LIBRARY MANAGEMENT SYSTEM CRUD APPLICATION REPORT

By:

Nadun Perera

Introduction

This report outlines the development process of a Library Management System built as a CRUD (Create, Read, Update, Delete) application. The goal of this system is to manage books in a library by providing functionality to add new books, retrieve all books, update details of existing books, and delete books. Frontend interacts with the backend via API calls, using RESTful methods for the CRUD operations. In addition, React Context was used for managing the global state and API calls, and a carousel feature was implemented on the homepage.

Project Overview

Features Implemented

- Add New Book: Users can add a new book to the library by providing details such as title, author, and ISBN.
- Get All Books: A page displays all books in the library with options to delete or update them.
- Update Book Details: Users can update the details of an existing book.
- Delete Book: Users can remove a book from the library by deleting it from the database.
- Global API Context: Using React Context, all API calls are managed in a centralized location, and the data is shared across various components.
- Homepage Carousel: A carousel on the homepage displaying featured books or content.

Technologies Used

- Frontend: React with TypeScript
- Backend: C# and .NET
- Database: SQLite database with Entity Framework
- Styling: CSS for styling the components

Backend Implementation

The backend of the application is built using .NET (C#) and SQLite, providing the API that interacts with the database. The API is developed following RESTful principles for handling CRUD operations.

API Routes

- GET /api/books : Fetches all books in the database.
- POST /api/books : Adds a new book to the database.
- PUT /api/books/:id : Updates the details of an existing book.
- DELETE /api/books/:id : Deletes a specific book from the database.
- GET /api/books/:id : Fetches one book in the database by id.

Challenges Faced

A key challenge was implementing the architecture. I used Swagger for proper documentation to help others understand it.

Frontend Implementation

The frontend utilizing functional components and React hooks to manage state and handle interactions with the backend. React Context was used to manage the global state and API calls.

Components

- Navbar: A simple navigation bar that allows users to navigate between pages.
- BookList: Displays all books in the library and provides options to update or delete each book.
- AddBook: A form that allows users to add a new book to the library.
- UpdateBook: A form to update the details of an existing book.
- Carousel: The homepage includes a carousel displaying featured books and enhancing the user experience.
- AuthContext: This context is used to manage API calls related to books, including fetching the list of books, adding new books, updating existing books, and deleting books. The context makes it easy to share this data across components without needing to prop-drill.

Context API for API Calls

The Context API is used to handle all API calls in one place. This includes actions such as fetching the list of books. The context makes the data accessible to all components that need it without having to pass props manually from parent to child components.

By using React Context, the state and API calls are kept in one centralized place, making the application more maintainable and reducing redundancy.

Additional Features

- AuthContext: This context is used to manage API calls related to books, including fetching the list of books, adding new books, updating existing books, and deleting books. The context makes it easy to share this data across components without needing to prop-drill.
- Navbar and Footer
- Swagger

How to Run the Project

Prerequisites

- Node.js: Ensure that Node.js is installed on your system
- .NET SDK: Ensure that the .NET SDK is installed.
- SQLite: SQLite is used for the database, and it should be properly set up in the backend.

Steps

Clone the Repository:

```
git clone < https://github.com/nadunchanna98/Library-Management-System-APS.Net-React-TS >
```

Install Dependencies:

```
cd frontend

npm install
```

Start:

```
npm start
```

GitHub Repository

The complete source code for this project, including both frontend and backend implementations, can be accessed on GitHub. This repository contains all necessary files, instructions, and setup details for running the project locally.

GitHub Repository Link: https://github.com/nadunchanna98/Library-Management-System-APS.Net-React-TS