Library Management System

Software Engineering Internship – Assignment

Rageeshan Chandrasegaran

Introduction

The **Library Management System** is a full-stack web application developed to manage a collection of books efficiently. The system allows users to:

- View all books
- View detailed information about a specific book
- Add new books
- Edit existing books (via modal interface)
- Delete books

The project demonstrates modern full-stack development using:

- **React + TypeScript** for the frontend
- **ASP.NET Core Web API (C#)** for the backend
- RESTful communication between client and server

The main objective was to design a clean, responsive, and scalable system while following good software architecture practices.

System Architecture

The application follows a **client-server architecture**:

Architecture Overview:

Frontend:

- Built using **React** with **TypeScript**
- Routing handled by **React Router**
- API calls handled using **Axios**
- Styled using **Tailwind CSS**
- State managed using React Hooks (useState, useEffect)

Backend:

- Built using **ASP.NET Core Web API**
- Written in **C#**
- RESTful endpoints implemented using Controllers
- Entity model represents **Book**
- Uses Entity Framework Core
- Handles HTTP requests and responses

Backend Implementation

Project Structure

- Controllers → Handle HTTP requests
- Models → Define Book entity
- Services/Repository (if implemented) → Business logic
- DbContext → Database connection

Implemented API Endpoints

Method	Endpoint	Description
GET	/api/books	Get all books
GET	/api/book/{id}	Get book by ID
POST	/api/book	Create new book
PUT	/api/book/{id}	Update existing book
DELETE	/api/book/{id}	Delete book

Backend Responsibilities

- Validates incoming data
- Returns appropriate HTTP status codes
- Handles errors (404 Not Found, 400 Bad Request, etc.)
- Ensures proper database interaction

Frontend Implementation (React + TypeScript)

Project Structure

- pages/ → DisplayPage, DetailsPage, AddPage
- services/ → API communication (Axios instance)
- types/ → TypeScript interfaces (Book)

Key Frontend Features

- Display books in table format
- View book details page
- Edit book via modal popup
- Delete book with UI update
- Loading spinner during API calls
- Error handling and user feedback
- Responsive design using Tailwind CSS

Challenges Faced

Frontend–Backend Integration

Ensuring correct communication between React and ASP.NET Core required careful handling of API routes, HTTP methods, and JSON formatting. Small mismatches in endpoint naming or route configuration initially caused request failures.

Dependency Mismatch Issues

One of the significant challenges faced during development was dependency mismatches between different packages in the frontend and backend environments.

Dependency challenges included:

- .NET SDK version compatibility
- NuGet package version mismatches
- Entity Framework Core version alignment with .NET version

These were resolved by:

- Ensuring consistent .NET SDK version
- Updating NuGet packages
- Cleaning and rebuilding the solution

Modal-Based Editing Implementation

Implementing editing functionality inside a modal required:

- Pre-filling form data
- Managing local component state
- Handling asynchronous PUT requests
- Updating UI without page refresh

Additional Features Implemented

- Modal-based inline editing
- Loading spinners
- Error message components
- Immediate UI updates after deletion
- Clean and responsive UI design
- RESTful structured backend

Key Insights Gained

- Importance of separating frontend and backend concerns
- Benefits of RESTful API design
- Strong typing in TypeScript reduces runtime errors
- ASP.NET Core provides clean structure for scalable APIs
- Proper error handling improves user experience
- State management is critical in dynamic web applications

Conclusion

The Book Management System successfully demonstrates full-stack development using modern technologies. The integration between React (frontend) and ASP.NET Core (backend) highlights practical knowledge of:

- REST API development
- Frontend state management
- Full CRUD operations
- Clean UI/UX implementation

The project strengthened understanding of client-server architecture and professional development workflows.