**🎯 API TESTING WITH SWAGGER**

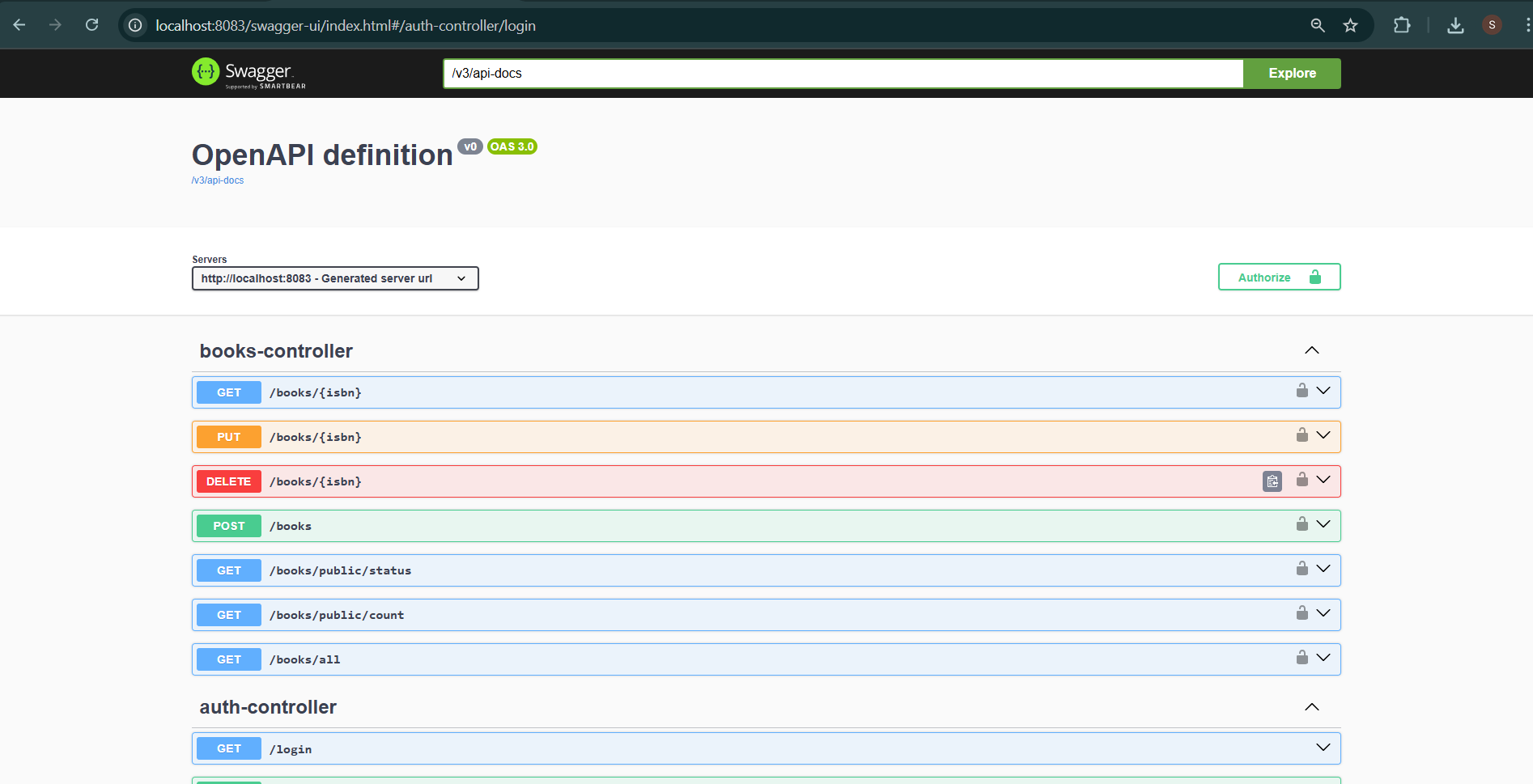
Access Swagger UI at: [**http://localhost:8083/swagger-ui.html**](vscode-file://vscode-app/c:/Users/vsour/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

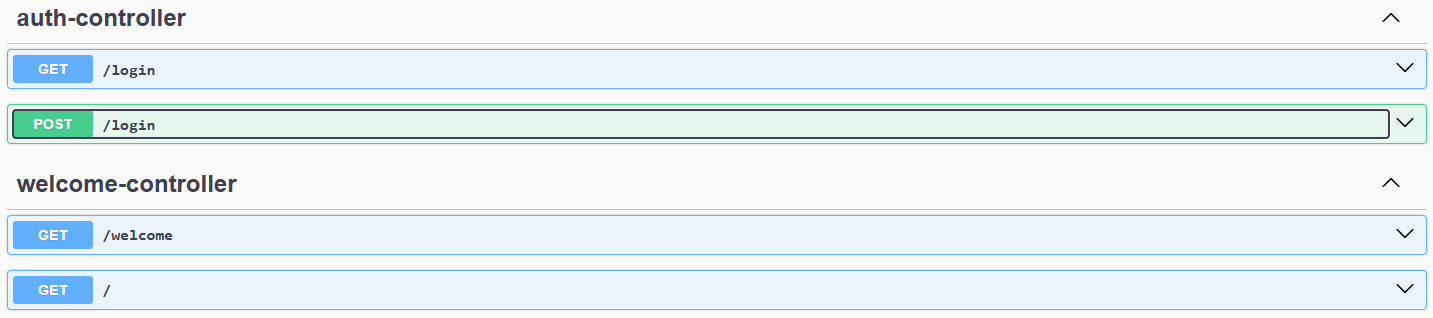
**Available Operations:**

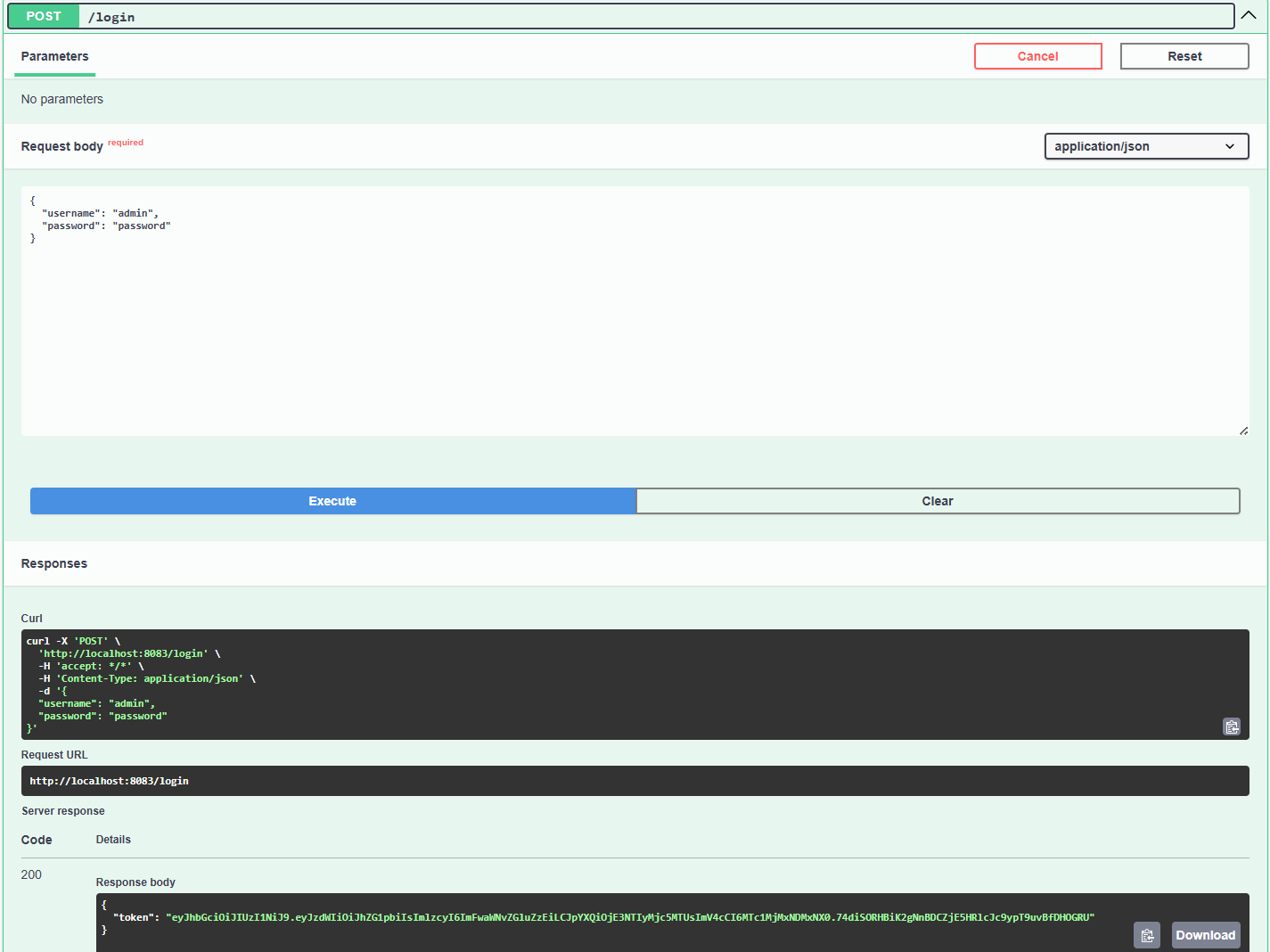
1. **Authentication Controller**
   * POST /login - User authentication
   * GET /login - Login help page
2. **Books Controller**
   * GET /books/all - List all books
   * POST /books - Add new book
   * GET /books/{isbn} - Get book by ISBN
   * PUT /books/{isbn} - Update book
   * DELETE /books/{isbn} - Delete book
   * GET /books/public/status - Health check
   * GET /books/public/count - Total book count

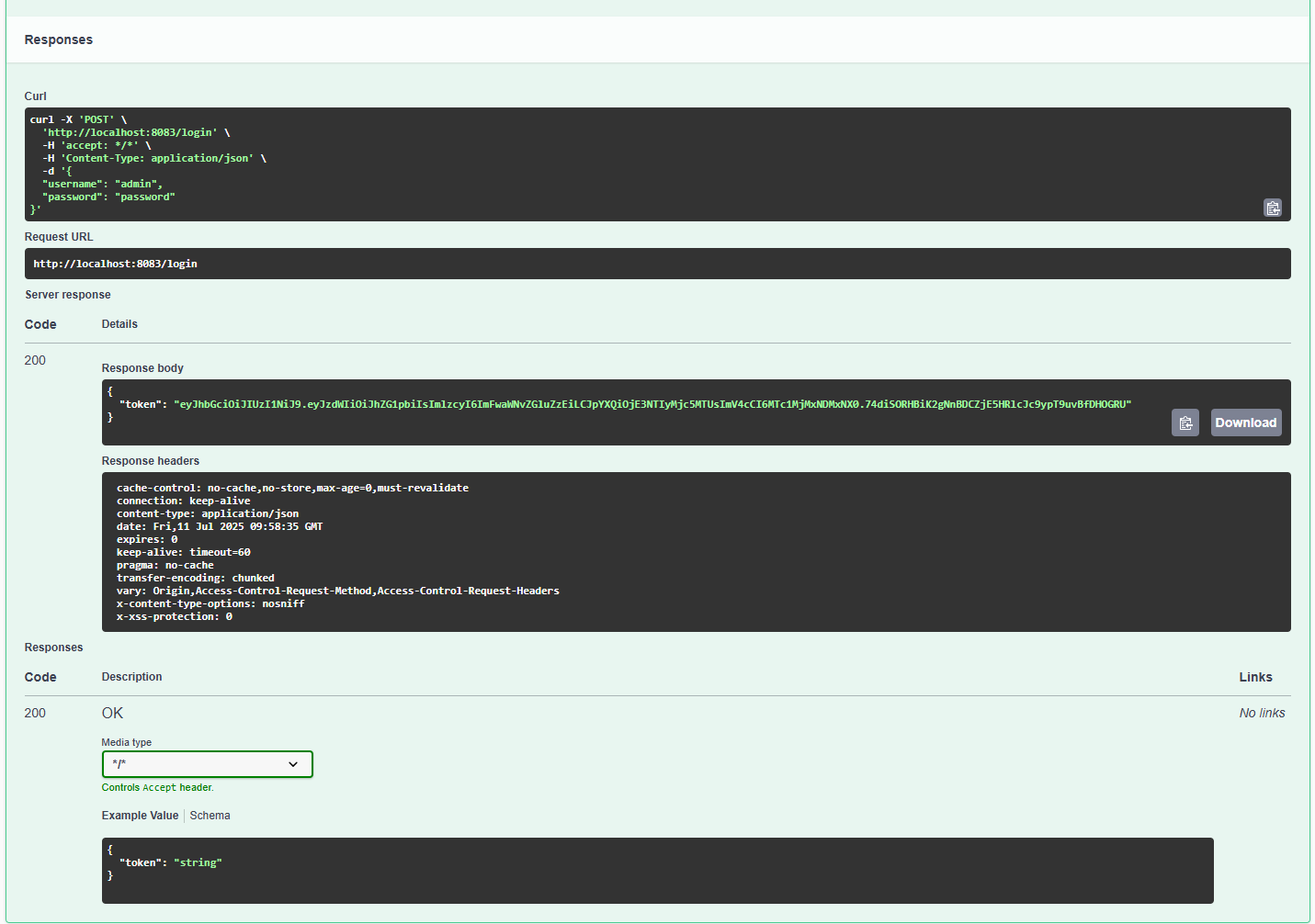
**Using Swagger:**

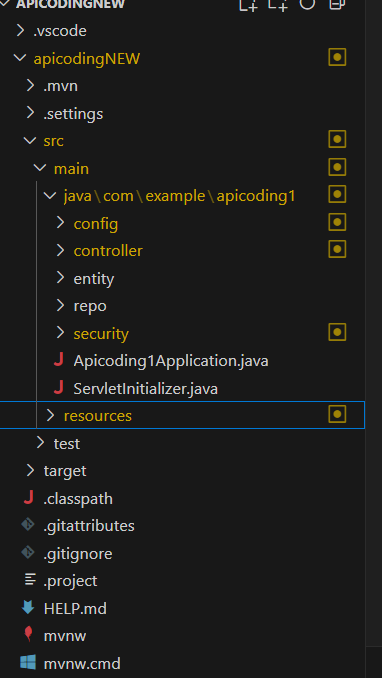
1. Click on an endpoint to expand
2. Click "Try it out"
3. Fill in parameters/request body
4. Click "Execute"
5. View response











**Backend Folder Structure**

apicodingNEW/

├── src/main/java/com/example/apicoding1/

│ ├── Apicoding1Application.java # Main Application Class

│ ├── controller/

│ │ ├── BooksController.java # Books REST API

│ │ └── AuthController.java # Authentication API

│ ├── entity/

│ │ └── books.java # Book Entity Model

│ ├── repo/

│ │ └── BooksRepository.java # Data Repository

│ ├── security/

│ │ ├── SecurityConfig.java # Security Configuration

│ │ ├── JwtAuthenticationFilter.java # JWT Filter

│ │ ├── JwtUtil.java # JWT Utilities

│ │ ├── AuthRequest.java # Login Request DTO

│ │ └── AuthResponse.java # Login Response DTO

│ ├── config/

│ │ ├── OpenAPIConfig.java # Swagger Configuration

│ │ └── CorsConfig.java # CORS Configuration

│ └── data/

│ └── DataInitializer.java # Sample Data Loader

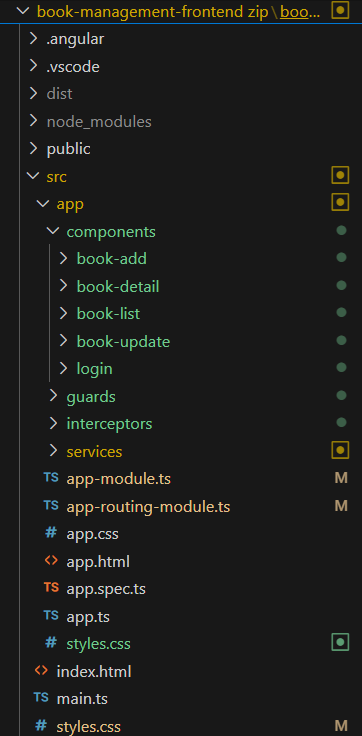
├── src/main/resources/

│ ├── application.properties # Application Configuration

│ └── static/ # Static Resources

├── pom.xml # Maven Dependencies

└── mvnw.cmd # Maven Wrapper

****

**Frontend Folder Structure**

book-management-frontend/

├── src/

│ ├── app/

│ │ ├── components/

│ │ │ ├── login/

│ │ │ │ ├── login.ts # Login Component

│ │ │ │ └── login.html # Login Template

│ │ │ ├── book-list/

│ │ │ │ ├── book-list.ts # Book List Component

│ │ │ │ └── book-list.html # Book List Template

│ │ │ └── book-add/

│ │ │ ├── book-add.ts # Add Book Component

│ │ │ └── book-add.html # Add Book Template

│ │ ├── services/

│ │ │ ├── auth.ts # Authentication Service

│ │ │ └── book.ts # Book API Service

│ │ ├── models/

│ │ │ └── book.model.ts # Book Interface

│ │ ├── interceptors/

│ │ │ └── jwt.interceptor.ts # JWT HTTP Interceptor

│ │ ├── app.module.ts # Main App Module

│ │ ├── app-routing.module.ts # Routing Configuration

│ │ └── app.html # Main App Template

│ ├── assets/

│ │ └── styles.css # Global Styles

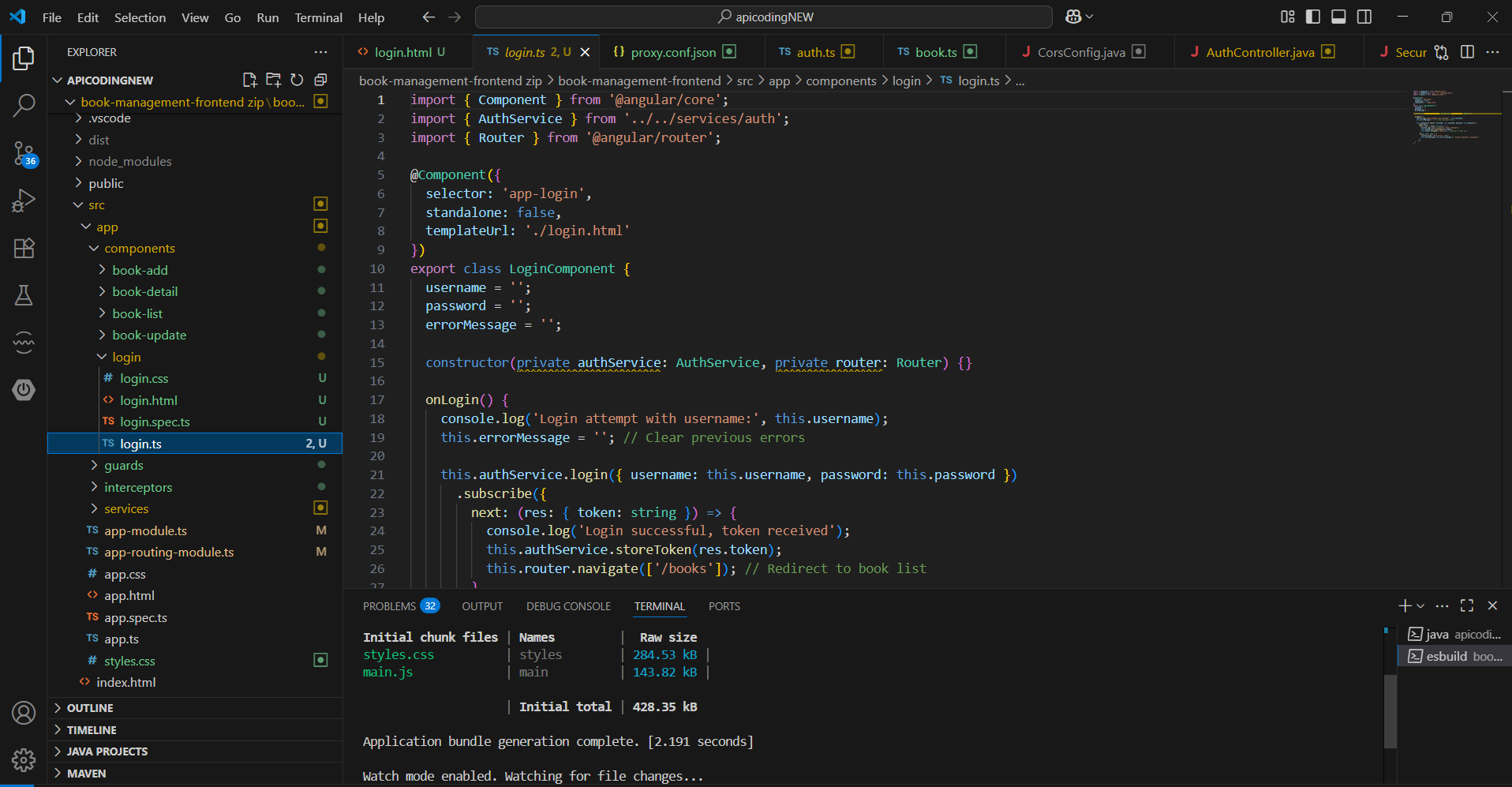
│ └── main.ts # Application Bootstrap

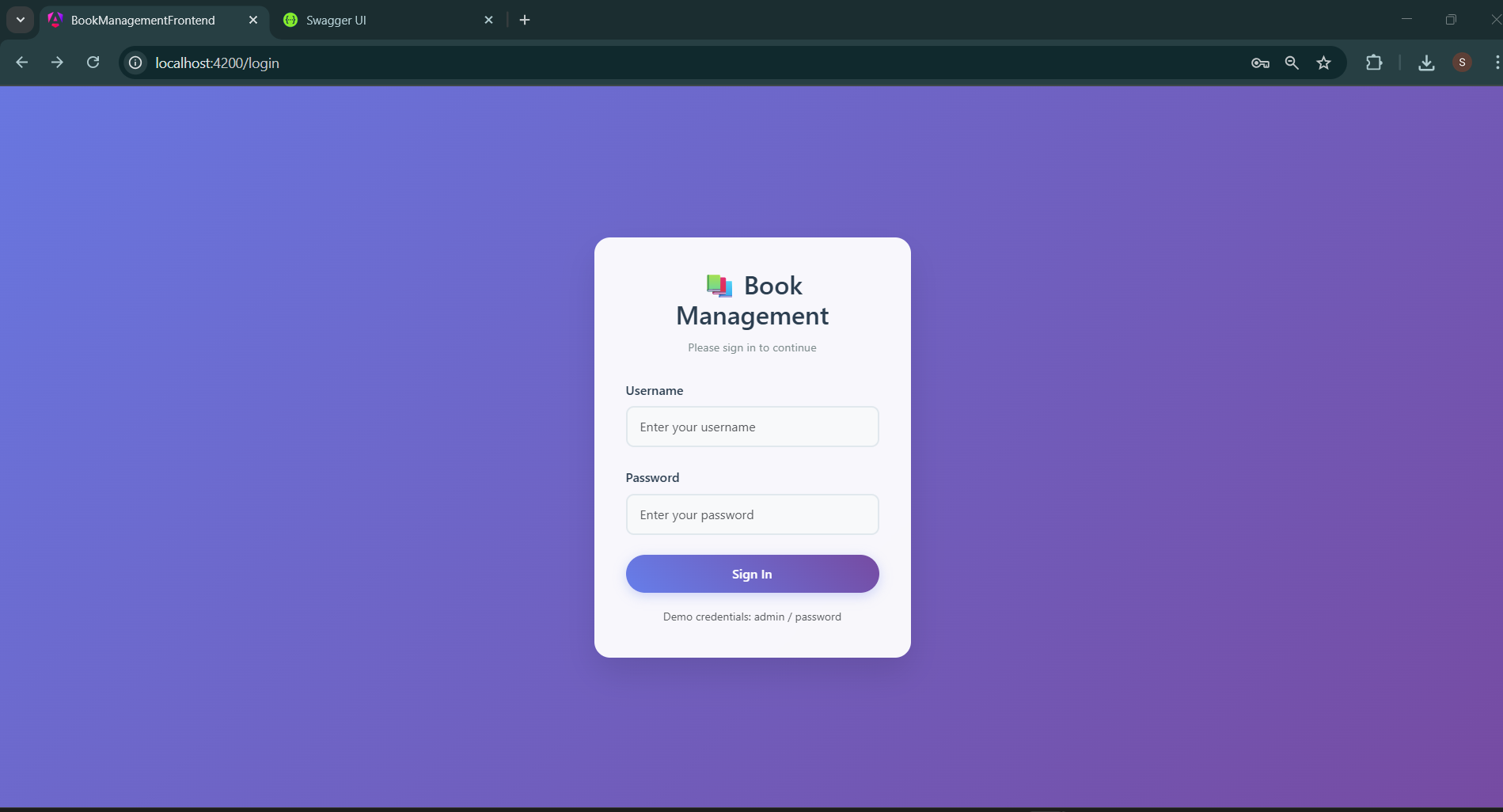
├── proxy.conf.json # Development Proxy Config

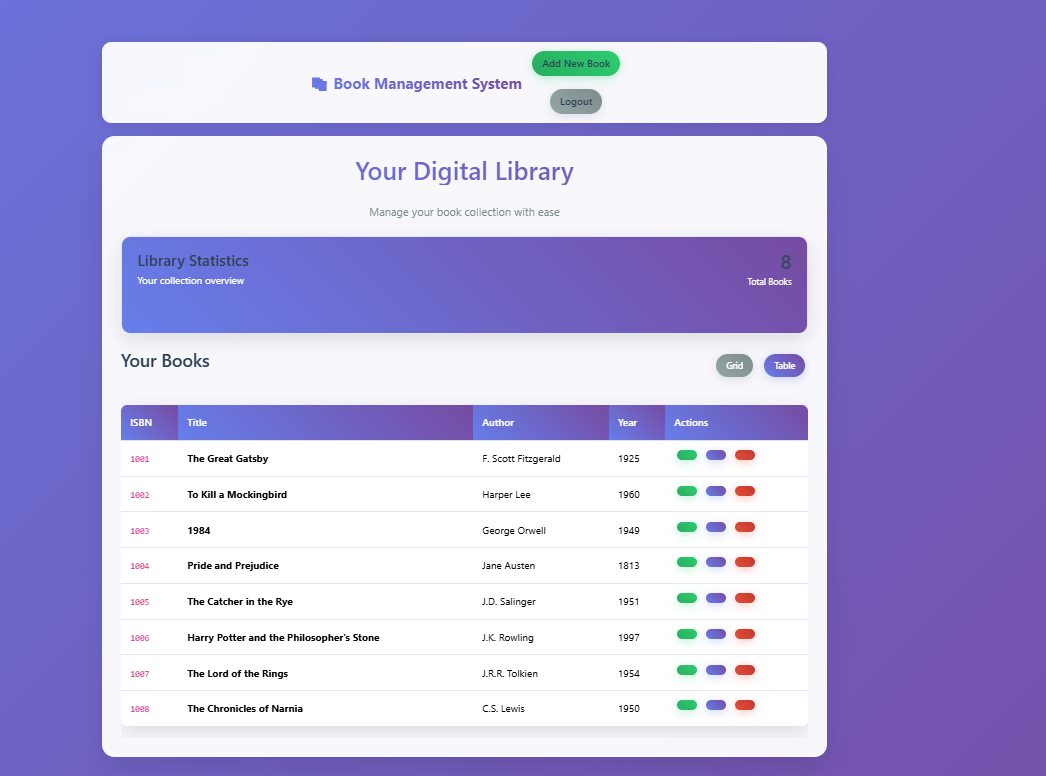
├── angular.json # Angular Configuration

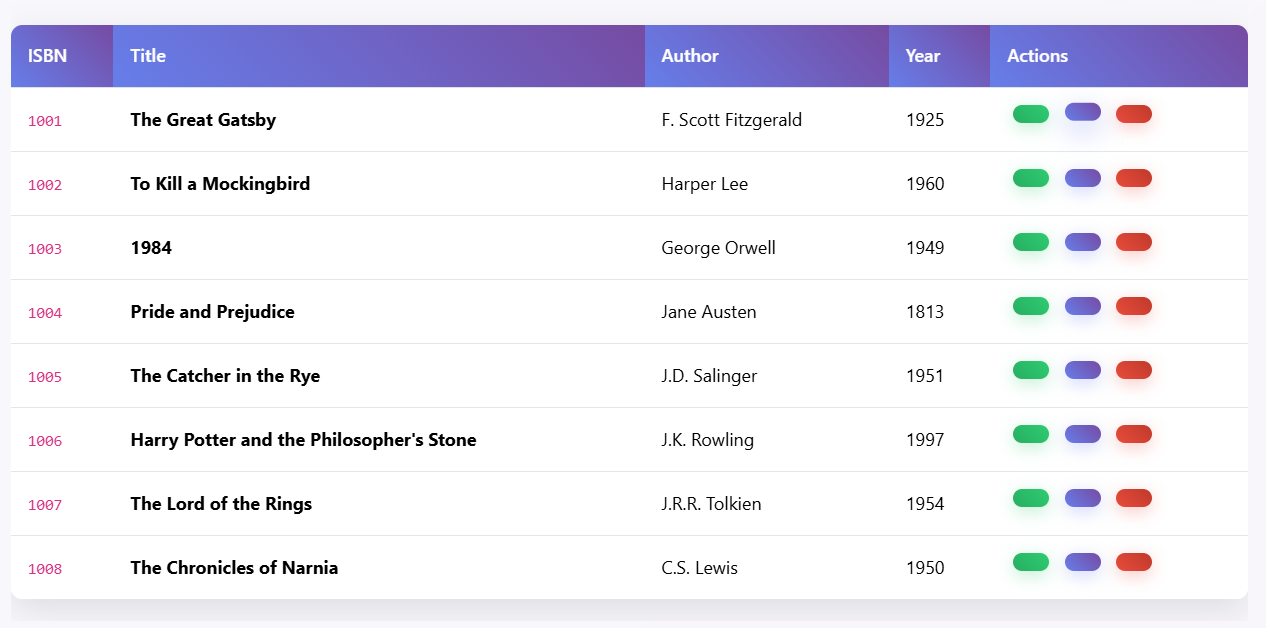
├── package.json # NPM Dependencies

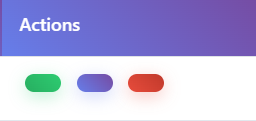
└── tsconfig.json # TypeScript Configuration







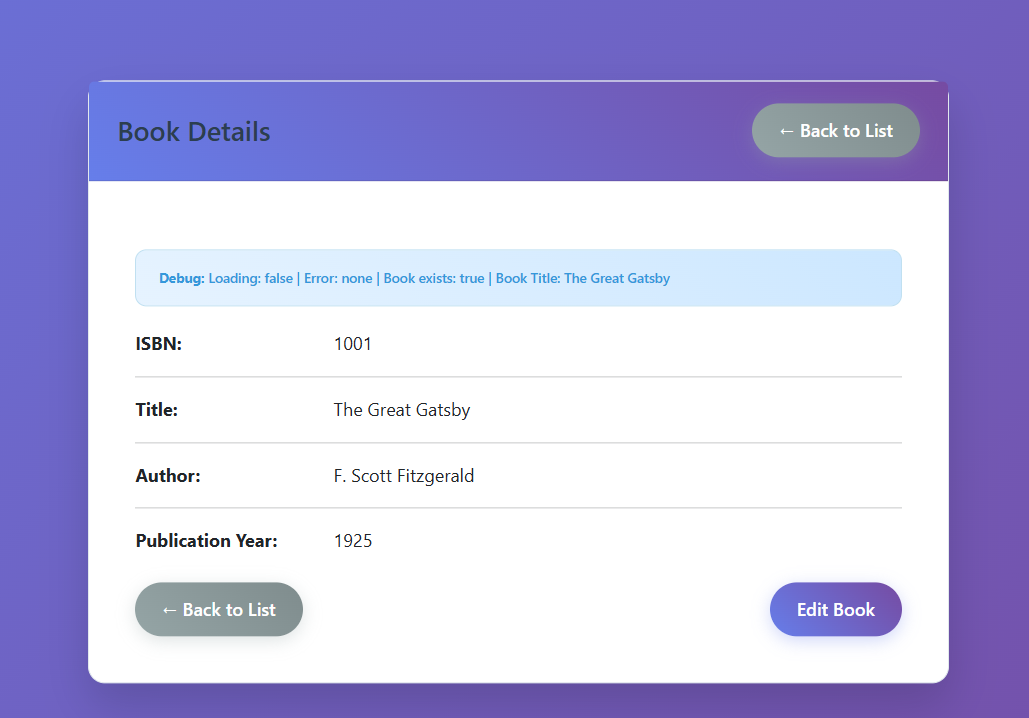


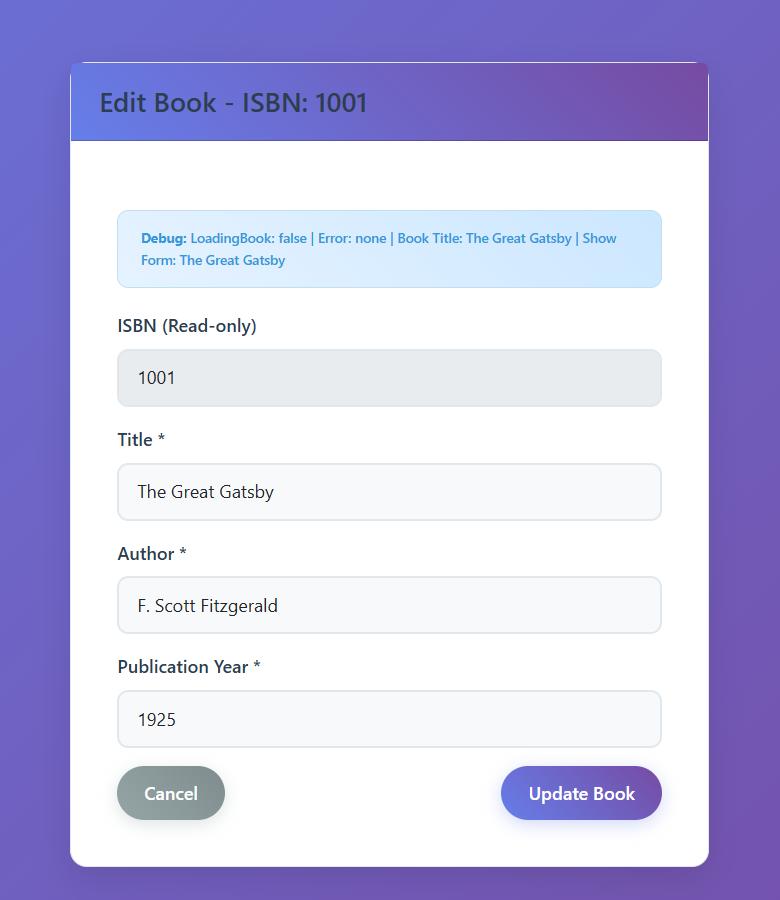


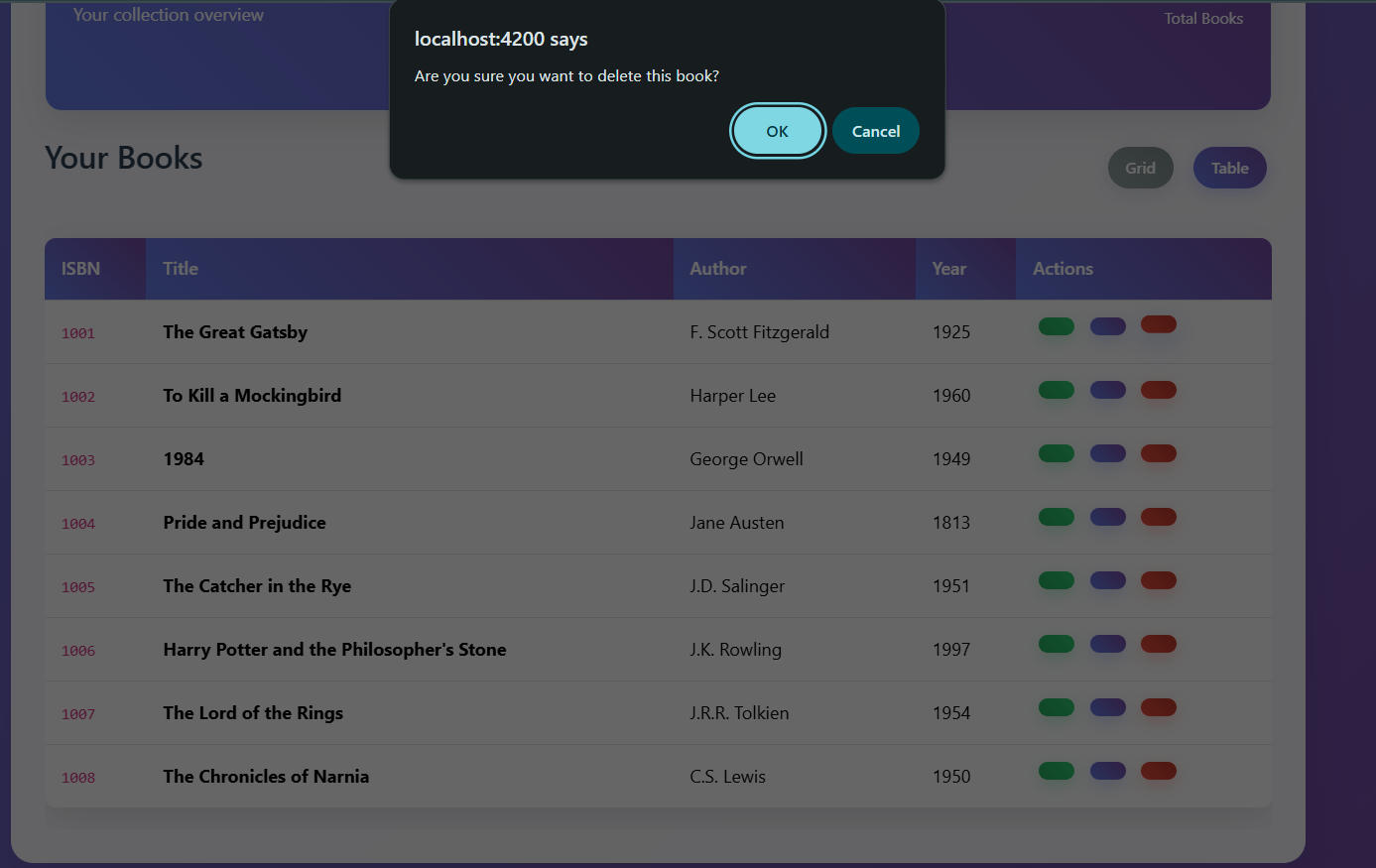
View Edit Delete

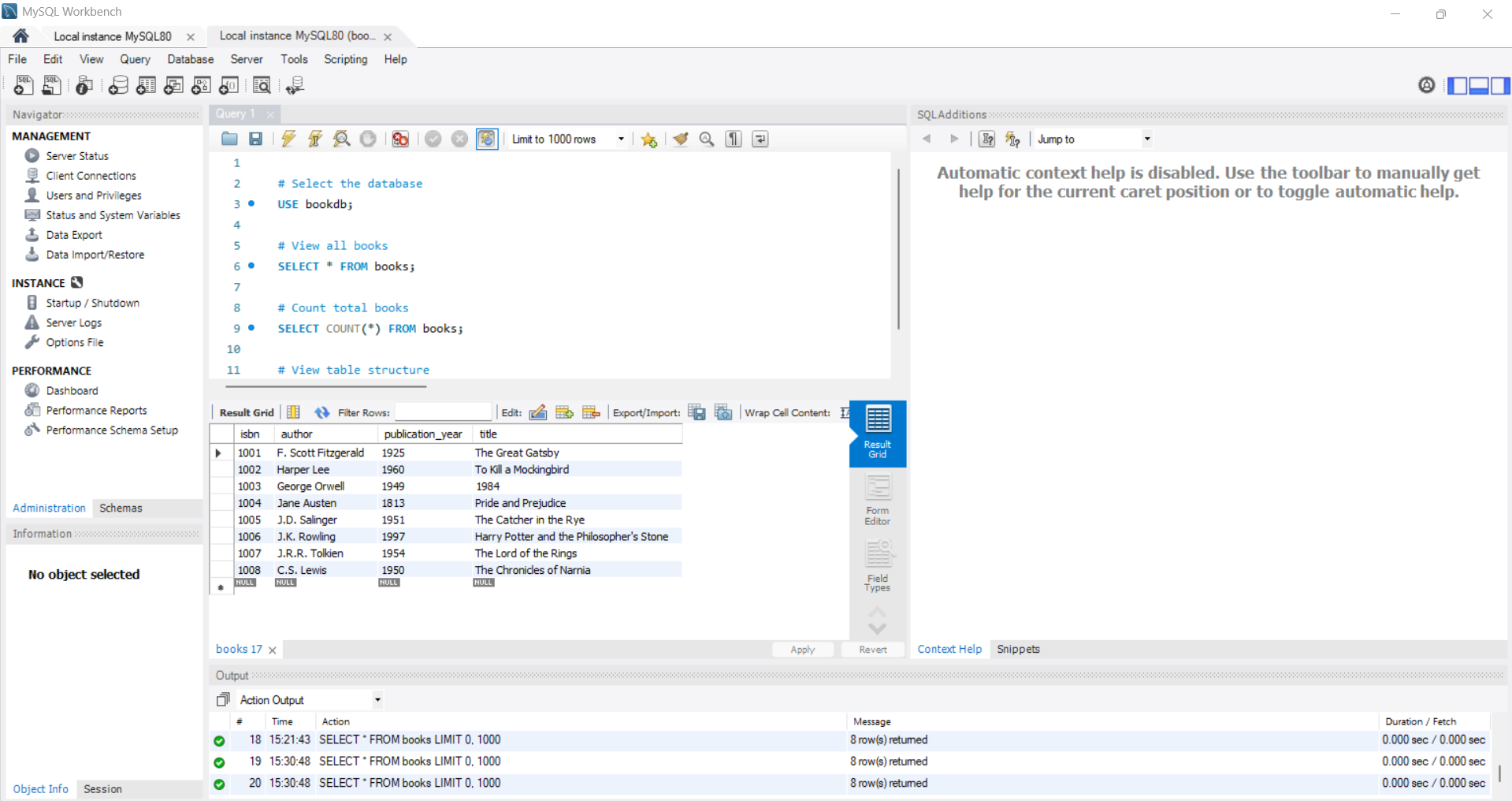
Details Book book











**Access URLs:**

* **Frontend**: [http://localhost:4200](vscode-file://vscode-app/c:/Users/vsour/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* **Backend API**: [http://localhost:8083](vscode-file://vscode-app/c:/Users/vsour/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* **Swagger UI**: [http://localhost:8083/swagger-ui.html](vscode-file://vscode-app/c:/Users/vsour/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* **H2 Console**: [http://localhost:8083/h2-console](vscode-file://vscode-app/c:/Users/vsour/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) (if H2 enabled)

**KEY FEATURES IMPLEMENTED**

**1. Authentication & Security**

* **JWT-based Authentication**: Secure token-based login system
* **Role-based Access**: Admin user with password protection
* **CORS Configuration**: Cross-origin requests enabled
* **Security Headers**: Protection against common attacks

**2. Book Management (CRUD Operations)**

* **Create**: Add new books with validation
* **Read**: View all books with search/filter capabilities
* **Update**: Edit existing book information
* **Delete**: Remove books from inventory

**3. Modern User Interface**

* **Responsive Design**: Works on desktop, tablet, and mobile
* **Beautiful UI**: Modern gradient design with Bootstrap
* **Grid/Table Views**: Multiple viewing options
* **Interactive Cards**: Hover effects and animations
* **Form Validation**: Real-time input validation

**4. API Documentation**

* **Swagger UI**: Interactive API documentation
* **OpenAPI 3**: Standard API specification
* **Endpoint Testing**: Built-in API testing interface

**SECURITY IMPLEMENTATION**

**Authentication Flow**

1. User logs in with username/password
2. Server validates credentials
3. JWT token generated and returned
4. Token stored in frontend localStorage
5. Token sent with each API request
6. Server validates token for each request

**Security Features**

* **Password Encryption**: BCrypt hashing
* **Token Expiration**: Configurable JWT expiry
* **CORS Protection**: Controlled cross-origin access
* **SQL Injection Prevention**: JPA/Hibernate protection
* **XSS Protection**: Input sanitization

**🌐 API ENDPOINTS**

**Authentication**

* POST /auth/login - User login
* POST /auth/refresh - Token refresh

**Books Management**

* GET /books - Get all books
* GET /books/{id} - Get book by ID
* POST /books - Add new book
* PUT /books/{id} - Update book
* DELETE /books/{id} - Delete book