Expertenic Assessment

Software Engineer Internship Candidate

Chamuditha Heshan

Report

Date : 2024-11-08

Library Management System

Contents

Introduction

This Book Management System is a practical application that allows to maintain records of books, including their titles, authors, and descriptions. Following processes highlight the key functionalities of this project.

1. Creating Book Recode – Users can add new records by entering key details of the book such as title, author, and description. Once submitted new book is stored.

2. View Books – To maintain the transparency and accessibility, the system offers user to view the existing books. This overview facilitates easy browsing and helps users stay informed about the available collection

3. Update Books – The application provides way for users to edit and update existing book records. This functionality ensures that the database remains accurate and up-to-date with minimal effort.

4. Delete Books – For instances where a book is no longer relevant or required, the system includes a feature to delete existing books. his helps keep the list organized and relevant, removing outdated or unnecessary entries to streamline the user's experience.

These are the core features of this system. To access these features, first of all user need to register for the system and log in to the system.

Project Overview

The Library Management System enables users to manage book records, with each record containing essential information such as the book's title, author, and description. The system is divided into two parts:

- Backend (C# .NET): A RESTful API that handles CRUD operations and interacts with the SQLite database by Entity Framework.
- Frontend (React and TypeScript): A responsive web interface that communicates with the backend to perform the CRUD operations.

System Requirement Specification

- Functional Requirements

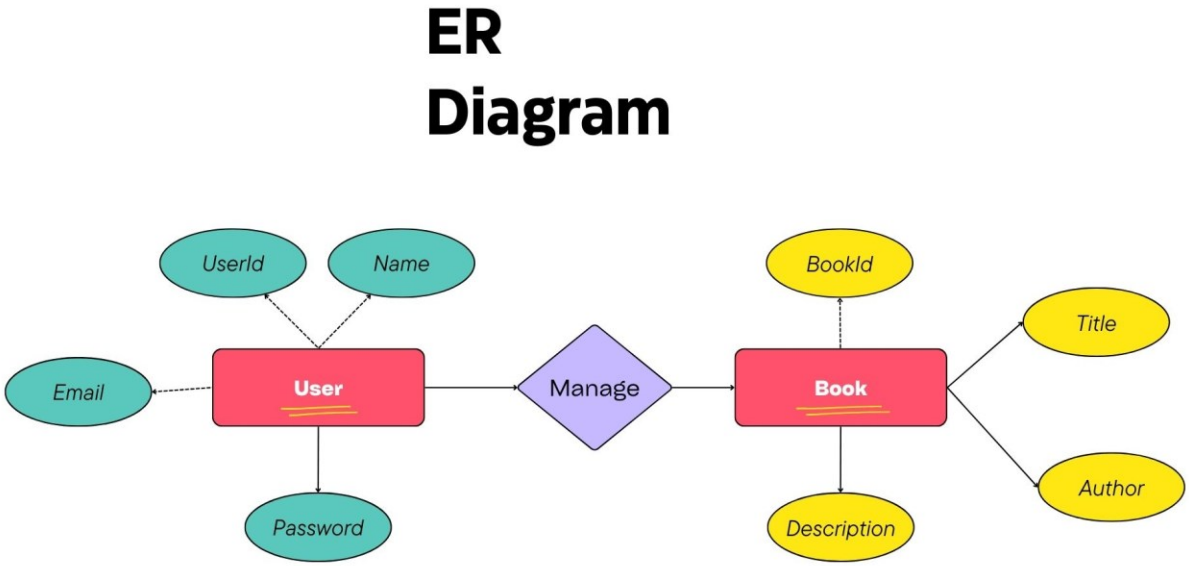
  For user,

  - User shall be able to add books
  - User shall be able to view books
  - User shall be able to update books
  - User shall be able to Delete books

- Relational Schema

  - User(<u>UserId</u>, Email, Password, name)
  - Book(<u>BookId</u>, Title, Author, Description, UserId)

ER Diagram



User Interfaces

- Landing Page (This page loads when the user loads the system)
- Login/SignUp Page (User login SignUp functionalities UI)
- Dashboard Page (Book management UI)
- Profile Page (User detail displaying UI)

Technologies used

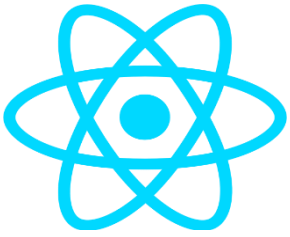
Frontend – React with Typescript

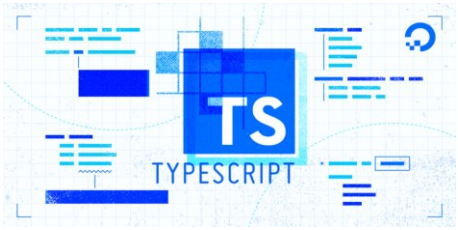
Bakckend - C#.net

Database – SQLite

IDEs – Visual Studio Code for frontend development

Visual Studio for backend development





Summary of Development Phase

The development phase of the Library Management System (LMS) was focused on creating both the backend and front end of the application, ensuring proper integration between them, and implementing key functionalities for managing book records.

- **Backend Development (C# .NET)**:
  - The backend API was developed using C# and .NET Core, with an SQLite database to store book records.
  - The application exposed four key RESTful API endpoints for **Create**, **Read**, **Update**, and **Delete** operations (CRUD) on book records.
  - **Entity Framework** was used for database interaction, providing seamless mapping between C# models and the SQLite database.
  - Error handling and input validation were incorporated to ensure that API requests were properly processed and returned meaningful responses.
- **Frontend Development (React and TypeScript)**:
  - The frontend was developed using React and TypeScript, creating a responsive and user-friendly interface for managing books.
  - CRUD functionalities were implemented on the frontend, allowing users to add new books, view existing records, update details, and delete books from the system.
  - React's component-based architecture was used to structure the interface, and **Fetch** was employed for making HTTP requests to the backend API.
  - The UI was styled to be simple yet intuitive, ensuring a smooth experience for users.
- **Integration**:
  - The backend and frontend were integrated to work seamlessly, allowing users to interact with the system through the frontend interface and have their actions reflected in the backend.
  - Proper communication between the frontend and backend was ensured by using HTTP requests, and the data flow was managed using React's state management system.

- **Challenges**:
    - The development process faced challenges, including resolving **CORS issues** when connecting the frontend and backend and ensuring proper **database migrations** with SQLite.
    - Input validation and error handling were crucial aspects to ensure the integrity of the data entered by the users.
    - Integrating Authentication : While not part of the core requirements, considering the implementation of user authentication added another layer of complexity. Deciding between implementing a basic JWT-based token authentication posed architectural challenges and required careful planning of user sessions, token expiration, and security considerations.

- **Additional Features**:
    - **User Authentication**
        - Implemented user Authentication also using traditional method. Integrated with Jwt tokens. Once a user logged by entering email and password. System checks for the email and password(decrypting hashed password) . If email and password correct, a new token will create with email, relevant UserId and token expiration time. After 60 minutes, the site will automatically be redirected to login after refreshes.

Overall, the development phase was a success, with both the backend and frontend being implemented as per the requirements. The project is functional, with key features such as book creation, updating, and deletion available with user authentication, making it a solid foundation for any future enhancements.

Git Repository

https://github.com/chamuditha01/Library-Management-System.git

Conclusion

The development of the Library Management System provided me with valuable experience in building a full-stack web application using C# .NET for the backend, React with TypeScript for the frontend, and SQLite as the database. Throughout the development process, I successfully implemented the core features of the system, including the ability to **Create**, **Read**, **Update**, and **Delete** book records, providing users with a seamless and intuitive interface for managing their library.

A significant challenge I faced was integrating the backend API with the frontend, ensuring smooth communication between the two components. This was made complex by CORS issues, asynchronous API calls, and managing loading states on the front end. I also overcame difficulties with database migrations, input validation, proper user authentication and proper error handling to ensure the stability and security of the application.

An additional feature I implemented was **user authentication**. Although not initially part of the core requirements, integrating user authentication using JWT tokens added an extra layer of security and personalized user experience. This feature allowed users to register, log in, and access book records only after authentication. The implementation of authentication involved handling, token management, and authorization to ensure secure access control.

Throughout the process, I paid careful attention to **responsive design**, ensuring the application provided a smooth experience across various devices. The UI was built to be user-friendly and aesthetically pleasing, with input validation and error handling incorporated to improve usability.

In conclusion, this project demonstrated my ability to integrate backend and frontend technologies, implement key features like CRUD operations and user authentication, and maintain a strong focus on code quality and security. The challenges I faced during development contributed to a deeper understanding of building full-stack applications and handling complex scenarios such as asynchronous operations, state management, and database integration. The resulting Library Management System is a robust and functional application, offering a solid foundation for managing books in a library.