Software Engineering Internship – Assignment

Library Management System

Overview

Candidate: Kushana Senadheera

Contents

Library Management System

1. Introduction

The Library Management System is an intuitive system, which is intended for the efficient resource management of libraries and library systems. This system enables holders of a library to address books more efficiently by providing the most basic operations that will enable the user to create a record, retrieve, modify and even delete records of books.

The application has a strongly typed but lightweight back-end which utilize C# .NET, together with an SQLite data base engine making the application light but efficient. Frontend built on React and TypeScript doesn't only ensure the simplicity and interactivity of the application interface but also provides users with a convenient list of actions.

This system has been designed to automate library systems in a way that will minimize manual work while boosting productivity, making available a new system that will suit the existing library.

1.1 GitHub Repositories

1. Frontend
2. Backend

2. Development Process

The development process started with planning and requirement analysis phase. The requirements for the project as well as the system architecture was determined by reviewing the resources provided by Team Expernatic. To capture all functionalities in the project, its scope was properly defined to capture all the pertinent functions.

Technology stack for the project was chosen as React with TypeScript for the creation of an interactive frontend of the application. SQLite was used for the data storage and retrieval as it is fast and reliable and the back end was coded in C# and .NET for secure and efficient execution of programs.

3. Backend Implementation

Architecture:

The backend has a structure of the MVC (Model-View-Controller) model in order to provide modularity and separation of concerns. BooksController is the main controller responsible for requests connected with books, models describe the structure of the data, and the database context is responsible for carrying out operations on data.

Database Design:

SQLite is used as the database, there is a table to store book information such as Title, Author, Category, Entry Date, and Description. Relations and optimizations make data management effective. It is used to get the database using the ApplicationDbContext class for better compatibility with Entity Framework Core.

APIs:

The **BooksController** provides RESTful API endpoints for all CRUD operations:

- **GET /api/Books:** It is used to obtain all the books in the database.
- **POST /api/Books:** Inserts a new book record into the database using the Data Transfer Object (DTO).
- **GET /api/Books/{id}:** Used to get a particular book by its id.
- **PUT /api/Books/{id}:** A method that modifies the information of a book in accordance with the given id and DTO.
- **DELETE/api/Books/{id}**: Deletes a book by its unique id.

Every API endpoint has a basic concept of HTTP status codes including 200 OK, 404 Not found, and 400 Bad Request.

4. Challenges Faced

Backend Environment Setup with Visual Studio IDE: One of the biggest issues which were encountered was to configure the backend environment in Visual Studio IDE. Some of the problems observed in the initial setup process included setting problems such as wrong configurations, missing dependencies and project setup problems.

Verified Required .NET Core Version: Made sure that all the correct versions of .NET Core were installed to meet the project specifications.

Resolved NuGet Package Errors: Fixed problems that resulted to the absence of NuGet packages through Package Restore in Visual Studio Package Manager.

Addressed Build and Runtime Issues: Searched through error reports and fixed any mistakes in the backend code that was causing problems with the build as well as during the runtime.

5. Prerequisites

Frontend:

- **Visual Studio Code**: Code editor for writing frontend code (React with TypeScript).
- **Node.js**: JavaScript runtime needed to run the React application.
- **npm**: Package managers for managing frontend dependencies.

Backend:

- **Visual Studio**: IDE for developing the backend with .NET.
- **.NET Core SDK**: Required for building and running the .NET application.
- **SQLite**: Lightweight database used for storing book data.

5.1 Frontend Deployment Instructions

1. Clone the repository:

   o git clone https://github.com/KushaRa/LMS_Assignment.git

2. Navigate to the project directory:
   o cd LMS_Assignment/lms_frontend
3. Install dependencies:
   o npm install

5. Start the development server:

   o npm run dev

   o The application will be available at http://localhost:5173

5.2 Backend Deployment Instructions

1. Clone the repository:

   o git clone https://github.com/KushaRa/LMS_Backend_Assignment.git

2. Navigate to the project directory:

   o cd LMS_Backend_Assignment

3. Restore dependencies:
   o dotnet restore

4. Start the server:
   o dotnet run
   o Swagger documentation: https://localhost:7270/swagger

Git was adopted as the version control system for the frontend and backend development of the project. They let for proper control over the project's codebase. Through a clean commit history, Git made it possible to trace all the changes made and review them or revert them if needed.

6. Future Implementation

While the current implementation focuses on core CRUD functionality, future enhancements could include:

- **Authentication**: Implementing user authentication to ensure secure access to the application, allowing only authorized users to perform certain actions.
- **Role-based Access Control**: Introducing role-based access control (RBAC) to restrict actions based on user roles, ensuring that users can only perform actions permitted by their roles (e.g., admin, user).
- **Search and Filter Functionality**: Implementing advanced search and filter options for better management of books in the system.
- **User Interface Enhancements**: Improving the UI/UX for a more intuitive and responsive experience across different devices.

7. Conclusion

LMS is used to strengthen the library management to streamline the resource management process. It has a simple backend coded with C# .NET and SQLite database to ensure the functionality of the program and the frontend is created with React and TypeScript for the users. It has basic CRUD functionality for handling the records of books and it contains additional operations like Insert, Update, Find and Delete. The problems encountered during development were establishing the backend environment in Visual Studio IDE, missing dependencies, and integration. Some of the possible future improvements are the authentication of the input, validation of the input that is provided, the access of a specific role and the search/filter functionalities. The use of the system is to automate the library operations and improve the quality of the service delivery and reduce human intervention.