Library Management System – Project Report

Title: Library Management System – Fullstack Web Application
Author: Sachindra Samadhi
Technologies Used: React, TypeScript, ASP.NET Core Web API, SQLite
Submitted On: 11/07/2025

Table of Contents

1. Introduction

This report describes the development of a fullstack Library Management System as part of a software engineering internship assignment. The goal was to create a simple, functional application allowing CRUD operations for library books with a React frontend and ASP.NET backend, using SQLite for storage.

2. Project Overview

The system allows users to:

- View a list of books
- Create, edit, or delete book records
- Connect a frontend (React + TypeScript) with a backend (C# .NET Web API)
- Persist data using a lightweight relational database (SQLite)

3. Technology Stack

Layer	Technology
Frontend	React, TypeScript, Semantic UI
Backend	ASP.NET Core Web API (C#)
Database	SQLite
Tools	JetBrains Rider, GitHub

4. Backend Implementation

- **Project**: Library-Management
- **Framework**: ASP.NET Core Web API
- **Database**: SQLite via Entity Framework Core
- **Features**:

  - RESTful endpoints: /api/books

  - operations using EF Core

  - Data validation

  - CORS enabled to allow frontend access

- **Routing & Services**:

  - BookController.cs: Handles all HTTP operations

  - LibraryDBContext.cs: DB context for EF Core

  - Auto-created SQLite file library.db included in project

5. Frontend Implementation

- **Project**: frontend (React + TypeScript)
- **Key Libraries**: React Router, Semantic UI, Axios
- **Pages/Components**:

  - BookTable: Displays all books

  - BookForm: Create/edit book

  - BookTableItem: Modal form component

- **API Integration**:

  - API Connector using Axios

  - API_BASE_URL points to Render backend

  - Styling: Semantic UI & MUI

6.Challenges Faced

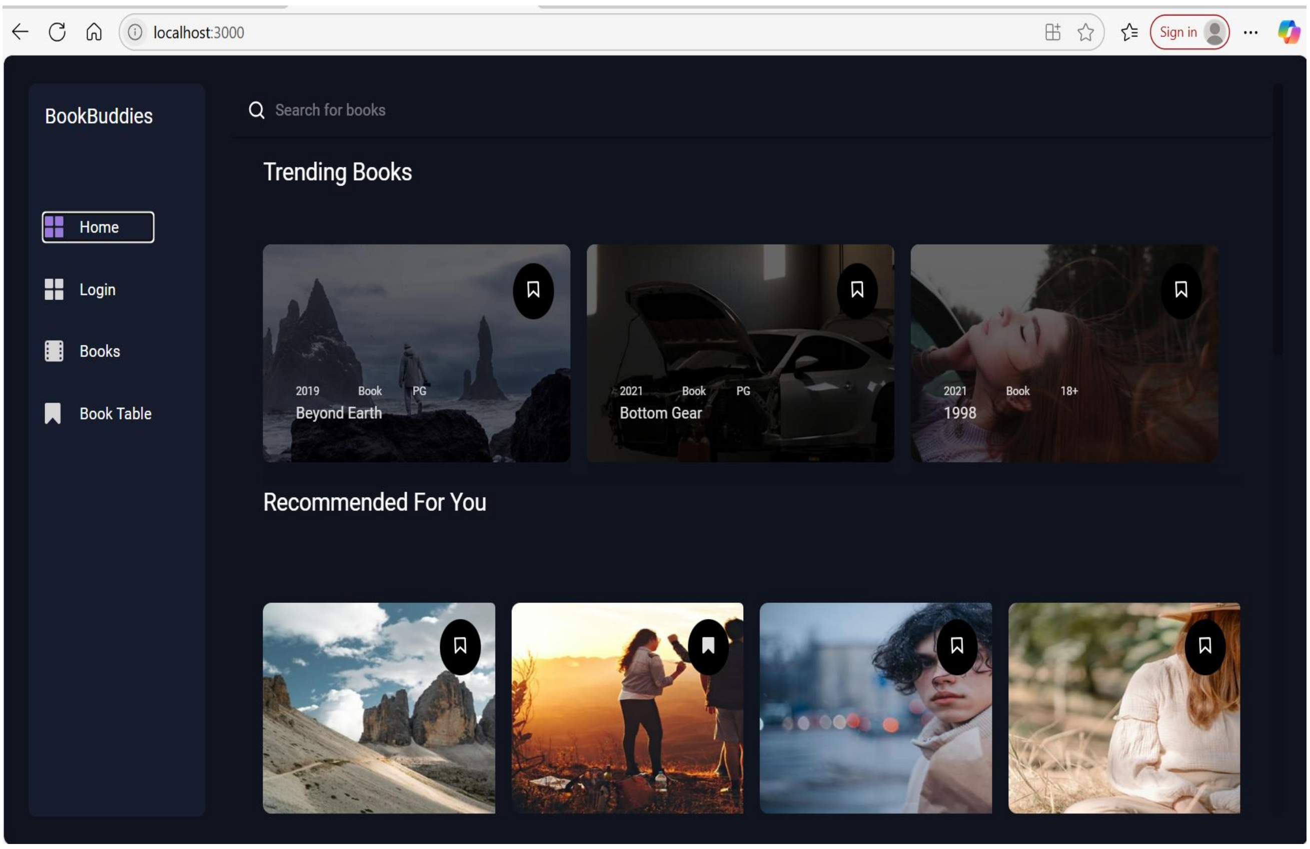
Challenge	Solution
CORS blocking requests	Added proper CORS policy in Program.cs
Maintaining controlled form inputs for both creating and editing books in React was initially difficult, especially when pre-filling data for editing.	Used React’s useState and useEffect hooks effectively to initialize form state based on route parameters. Created a generic handleInputChange function to update form fields dynamically.
Handling asynchronous API calls and ensuring UI updates correctly when data is loaded or modified (e.g., after creating or editing a book).	Utilized Axios for API calls, wrapped requests in async/await, and implemented conditional rendering (Loading...) while waiting for data. Ensured .then() or await was used correctly to wait for API responses before navigating.
React Router’s navigation after creating/editing a book didn’t refresh the book list automatically.	Ensured the navigate('/') call redirected to the list page, and used useEffect in the book list component to fetch data every time the component was mounted.

7. Conclusion

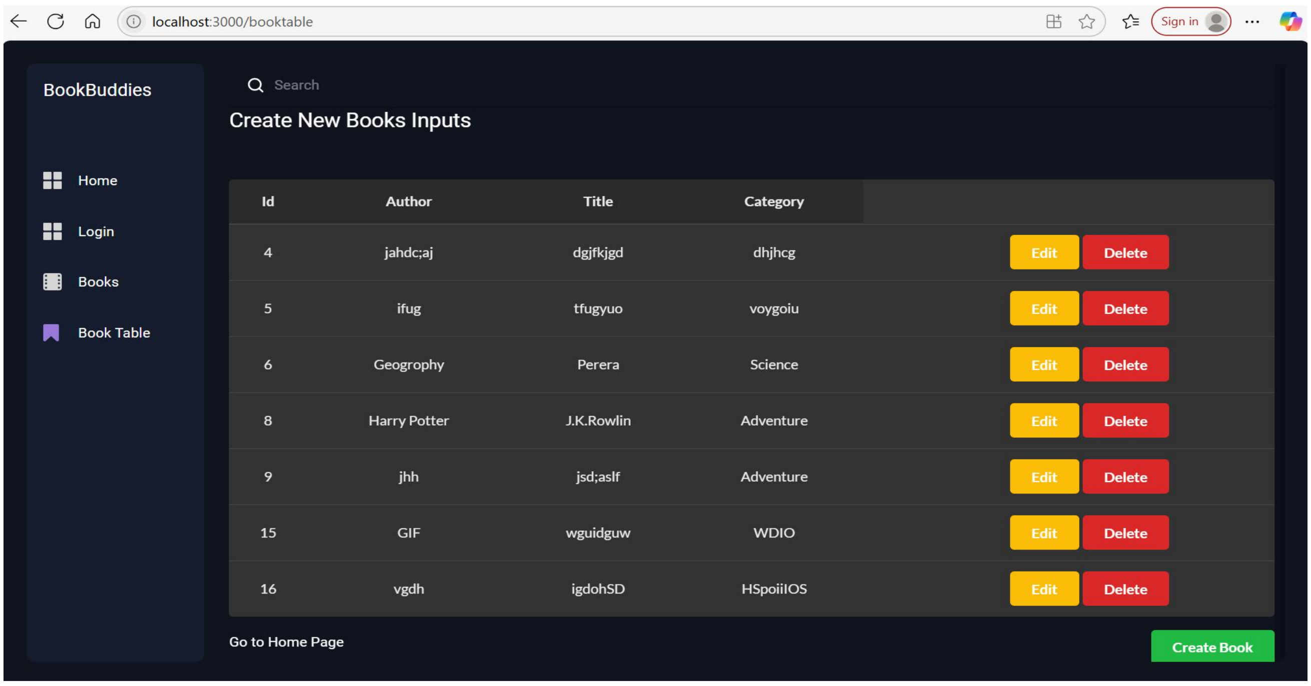
The project successfully achieved its goal of building a working Library Management System using modern web technologies. It strengthened my understanding of fullstack development.

8. Screenshots

Homepage -



Book list -



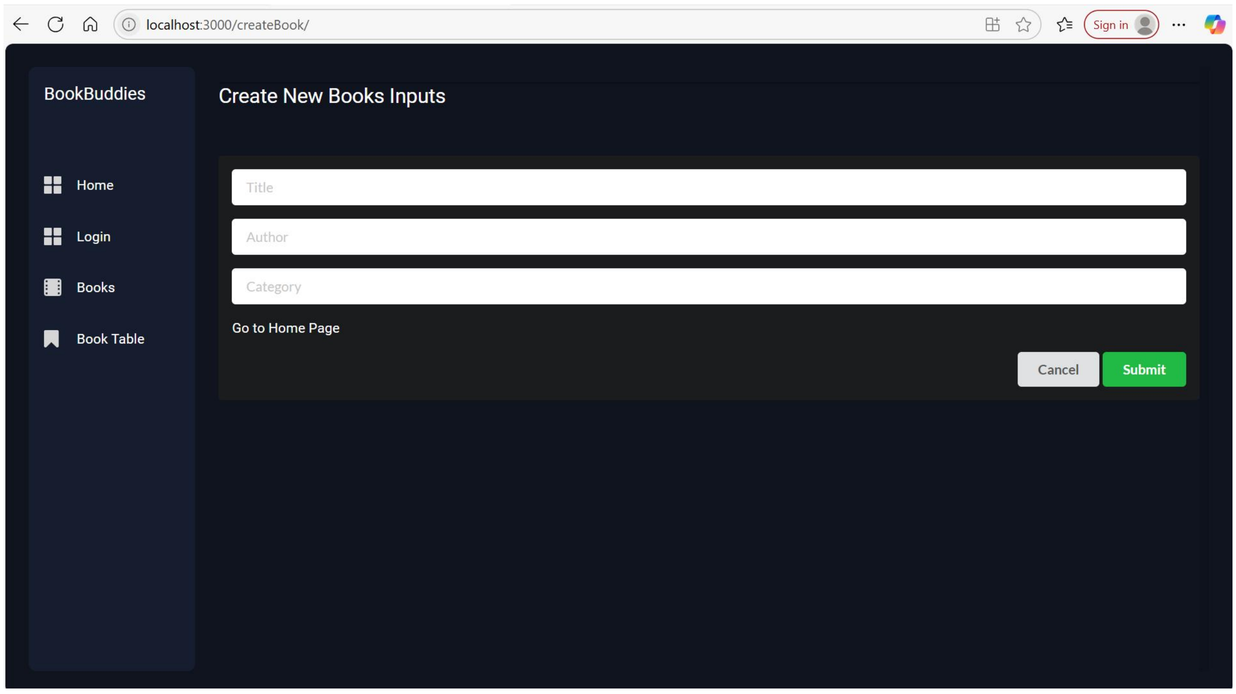
Create New Books Inputs

Id	Author	Title	Category	
4	jahdc;aj	dgjfkjgd	dhjhcg	Edit Delete
5	ifug	tfugyuo	voygoiu	Edit Delete
6	Geogrophy	Perera	Science	Edit Delete
8	Harry Potter	J.K.Rowlin	Adventure	Edit Delete
9	jhh	jsd;aslf	Adventure	Edit Delete
15	GIF	wguidguw	WDIO	Edit Delete
16	vgdh	igdohSD	HSpoiiIOS	Edit Delete

Go to Home Page

Create Book

Create/edit form modal


localhost:3000/createBook/

BookBuddies

Home

Login

Books

Book Table

Create New Books Inputs

Title

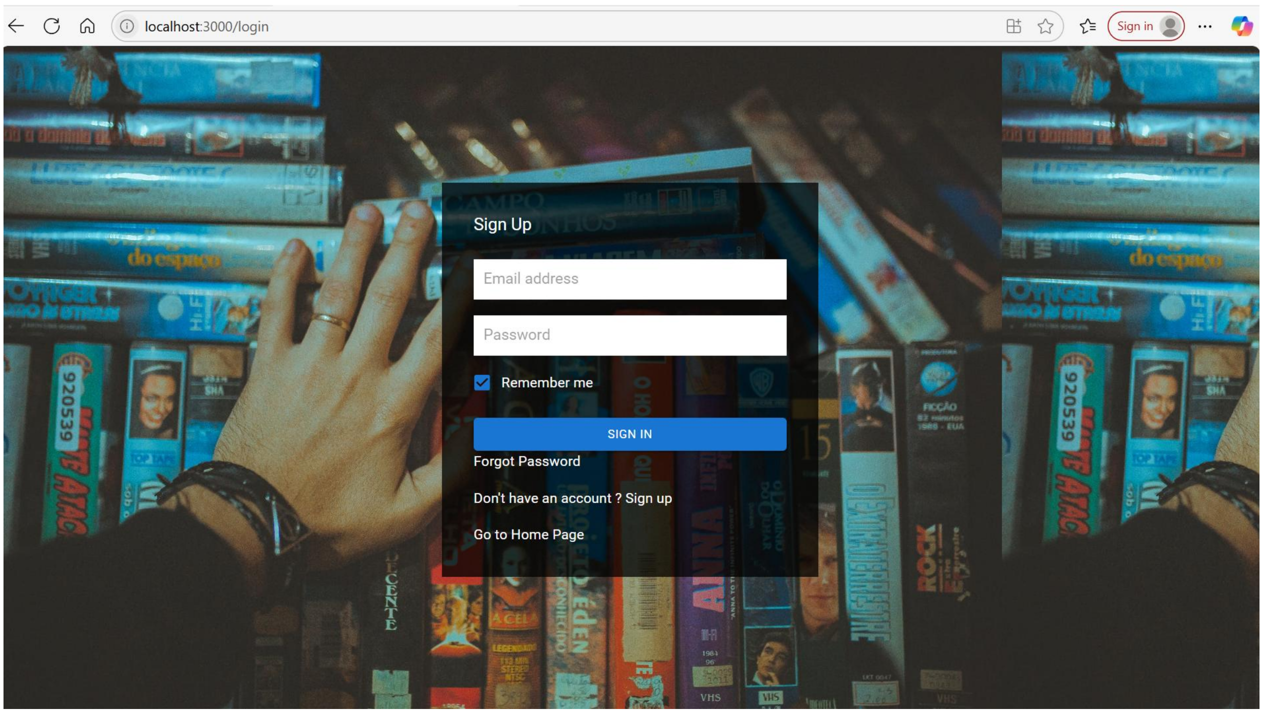
Author

Category

Go to Home Page

Cancel Submit

Login Page


localhost:3000/login

Sign Up

Email address

Password

Remember me

SIGN IN

Forgot Password

Don't have an account ? Sign up

Go to Home Page