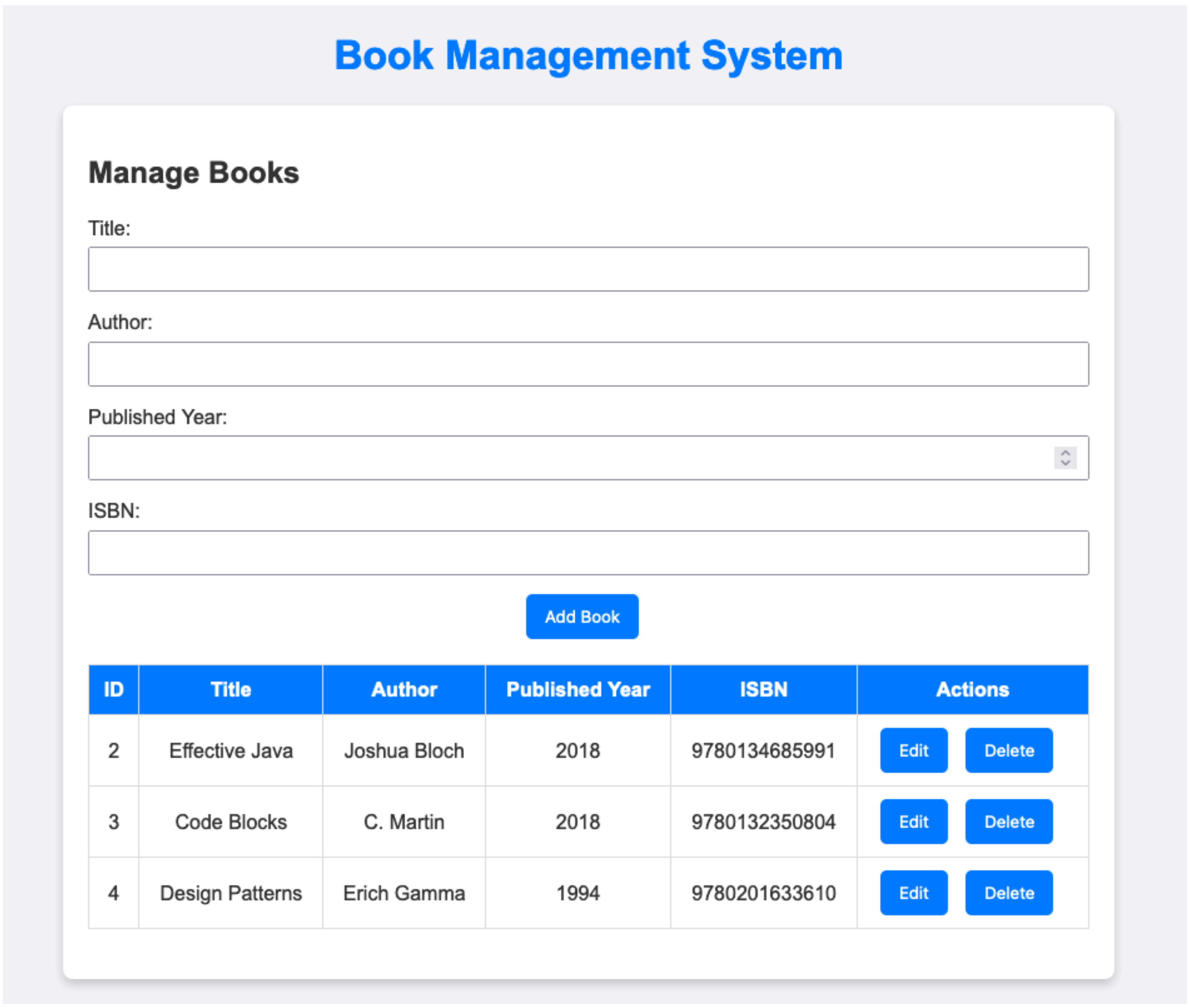
Book Management System: Functionalities and Features



Book Management System

Manage Books

Title:

Author:

Published Year:

ISBN:

ID	Title	Author	Published Year	ISBN	Actions
2	Effective Java	Joshua Bloch	2018	9780134685991	Edit Delete
3	Code Blocks	C. Martin	2018	9780132350804	Edit Delete
4	Design Patterns	Erich Gamma	1994	9780201633610	Edit Delete

Mohamed Zakee

GS/COMP/126

Table of Contents

1. Introduction

The Book Management System is a web application designed to manage books in a library or inventory system. It supports essential CRUD operations such as adding, viewing, editing, and deleting books. The application is built with Spring Boot for the backend and a user-friendly HTML, CSS, and JavaScript interface for the frontend.

2. System Overview

The system allows users to:

- Add new books with details such as title, author, published year, and ISBN.
- View all books in a tabular format.
- Edit existing book details.
- Delete books from the system.

It interacts with a backend API, ensuring seamless data handling and persistence in a MySQL database.

3. Functionalities

Add Books

- Users can fill in the form fields for title, author, published year, and ISBN to add a book.
- Clicking the **Add Book** button sends a POST request to the backend API.

API Endpoint:

POST /api/books

View Books

- The system retrieves all books and displays them in a table format.

- Each row contains the book’s ID, title, author, published year, ISBN, and action buttons.

API Endpoint:

GET /api/books

Edit Books

- Clicking the **Edit** button next to a book pre-fills the form fields with the book’s current details.
- Updating the details and submitting the form sends a PUT request to the backend.

API Endpoint:

PUT /api/books/{id}

Delete Books

- Users can click the **Delete** button to remove a book from the system.
- A DELETE request is sent to the backend, and the list of books is updated.

API Endpoint:

DELETE /api/books/{id}

4. Architecture and Design

System Architecture

Include a diagram showing the system components:

1. Client-side (Browser with HTML/JavaScript).
2. Backend (Spring Boot REST API).
3. Database (MySQL).

Frontend Design

- **HTML**: Structure of the user interface.
- **CSS**: Styling for a user-friendly appearance.
- **JavaScript**: Handles API calls and DOM updates.

Backend Design

- **Spring Boot**: RESTful API implementation with controllers, services, and repositories.

5. Technologies Used

- **Frontend**: HTML, CSS, JavaScript
- **Backend**: Spring Boot
- **Database**: MySQL
- **Tools**: NetBeans, Postman (for testing APIs)

6. API Endpoints

Summarize the available endpoints:

HTTP Method	Endpoint	Description
GET	`/api/books`	Fetch all books
GET	`/api/books/{id}`	Fetch a specific book
POST	`/api/books`	Add a new book
PUT	`/api/books/{id}`	Update an existing book
DELETE	`/api/books/{id}`	Delete a book

7. Testing and Validation

Explain how the system was tested:

1. **Frontend Testing**:
    - Form submission for valid and invalid inputs.
    - UI updates after API responses.
2. **API Testing**:
    - Used Postman to test all API endpoints.
    - Verified data persistence in MySQL.
3. **Error Handling**:
    - Validation for empty or invalid fields.