Book Management System - Output Report

Name: Yash Sudhakar Jadhav

Superset ID: 5298089

College: Imperial College of Engineering and Research, Pune

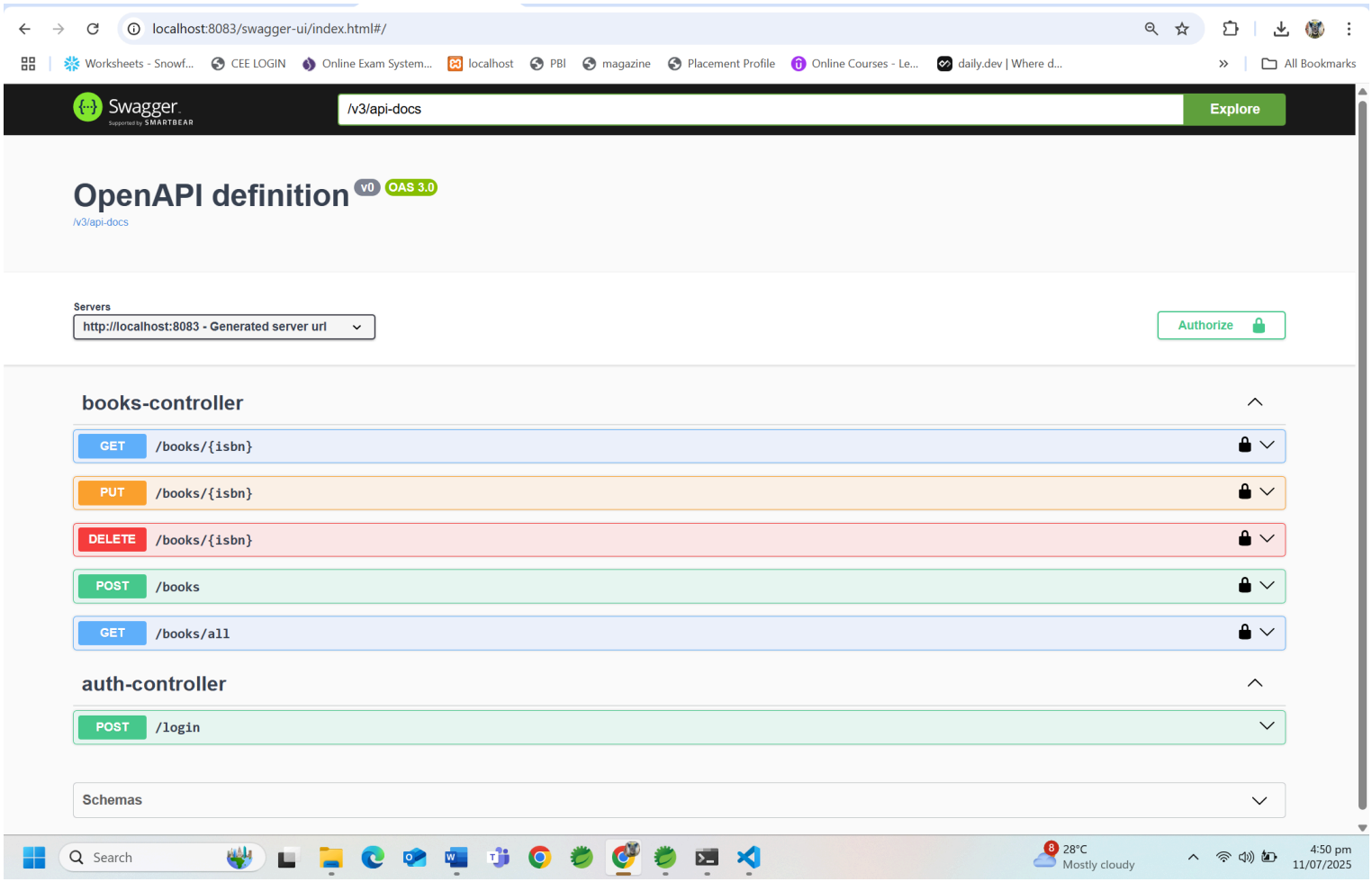
Email: yashj98819@example.com

Date & Time: July 11, 2025 - 05:00 PM

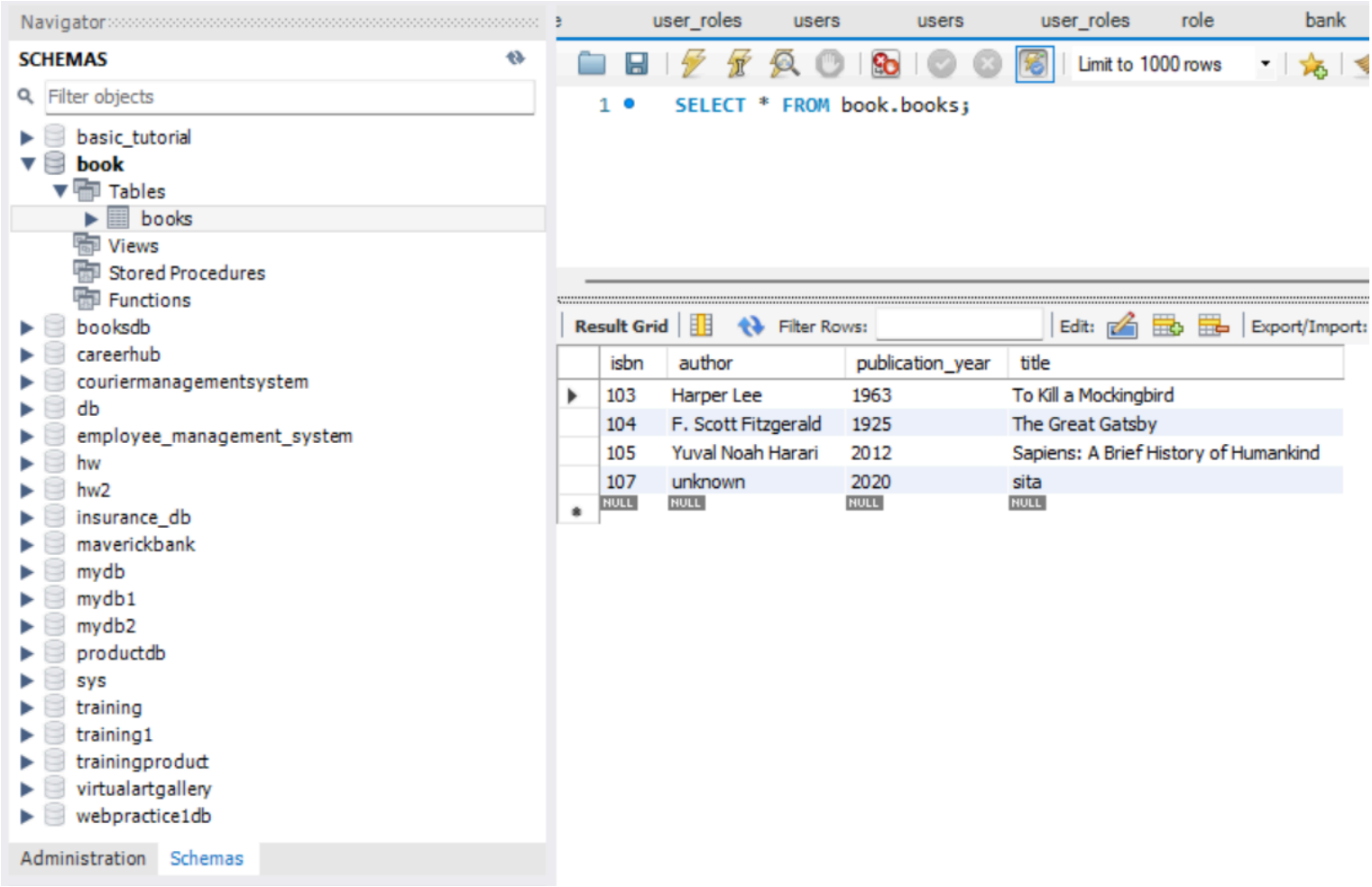
This document contains visual outputs of the Book Management System project, including:

- Swagger/API Screens
- Database Output
- User Interface Screenshots
- **Book Management System - Complete Project Documentation**

- Swagger/API Screens

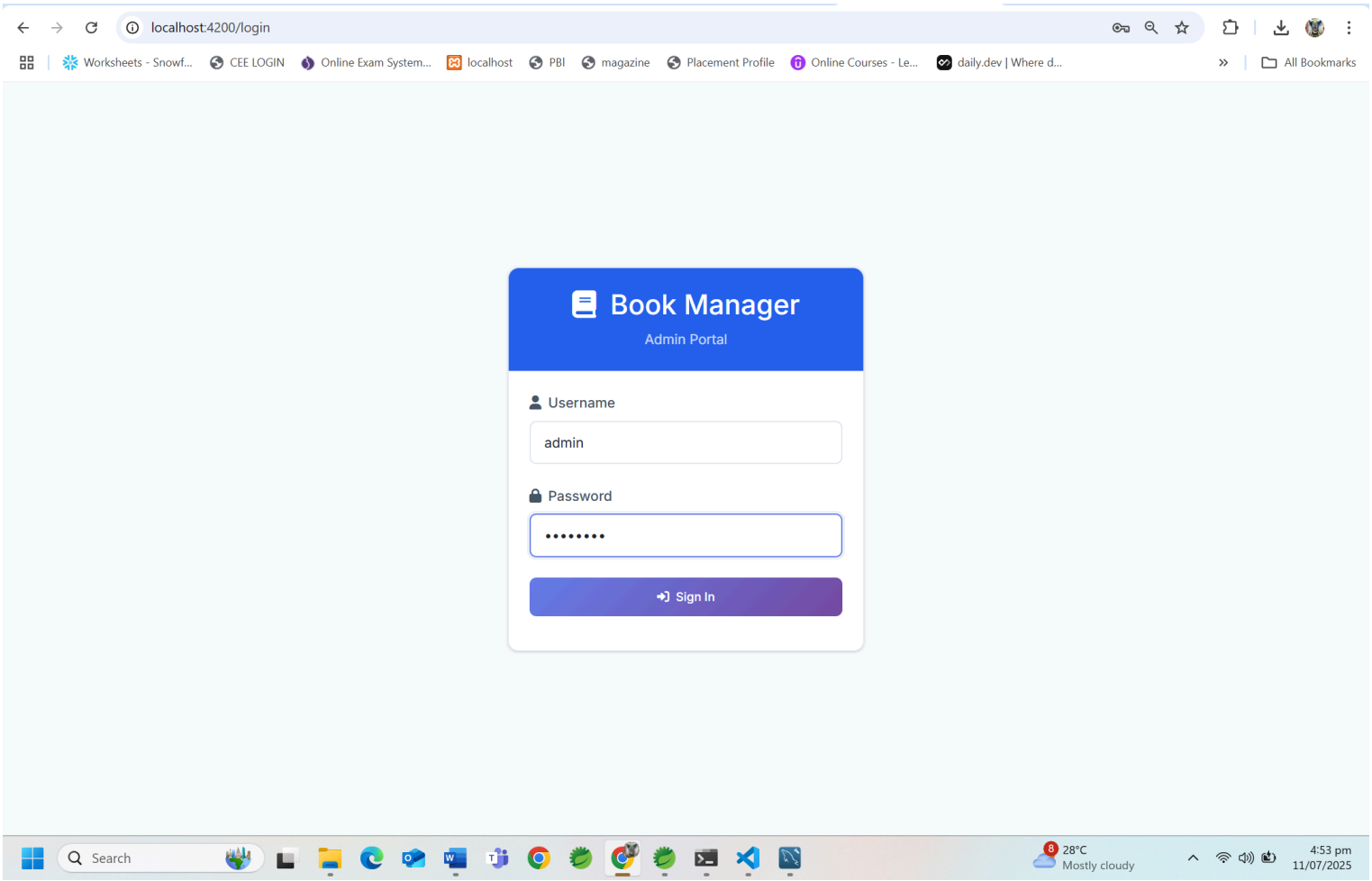


- Database Output

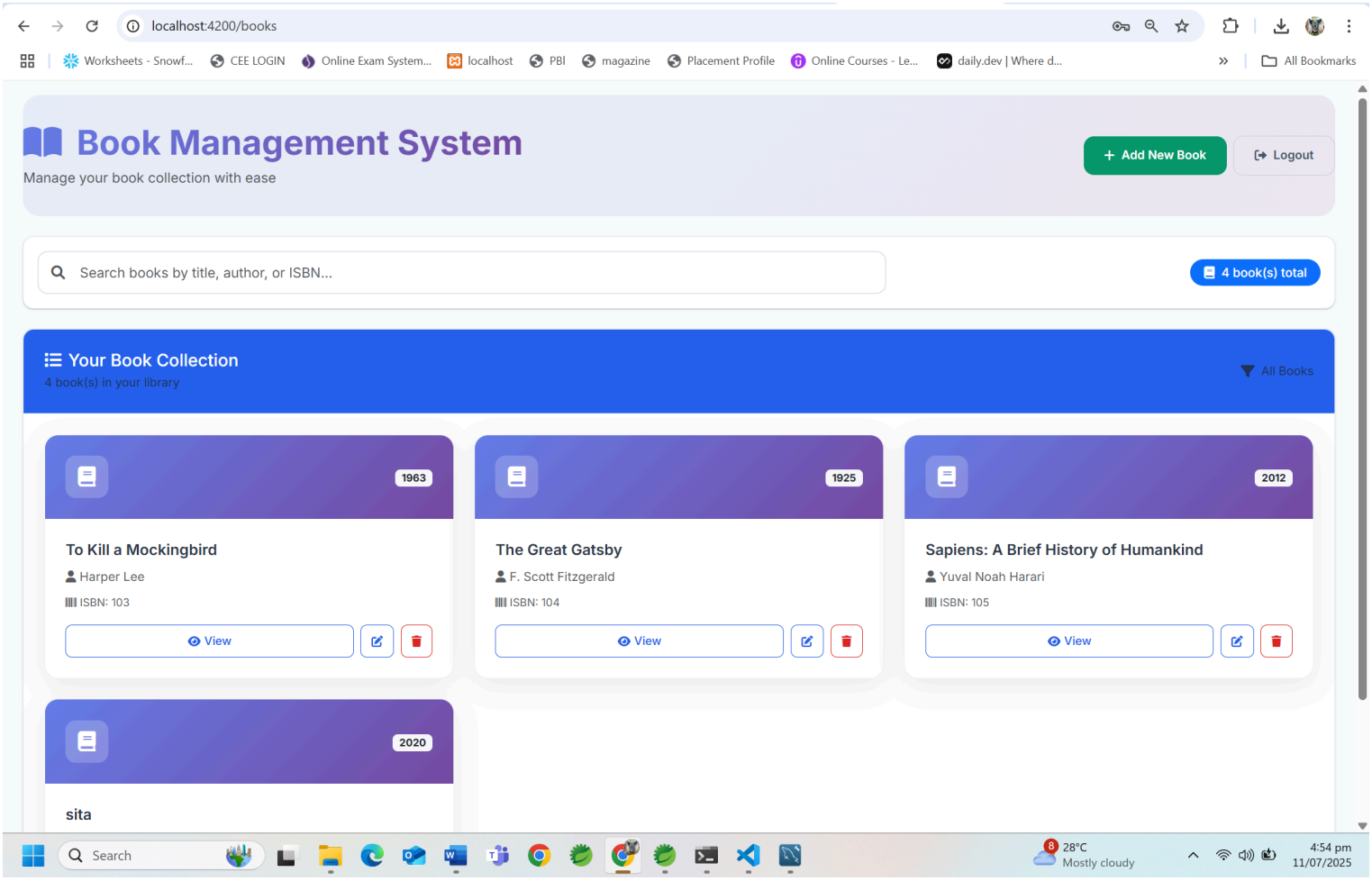


- User Interface Screenshots

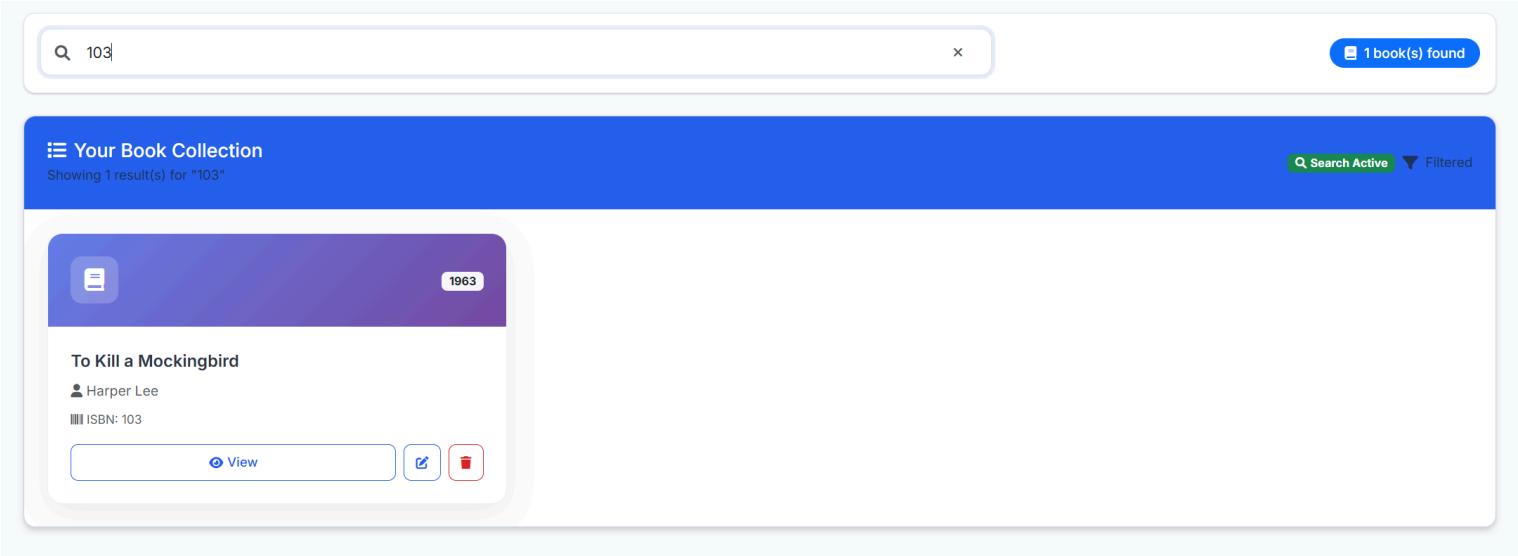
Login page



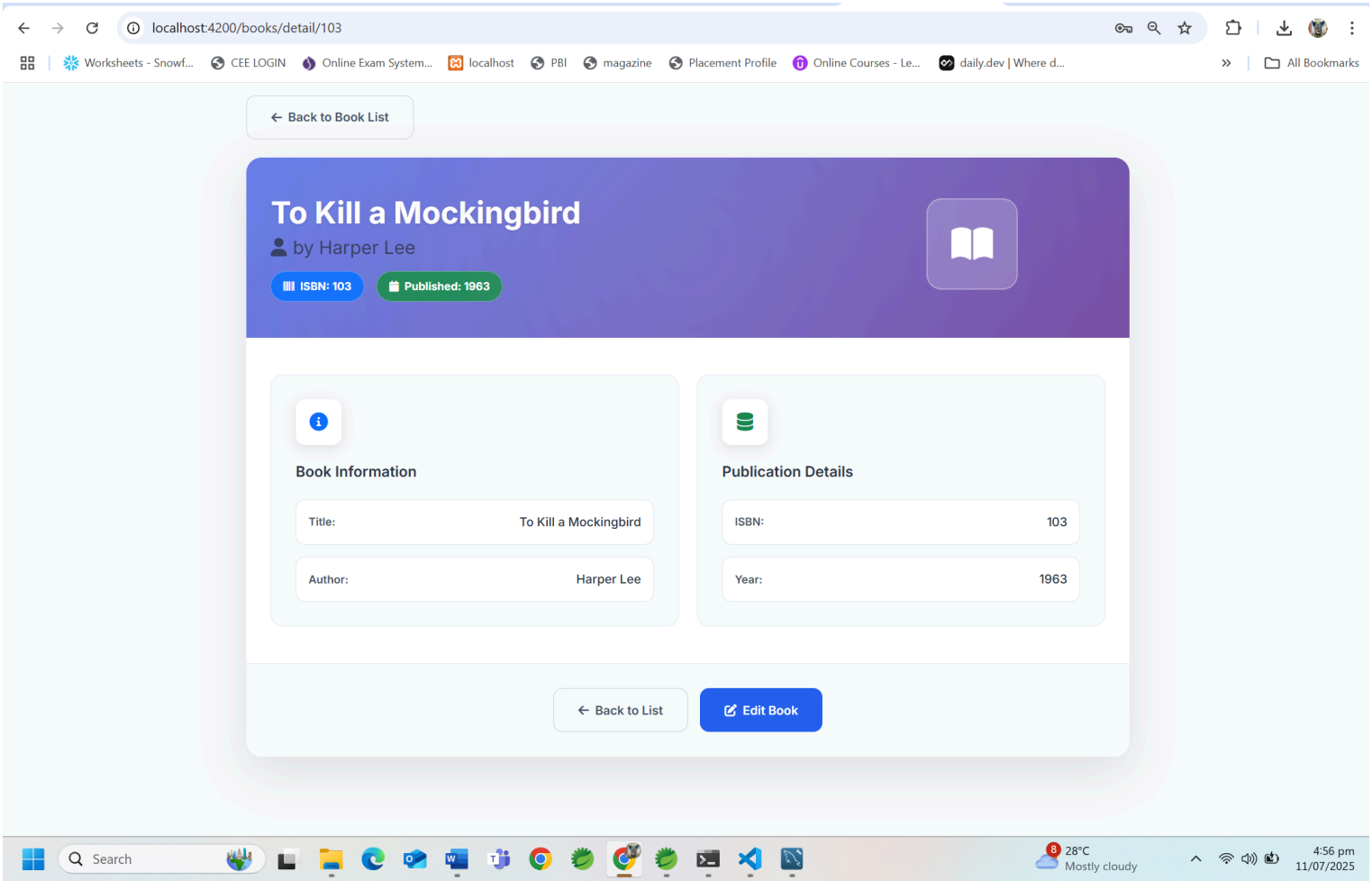
Dashboard


localhost:4200/books
Worksheets - Snowf...
CEE LOGIN
Online Exam System...
localhost
PBI
magazine
Placement Profile
Online Courses - Le...
daily.dev | Where d...
All Bookmarks
Book Management System
Manage your book collection with ease
+ Add New Book
Logout
Search books by title, author, or ISBN...
4 book(s) total
Your Book Collection
4 book(s) in your library
All Books
1963
To Kill a Mockingbird
Harper Lee
ISBN: 103
View
1925
The Great Gatsby
F. Scott Fitzgerald
ISBN: 104
View
2012
Sapiens: A Brief History of Humankind
Yuval Noah Harari
ISBN: 105
View
2020
sita
Search
28°C
Mostly cloudy
4:54 pm
11/07/2025

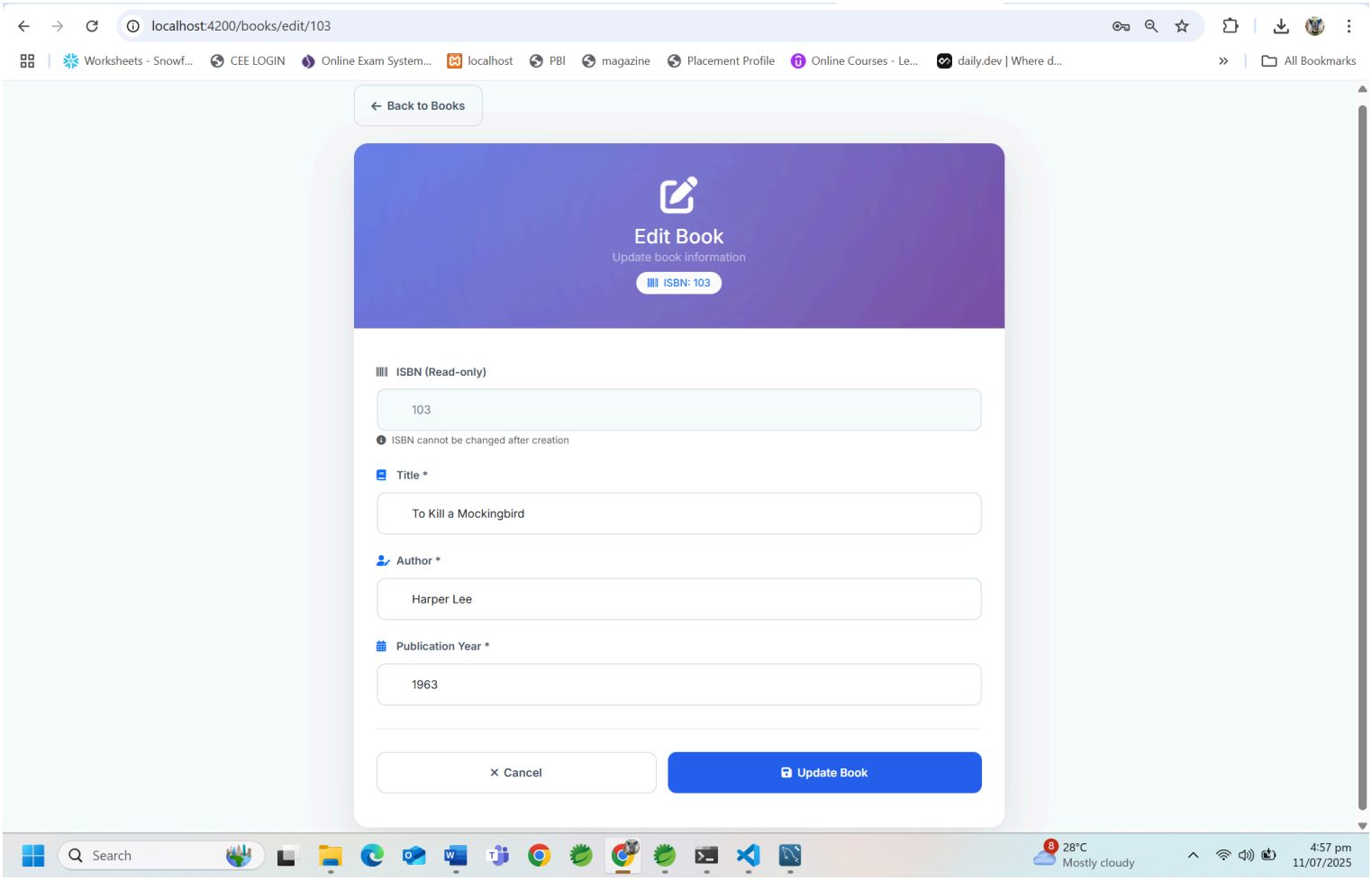
Search by ISBN Number



Book Details



Book Update


localhost:4200/books/edit/103
Worksheets - Snowf...
CEE LOGIN
Online Exam System...
localhost
PBI
magazine
Placement Profile
Online Courses - Le...
daily.dev | Where d...
All Bookmarks
Back to Books
Edit Book
Update book information
ISBN: 103
ISBN (Read-only)
103
ISBN cannot be changed after creation
Title *
To Kill a Mockingbird
Author *
Harper Lee
Publication Year *
1963
Cancel
Update Book
Search
28°C
Mostly cloudy
4:57 pm
11/07/2025

Book Deletion

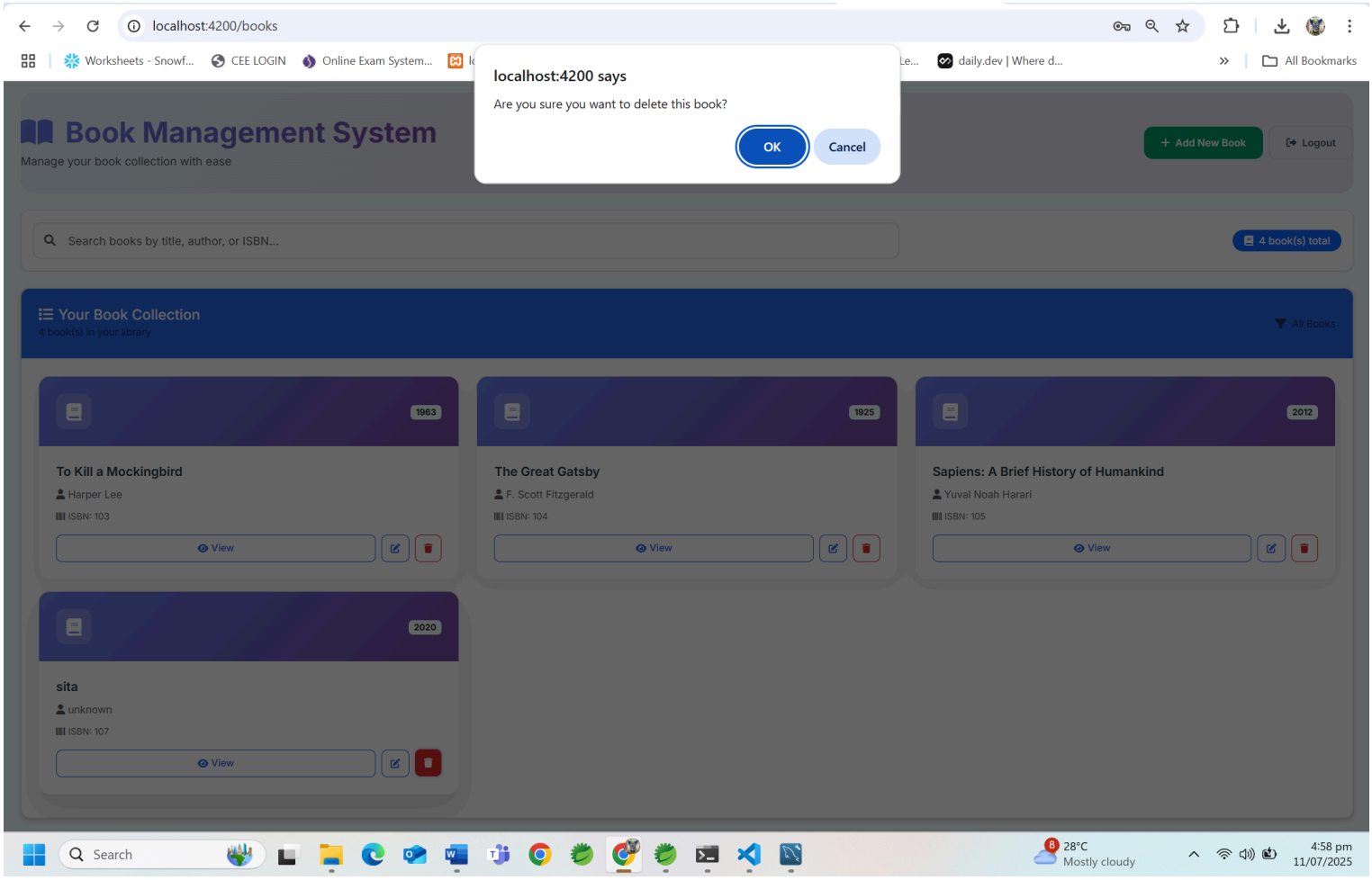


Table of Contents

1. Project Overview
2. System Architecture
3. Technology Stack
4. Backend Implementation
5. Database Design
6. Frontend Implementation
7. API Documentation
8. Security Implementation
9. UI/UX Design
10. Installation and Setup
11. Usage Guide with Screenshots
12. Project Screenshots
13. Testing and Troubleshooting

Project Overview

Project Name

Book Management System - Full Stack Application

Description

A comprehensive full-stack web application for managing book collections with secure authentication, CRUD operations, and modern responsive design. The system includes both frontend (Angular) and backend (Spring Boot) components with MySQL database integration.

Project Goals

- Create a complete full-stack book management solution
- Implement secure JWT-based authentication
- Provide efficient book search and filtering capabilities
- Ensure responsive design for all devices
- Maintain professional, modern UI/UX standards
- Demonstrate full-stack development best practices

Key Benefits

- Full-Stack Solution: Complete frontend and backend integration
- Security: JWT-based authentication with secure API endpoints
- Performance: Optimized database queries and efficient frontend
- Usability: Modern, intuitive interface design
- Scalability: Modular architecture for future enhancements

Technology Highlights

- Frontend: Angular 18+ with TypeScript

- Backend: Spring Boot with Java
- Database: MySQL with JPA/Hibernate
- Authentication: JWT tokens
- API: RESTful web services

System Architecture

Full-Stack Architecture Overview

```
┌─────────────────────────────────────────┐
│         Client Layer (Browser)          │
├─────────────────────────────────────────┤
│  • Angular Application                  │
│  • TypeScript Components                │
│  • Responsive UI/UX                     │
│  • JWT Token Management                 │
└─────────────────────────────────────────┘
                    │
                    │ HTTP/HTTPS Requests
                    │ REST API Calls
                    │
┌─────────────────────────────────────────┐
│       Application Layer (Server)        │
├─────────────────────────────────────────┤
│  • Spring Boot Application              │
│  • REST Controllers                     │
│  • JWT Authentication                   │
│  • Service Layer                        │
│  • Data Validation                      │
└─────────────────────────────────────────┘
                    │
                    │ JPA/Hibernate
                    │ Database Queries
                    │
┌─────────────────────────────────────────┐
│        Data Layer (Database)            │
├─────────────────────────────────────────┤
│  • MySQL Database                       │
│  • Book Entity Table                    │
│  • User Authentication Table            │
│  • Indexes and Relationships            │
└─────────────────────────────────────────┘
```

Data Flow Architecture

14. User Request → Angular Frontend
15. HTTP Request → Spring Boot Backend
16. JWT Validation → Security Layer
17. Business Logic → Service Layer
18. Database Query → MySQL Database
19. Response Processing → JSON Response
20. UI Update → Angular Components

Technology Stack

Frontend Technologies

Technology	Version	Purpose
Angular	18+	Frontend Framework
TypeScript	Latest	Programming Language
HTML5/CSS3	Latest	Markup and Styling
Bootstrap	5.x	UI Framework
Font Awesome	6.0	Icon Library
RxJS	Latest	Reactive Programming

Backend Technologies

Technology	Version	Purpose
Java	17+	Programming Language
Spring Boot	3.x	Backend Framework
Spring Security	6.x	Security Framework
Spring Data JPA	3.x	Data Access Layer
Hibernate	6.x	ORM Framework
Maven	3.x	Build Tool

Database & Infrastructure

Technology	Version	Purpose
MySQL	8.x	Primary Database
JWT	Latest	Authentication Tokens
Tomcat	Embedded	Application Server
Jackson	Latest	JSON Processing

Backend Implementation

Project Structure

```
book-management-backend/
├── src/main/java/com/bookmanagement/
│   ├── controller/
│   │   ├── AuthController.java
│   │   ├── BookController.java
│   │   └── GlobalExceptionHandler.java
│   ├── service/
│   │   ├── AuthService.java
│   │   ├── BookService.java
│   │   └── JwtService.java
│   ├── repository/
│   │   ├── BookRepository.java
│   │   └── UserRepository.java
│   ├── entity/
│   │   ├── Book.java
│   │   ├── User.java
│   │   └── BaseEntity.java
│   ├── dto/
│   │   ├── BookDTO.java
│   │   ├── LoginRequest.java
│   │   └── AuthResponse.java
│   ├── config/
│   │   ├── SecurityConfig.java
│   │   ├── CorsConfig.java
│   │   └── JwtAuthenticationFilter.java
│   └── BookManagementApplication.java
├── src/main/resources/
│   ├── application.properties
│   └── data.sql
└── pom.xml
```

Key Backend Components

1. Book Entity (Book.java)

sts - apicodingNEW/src/main/java/com/example/apicoding1/entity/books.java - Spring Tools for Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- apicoding1 [boot] [devtools]
- apicodingNEW [boot] [devtools]
  - src/main/java
    - com.example.apicoding1
    - com.example.apicoding1.config
      - CorsConfig.java
      - OpenAPIConfig.java
    - com.example.apicoding1.controller
    - com.example.apicoding1.entity
      - books.java
    - com.example.apicoding1.repo
    - com.example.apicoding1.security
      - AuthController.java
      - AuthRequest.java
      - AuthResponse.java
      - JwtAuthenticationFilter.java
      - JwtUtil.java
      - SecurityConfig.java
  - src/main/resources
    - static
    - templates
    - application.properties
  - src/test/java
  - JRE System Library [JavaSE-17]
  - Maven Dependencies
  - src
  - target
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml
  - Untitled.txt
- backend-code [Angular18WithJwtTokenInt
- basicCrudAppDemo [boot] [devtools]
- bookManagementSystem [boot] [devtools]
- CouponService-1 [boot]
- FSD [boot]
- ProductManagementSystem1 [boot] [devtoc
- ProductManagmentSystem2 [boot] [devtool
- ProductService-1 [boot]
- property-booking-app
- springboot-backend [boot] [devtools]

CorsConfig.java | SecurityConfig.java | OpenAPIConfig.java | application.properties | books.java

```java
package com.example.apicoding1.entity;

import jakarta.persistence.Entity;

@Entity
public class books {
    @Id
    private Integer isbn;
    private String title;
    private String author;
    private int publicationYear;
    public books() {
        super();
        // TODO Auto-generated constructor stub
    }
    public books(Integer isbn, String title, String author, int publicationYear) {
        super();
        this.isbn = isbn;
        this.title = title;
        this.author = author;
        this.publicationYear = publicationYear;
    }
    public Integer getIsbn() {
        return isbn;
    }
    public void setIsbn(Integer isbn) {
        this.isbn = isbn;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getAuthor() {
        return author;
    }
    public void setAuthor(String author) {
        this.author = author;
    }
    public int getPublicationYear() {
        return publicationYear;
    }
    public void setPublicationYear(int publicationYear) {
        this.publicationYear = publicationYear;
    }
    @Override
    public String toString() {
        return "book [isbn=" + isbn + ", title=" + title + ", author=" + author + ", publicationYear=" + publicationYear
                + "]";
    }
```

Writable | Smart Insert | 1 : 1 : 0

Search | 28°C Mostly cloudy | 5:05 pm 11/07/2025

2. Book Controller (BookController.java)

sts - apicodingNEW/src/main/java/com/example/apicoding1/controller/BooksController.java - Spring Tools for Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- apicoding1 [boot] [devtools]
- apicodingNEW [boot] [devtools]
  - src/main/java
    - com.example.apicoding1
    - com.example.apicoding1.config
      - CorsConfig.java
      - OpenAPIConfig.java
    - com.example.apicoding1.controller
      - BooksController.java
    - com.example.apicoding1.entity
      - books.java
    - com.example.apicoding1.repo
    - com.example.apicoding1.security
      - AuthController.java
      - AuthRequest.java
      - AuthResponse.java
      - JwtAuthenticationFilter.java
      - JwtUtil.java
      - SecurityConfig.java
  - src/main/resources
    - static
    - templates
    - application.properties
  - src/test/java
  - JRE System Library [JavaSE-17]
  - Maven Dependencies
  - src
  - target
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml
  - Untitled.txt
- backend-code [Angular18WithJwtTokenInt
- basicCrudAppDemo [boot] [devtools]
- bookManagementSystem [boot] [devtools]
- CouponService-1 [boot]
- FSD [boot]
- ProductManagementSystem1 [boot] [devtoc
- ProductManagmentSystem2 [boot] [devtool
- ProductService-1 [boot]
- property-booking-app

CorsConfig.java | SecurityConfig.java | OpenAPIConfig.java | application.properties | books.java | BooksController.java

```java
package com.example.apicoding1.controller;

import java.util.List;

@RestController
@RequestMapping("/books")
@SecurityRequirement(name = "bearerAuth")
public class BooksController {

    @Autowired
    private BooksRepository brepo;

    @PostMapping()
    public books addbooks(@RequestBody books book) {
        return brepo.save(book);
    }

    @GetMapping("/all")
    public List<books> getallbooks() {
        return brepo.findAll();
    }

    @GetMapping("/{isbn}")
    public ResponseEntity<books> getbyid(@PathVariable int isbn){
        Optional<books> bk = brepo.findById(isbn);
        if (bk.isPresent()) {
            return ResponseEntity.ok(bk.get());
        } else {
            return ResponseEntity.notFound().build();
        }
    }


    @PutMapping("/{isbn}")
    public ResponseEntity<books> updatebooks(@PathVariable int isbn,@RequestBody books ubook){
        Optional<books> bookOpt = brepo.findById(isbn);
        if(bookOpt.isPresent()) {
            books bk = bookOpt.get();
            bk.setTitle(ubook.getTitle());
            bk.setAuthor(ubook.getAuthor());
            bk.setPublicationYear(ubook.getPublicationYear());
            return ResponseEntity.ok(brepo.save(bk));
        }else {
            return ResponseEntity.notFound().build();
        }

    }

    @DeleteMapping("/{isbn}")
    public  ResponseEntity<?> deletebooks(@PathVariable int isbn){
        if(brepo.existsById(isbn)) {
            brepo.deleteById(isbn);
```

Writable | Smart Insert | 35 : 24 : 1174

Search

28°C Mostly cloudy

5:06 pm 11/07/2025

3. Security Configuration

sts - apicodingNEW/src/main/java/com/example/apicoding1/security/SecurityConfig.java - Spring Tools for Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- apicoding1 [boot] [devtools]
- apicodingNEW [boot] [devtools]
  - src/main/java
    - com.example.apicoding1
    - com.example.apicoding1.config
      - CorsConfig.java
      - OpenAPIConfig.java
    - com.example.apicoding1.controller
      - BooksController.java
    - com.example.apicoding1.entity
      - books.java
    - com.example.apicoding1.repo
    - com.example.apicoding1.security
      - AuthController.java
      - AuthRequest.java
      - AuthResponse.java
      - JwtAuthenticationFilter.java
      - JwtUtil.java
      - SecurityConfig.java
  - src/main/resources
    - static
    - templates
    - application.properties
  - src/test/java
  - JRE System Library [JavaSE-17]
  - Maven Dependencies
  - src
  - target
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml
  - Untitled.txt
- backend-code [Angular18WithJwtTokenInt
- basicCrudAppDemo [boot] [devtools]
- bookManagementSystem [boot] [devtools]
- CouponService-1 [boot]
- FSD [boot]
- ProductManagementSystem1 [boot] [devtoc
- ProductManagmentSystem2 [boot] [devtool
- ProductService-1 [boot]
- property-booking-app

CorsConfig.java | SecurityConfig.java | OpenAPIConfig.java | application.properties | books.java | BooksController.java | AuthController.java | JwtAuthenticationFilter.java

```java
package com.example.apicoding1.security;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.web.cors.CorsConfigurationSource;

@Configuration
public class SecurityConfig {

    @Autowired
    private JwtAuthenticationFilter jwtFilter;

    @Autowired
    private CorsConfigurationSource corsConfigurationSource; // Use the bean from CorsConfig

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.cors(cors -> cors.configurationSource(corsConfigurationSource))
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/login").permitAll()
                .requestMatchers("/v3/api-docs/**", "/swagger-ui/**", "/swagger-ui.html").permitAll()
                .anyRequest().authenticated()
            )
            .addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }

    // REMOVE the corsConfigurationSource() method - it's already defined in CorsConfig

    @Bean
    public UserDetailsService userDetailsService() {
        return new InMemoryUserDetailsManager(
                User.builder()
                        .username("admin")
                        .password(passwordEncoder().encode("password"))
                        .roles("USER")
                        .build()
        );
    }
```

Writable | Smart Insert | 52 : 6 : 2355

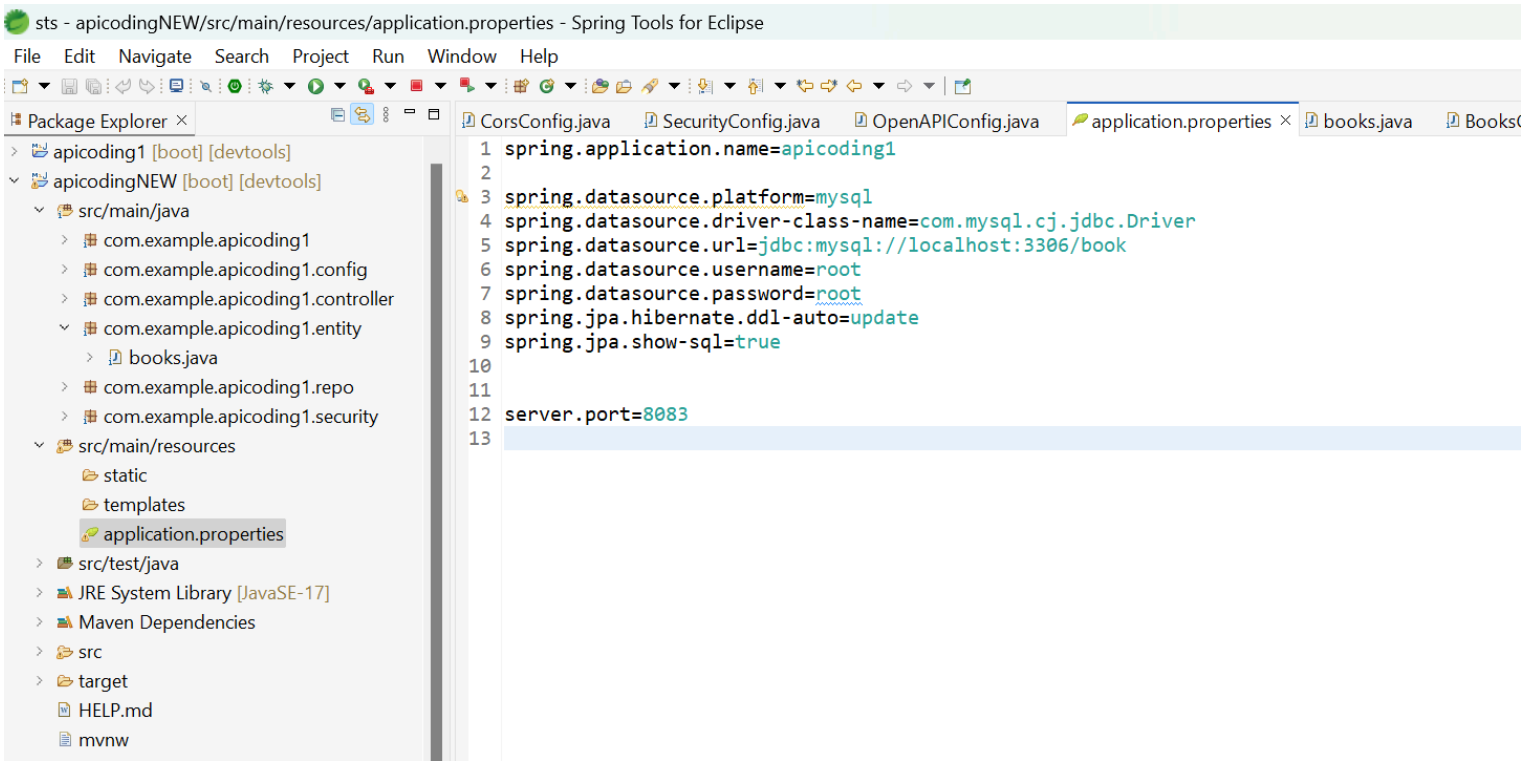
Search | 28°C Mostly cloudy | 5:06 pm 11/07/2025

Backend Features

- RESTful API Design: Standard HTTP methods and status codes
- JWT Authentication: Secure token-based authentication
- CORS Configuration: Cross-origin request handling
- Exception Handling: Global error handling with proper HTTP responses
- Data Validation: Input validation and sanitization
- Logging: Comprehensive logging for debugging and monitoring

Database Configuration

Application Properties



Database Features

- Primary Keys: Unique identifiers for all entities
- Foreign Keys: Proper relationships between tables
- Indexes: Optimized queries for better performance
- Constraints: Data integrity and validation
- Timestamps: Audit trails for data changes

Frontend Implementation

Component Architecture

1. App Component (app.ts)

Purpose: Root application component
- Application initialization and routing setup

2. Authentication Components

- Login Component: User authentication interface
- Auth Guard: Route protection
- JWT Interceptor: Automatic token management

3. Book Management Components

- Book List: Display all books with search functionality
- Book Add: Create new book entries
- Book Update: Edit existing book information
- Book Detail: View detailed book information

4. Services

- Auth Service: Authentication and token management
- Book Service: All book-related API operations

Key Frontend Features

- Responsive Design: Works on all devices

- Real-time Search: Instant book filtering
- Form Validation: Client-side input validation
- Error Handling: User-friendly error messages
- Loading States: Visual feedback during operations

Security Implementation

JWT Authentication Flow

21. User submits credentials to /auth/login
22. Backend validates credentials
23. Backend generates JWT token
24. Frontend stores token in localStorage
25. All API requests include token in Authorization header
26. Backend validates token for each protected endpoint

Security Features

- Password Encryption: BCrypt hashing
- Token Expiration: 24-hour token validity
- CORS Configuration: Controlled cross-origin access
- Input Validation: Server-side data validation
- SQL Injection Prevention: JPA/Hibernate protection

UI/UX Design

Design Philosophy

- Minimalist: Clean, professional interface
- Responsive: Mobile-first design approach
- Accessible: High contrast and readable fonts
- Intuitive: Easy navigation and clear actions

Installation and Setup

Prerequisites

- Java: JDK 17 or higher
- Node.js: Version 18 or higher
- MySQL: Version 8.x
- Maven: Version 3.x
- Angular CLI: Latest version

Access the Application

- Frontend: http://localhost:4200
- Backend API: http://localhost:8083
- Database: localhost:3306/book

Testing and Troubleshooting

Quick Testing Guide

Common Issues

27. CORS Errors: Check backend CORS configuration
28. Authentication Issues: Verify JWT token and credentials
29. Database Connection: Check MySQL service and credentials
30. Port Conflicts: Ensure ports 4200 and 8083 are available

Future Enhancements

Planned Features

- Advanced Search: Filter by publication year range
- Book Categories: Genre-based organization
- User Management: Multiple user support
- Export/Import: CSV functionality

Conclusion

This Book Management System demonstrates a complete full-stack application with:
- Modern Frontend: Angular with TypeScript
- Robust Backend: Spring Boot with MySQL
- Security: JWT authentication
- Professional UI: Responsive design
- Best Practices: Clean code and architecture

The system provides a solid foundation for book collection management with room for future enhancements and scalability.

Document Version: 2.0
Last Updated: July 11, 2025
Project Status: Production Ready
Maintained By: Yash Jadhav