Library Management System

Development and Implementation Report

Sameera Madusanka

sameeramjayawickrama@gmail.com

0755433842

Contents

01. Introduction

The purpose of this project was to develop a Library Management System and enable users to perform basic library operations such as adding, viewing, updating, and deleting books. The system incorporates user authentication access using JWT (JSON Web Tokens).

1.1 Scope

The project includes the following functionalities:

- User authentication using JWT for secured API access.
- CRUD (Create, Read, Update, Delete) operations for book management.

1.2 Technologies Used

- **Backend**: C# .NET 8, Entity Framework, Identity EntityFrameworkCore
- **Frontend**: React, TypeScript, Axios
- **Database**: SQLite
- **Styling**: Bootstrap, React-Bootstrap, Sweet Alert

02. Development Process

2.1 Backend Implementation (C# .NET 8)

Architecture

- The backend follows a Layered services Architecture, including separate project libraries with Controllers, Services, and Data Transfer Objects (DTOs). The database schema includes tables for Users and Books.

Key Features

Authentication

- JWT-based authentication was implemented using ASP.NET Core Identity. Users receive a token upon login, which is used to secure API requests.

API Endpoints

Endpoints include for the Book CRUD operations:

- GET /api/Book: Retrieves all books.
- POST /api/Book: Adds a new book.
- GET/api/Book/{id}: Retrieve book based on Id
- PUT/api/Book/{id}: Update existing book
- DELETE/api/Book/{id}: Remove existing book from the data base

Database

SQLite is used as the database and the database schema includes:

- User Registration and login: Stores user information like email and Password.
- Books: Contains book details such as title, Description and author.

2.2 Frontend Implementation

Structure

- The React front end is organized with separate folders for components, services, Interface and routes and also uses config.ts file to keep the Base URL in globally .

Key Features

Authentication Component

- Authentication Component contain the Login and Register tsx files and used React-Bootstrap for the UI

Books Component

- It contains the AddBook.tsx, UpdateBook.tsx and ViewBooks.tsx file and used React-Bootstrap for the UI.

CRUD Functionality

- Users can add, update, or delete books via forms and button actions.

03. Challenges Faced

3.1 CORS Issues

- While integrating the backend and frontend, I faced cross-origin request (CORS) issues. This was resolved by configuring the backend to allow requests from the frontend's origin, enabling seamless communication between the two.

3.2 Learning TypeScript

- This was my first time using TypeScript for a project. Before starting the development, I had to go through several TypeScript tutorial videos and documentation to understand its concepts, such as typing, interfaces, and decorators. While it was challenging initially, it greatly improved the quality and maintainability of my front-end code.

3.3 Frontend Challenges

- Although I have experience with backend development, working on the frontend was a new experience for me.

04. Additional Features

4.1 SweetAlert for Enhanced User Feedback

- Implemented SweetAlert for displaying user-friendly alerts and confirmations.
- Used it to show success messages (e.g., "Book added successfully") and error notifications (e.g., "Something went wrong").

4.2 React-Bootstrap for Streamlined UI Development

- Leveraged React-Bootstrap to create a clean and responsive user interface.
- Used components like forms, buttons, and cards to enhance the overall user experience.

4.3 Bootstrap for Styling

- Used Bootstrap's grid system and utility classes to ensure a consistent layout and responsive design.

4.4 User Authentication

- Implemented user authentication on the backend using JSON Web Tokens (JWT).
- Although not required in the assignment, this feature was added to improve security and demonstrate practical implementation of authentication workflows.

05. Key Insights

5.1 Technical Skills

Enhanced Typescript Knowledge:

- Gained a solid understanding of TypeScript, including its type system, interfaces, and error handling. As mentioned earlier this was my first experience React with TypeScript.

Improved React and Axios Skills:

- Gained proficiency in using Axios for API integration, including handling requests, responses, and errors.

Strengthened Understanding of RESTful APIs:

- Developed better insights into designing and consuming RESTful APIs efficiently in a full-stack environment.

5.2 Collaboration and Planning

Importance of Modular Design and Documentation:

- Realized the value of breaking the application into smaller, reusable components and maintaining clear documentation for easier project navigation and future updates.

06. User Accounts And credentials & others

The system includes a user account management module for secure access. Below are the details of a pre-registered user for testing and demonstration purposes:

- **Email**: sameera@gmail.com
- **Password**: Sameera@123

If the above credentials are not applicable, users can create their own accounts via the **Register** functionality provided in the application.

07. Libraries, Software and Framework Versions

The following versions were used during the development of this system:

- **Node.js**: 20.10.0
- **Axios**: 1.7.7
- **React**: 18.3.1
- **React Router DOM**: 6.28.0
- **SweetAlert2**: 11.14.5
- **.NET**: 8

These versions ensure compatibility and optimal performance of the application.

08. Conclusion

The Library Management System successfully achieved its objectives, including secure user authentication and CRUD functionality.