**📚 Book Management System**

A full-stack Book Management application built with **ASP.NET Core** and **React**, implementing modern best practices such as JWT authentication, SOLID principles, repository pattern, and fluent validation.

**🔧 Technologies Used**

**Backend**

* **ASP.NET Core Web API**
* **Entity Framework Core (Code First + Migrations)**
* **JWT Authentication**
* **SQL Server** with indexed tables
* **FluentValidation** (in progress)
* **Custom Middleware for Exception Handling**
* **Logging**
* **SOLID Principles**
* **Generic Repository Pattern**
* **Custom API Response Wrappers**
* **Dependency Injection**
* **CancellationToken Support**

**Frontend**

* **React** (with TypeScript)
* **Material UI (MUI)**
* **Axios** for HTTP requests
* **JWT Authentication**
* **Responsive UI**
* **Book Listing, Search, Pagination, Add, Edit, Delete**

**✅ Features Implemented**

**🛠 Backend Functionality**

* **Database Design**
  + Authors, Books, Users tables
  + Relationships and indexing for performance
* **Authentication & Authorization**
  + JWT-based authentication
  + Login system
  + Role-based authorization (expandable)
  + Secure password hashing
* **Controllers**
  + AuthenticationController – login endpoint with JWT response
  + BooksController – full CRUD + search + pagination
  + UsersController – user management endpoints (if exposed)
* **Services**
  + BookService – handles business logic for books
  + AuthorService – manages authors
  + AuthenticationService – login logic and token generation
  + UserService – user creation and management
* **Validation & Error Handling**
  + Custom ExceptionMiddleware for clean API errors
  + Centralized list of error codes (via ErrorCodes enum)
  + FluentValidation (incomplete)
* **Best Practices**
  + SOLID architecture
  + Repositories: GenericRepository, BookRepository, UserRepository
  + Separation of concerns
  + Pagination and CancellationToken support
  + Logging at critical points

**💻 Frontend Functionality**

* **Book List Page**
  + View all books with pagination
  + Search by title, author, or ISBN
  + Edit and Delete functionality
  + Responsive layout
* **Book Form**
  + Create or update a book
  + Auto-populated fields when editing
  + Cancel and Save buttons
* **Authentication**
  + Login screen with JWT token storage
  + Logout support
* **Usability**
  + Pagination with state management
  + Search box with debounce (soon to improve UX)
  + Table UI with Material UI components
  + Loading spinners and error messaging

**🧪 What’s Missing / To Do**

* ✅ **Unit Tests**
  + AuthenticationService
  + BookService
  + Controller tests
* 🔁 **Integration Tests**
* 🎯 **Code Cleanup & Refactoring**
  + Consolidate duplicate logic
  + Better naming and comments
* 🔐 **Encryption**
  + Use for sensitive fields (optional enhancement)
* 📏 **Complete FluentValidation Rules**
* 💡 **Improve Search UX**
  + Debounce API calls to avoid flickering on each keystroke
* 🧑‍🎨 **UI Enhancements**
  + Toasts/snackbars for feedback
  + Better error visuals

**🔐 Authentication Flow**

1. User logs in using email and password.
2. Backend validates credentials and returns a JWT token.
3. Token is stored in local storage on the frontend.
4. Token is sent in Authorization headers for protected endpoints.

**🐛 Known Bug – Search Input Refresh /🐞 Bug – Search Field Loses Focus**

* **On the Book List page, as you type in the search box, the entire page refreshes to fetch new data.**
* **This causes the search box to lose focus, making it feel like it's playing hide and seek with your cursor. 🎯⌨️**
* **What's happening? The search triggers a backend call that refreshes the component without preserving input focus.**
* **Suggested Fix: Debounce the input, manage search state properly, and avoid full component refresh on each keystroke.**

**🗂️ Sample Data**

* **An extra SQL script for inserting sample data is available in the project’s root directory. Just run it to populate the database for testing.**

.

**🚀 Getting Started**

1. **Backend**

**cd Backend/**

**dotnet ef database update**

**dotnet run**

1. **Frontend**

**cd book-list-app/**

**npm install**

**npm start**