Library Management System

Project Report

Submitted by

Deelaka Senal Jayarathna

Undergraduate in Software Engineering

Declaration

I hereby declare that the project report titled “Library Management System” is an original piece of work, created and presented solely by me. This report has not been submitted previously for any academic or professional purposes to any other institution or organization. I affirm that the information contained within this report is accurate to the best of my knowledge and belief.

Deelaka Senal
2025.12.07

Abstract

The **Library Management System** is a simple web application designed to manage library resources and streamline user interactions. The backend is built using ASP.NET Core and Entity Framework Core, ensuring efficient database management and smooth data operations. The frontend is developed with React and styled with pure CSS, providing a clean and responsive user interface. The system features a registration and login functionality, allowing users to securely access the platform. After logging in, all users can view, insert, update, and delete book details, making it a fully interactive solution for managing library operations. The **Library Management System** provides a seamless and straightforward approach to managing book data.

Table of Contents

Acknowledgement

I would like to express my heartfelt gratitude to my instructors and friends for their invaluable guidance and support throughout the development of this project. Special thanks to my friends for their constant encouragement and assistance. I am deeply grateful to my family for their unwavering support, motivation, and belief in me, which has been a source of strength during this journey.

Deelaka Senal

Introduction

The **Library Management System** is a web-based application designed to manage and streamline the operations of a library. It enables users to interact efficiently with the library's catalog, allowing them to perform tasks such as viewing, adding, updating, and deleting book records. The system focuses on simplicity, user-friendliness, and security, ensuring that library operations are carried out in an organized manner.

Overview

The Library Management System provides a straightforward interface for managing books and user data. The backend is developed using **ASP.NET** and **SQLite**, while the frontend is built with **React** and **TypeScript** for a dynamic and responsive experience. Users can securely register, log in, and perform CRUD operations on book records. The system is designed to cater to library staff and regular users, providing an efficient solution for managing library resources.

Objective

The primary goal of this project is to create an effective solution for managing library operations. The objectives include:

- Enabling users to register and securely log in to access the system.
- Providing a user-friendly interface for managing books, including adding, viewing, updating, and deleting book details.
- Ensuring data security and integrity, particularly for user credentials.
- Streamlining library operations by automating processes and improving efficiency.

Scope of the Project

The **Library Management System** includes the following features:

- **User Registration and Login:** Users can create accounts and log in securely.
- **Book Management:** After logging in, users can perform CRUD operations on book records.
- **Database Integration:** The system is connected to an **SQLite** database for storing user and book data, with **ASP.NET** handling backend logic.
- **Frontend Interface:** Developed using **React** with **TypeScript**, and styled with pure CSS, the frontend ensures a responsive and intuitive design.
- **Security:** Basic user authentication is implemented, though the system does not include role-based access control, meaning all users have the same permissions after login.

Methodologies

The development of the **Library Management System** follows an agile approach, allowing for iterative progress and continuous feedback. The system's design and implementation focus on simplicity and ease of use, ensuring a user-friendly interface while adhering to modern software development practices.

Programming Languages

The following programming languages were utilized in the development of the Library Management System:

- **TypeScript:** Used for the front-end development with React, providing static typing and enhancing code reliability.
- **C#**: Employed in backend development in ASP.NET, ensuring efficient handling of user requests, business logic, and database operations.

Backend Implementation

The backend of the Library Management System is developed using **ASP.NET Core**. This provides a robust framework for handling HTTP requests, managing data flow, and ensuring the system's scalability and security. The backend includes:

- **User Authentication**: The system handles user login and registration, storing credentials securely.
- **Book Management**: CRUD (Create, Read, Update, Delete) operations are performed on book records stored in the database.
- **Database Operations**: The backend interacts with an **SQLite** database using **Entity Framework Core**, which simplifies data operations and ensures easy querying and updating of records.

Frameworks and Libraries

The system uses several frameworks and libraries to ensure efficiency and ease of development:

- **React**: A JavaScript library for building the user interface. React's component-based architecture allows for reusable and maintainable code.
- **TypeScript**: A superset of JavaScript, TypeScript adds static typing to JavaScript, improving code quality, early error detection, and maintainability.
- **ASP.NET Core**: A cross-platform framework used for developing the backend. ASP.NET Core is known for its performance, security, and scalability, which are crucial for handling user data and library operations.
- **Entity Framework Core**: An Object-Relational Mapping (ORM) tool that simplifies database operations by mapping objects in C# to relational database tables.
- **SQLite**: A lightweight, serverless database used to store user and book data. SQLite was chosen for its simplicity and ease of integration with ASP.NET Core.

Challenges Faced

During the development of the Library Management System, several challenges were encountered:

- Database Integration: Integrating SQLite with ASP.NET Core posed some initial challenges due to the need for careful management of data migrations and maintaining consistency between the database and the backend logic.
- Authentication Security: Ensuring secure user authentication and password storage required implementing proper password hashing techniques. By using hashing

algorithms, such as bcrypt, the passwords are stored in a hashed form, ensuring that even if the database is compromised, user passwords remain secure.

- UI Design: Designing a clean, responsive user interface using pure CSS without relying on libraries like Tailwind CSS was challenging. Ensuring consistency and usability across different screen sizes required detailed styling and adjustments.

Implementation Plan

The implementation plan for the Library Management System followed these steps:

1. **Planning and Requirements Gathering**: Defined the project's goals, key features, and technology stack. The main requirements were user authentication, book management (CRUD operations), and a responsive UI.
2. **Backend Development**:
    - Set up the ASP.NET Core backend, including the authentication system, user management, and book CRUD functionalities.
    - Integrated **SQLite** as the database and used **Entity Framework Core** for efficient database interaction.
3. **Frontend Development**:
    - Developed the frontend using **React** and **TypeScript**, focusing on creating a responsive, user-friendly interface.
    - Styled the components using pure CSS to ensure a clean, modern design.
4. **Testing**:
    - Conducted unit testing for individual components and functionalities, particularly for backend operations like CRUD and user authentication. ○ Performed end-to-end testing to ensure seamless interaction between the frontend and backend.
5. **Documentation**:
    - Created a detailed report and user manual to provide an overview of the system and guide users on how to navigate the platform.

Solution Design

The **Library Management System** was designed to ensure seamless user experience, efficient management of book records, and secure handling of user data. The solution is divided into two primary components: **Interface Design** and **Database Design**.

Interface Design

The interface of the Library Management System is designed to be simple, intuitive, and user-friendly. The focus was on providing a clean and functional user experience while ensuring easy navigation between different system features.

Login Page:

- **Fields**: Username or Email and Password.
- **Actions**: 'Login' button for user authentication and a 'Register' link for new user registration.
- **Validation**: Ensures that the input fields are not empty and that the user credentials are valid.

Registration Page:

- **Fields**: Username, Email and Password.
- **Actions**: 'Submit' button to register new users.
- **Validation**: Ensures proper email format, password strength, and that the username does not already exist.

Book Management Interface:

- **Book List**: Displays all books with details like title, author description and published year.
- **CRUD Operations**:
    - **Add Book**: Form to input new book details.
    - **Update Book**: Allows users to update book details.
    - **Delete Book**: Button to remove a selected book from the system.

- **Search Functionality**: Allows users to search for books by title, author, or genre.

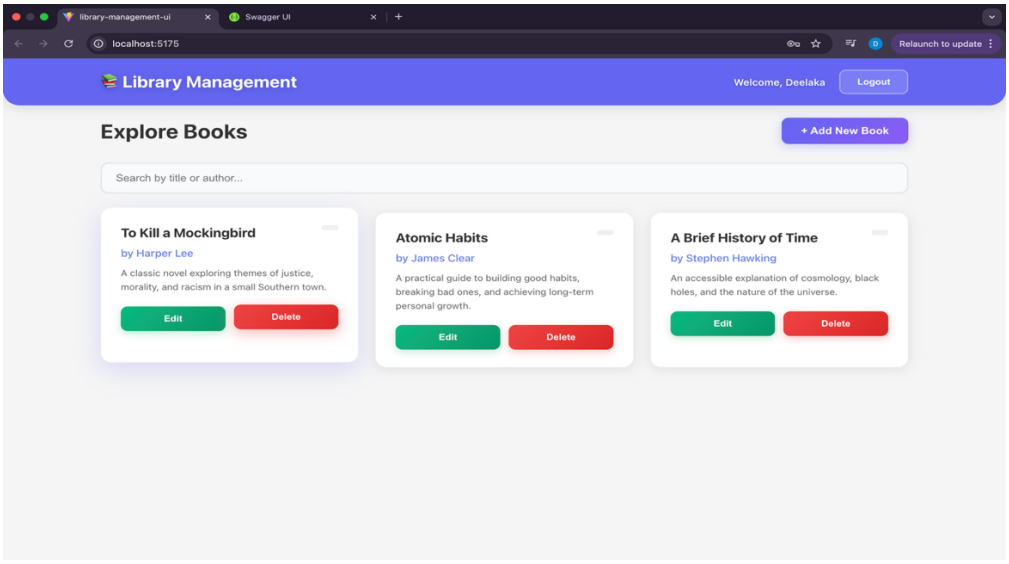

Figure 3 Dashboard

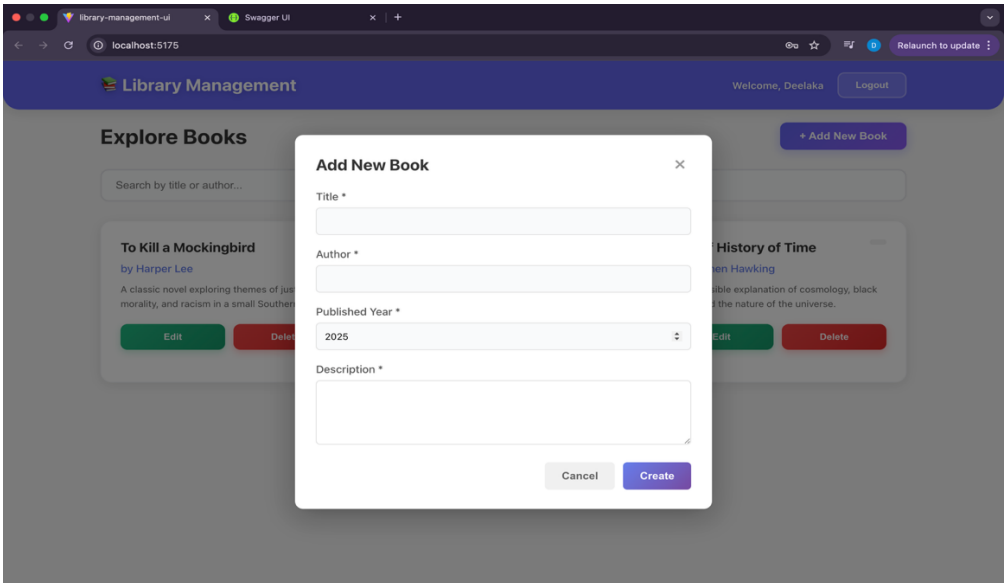

Figure 2 Add Book

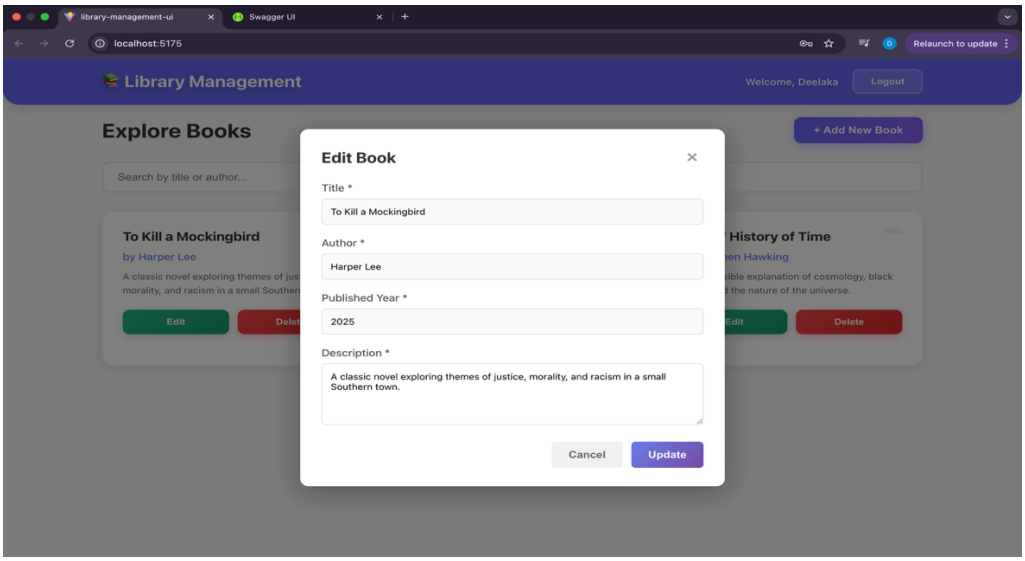

Figure 1 Edit Book

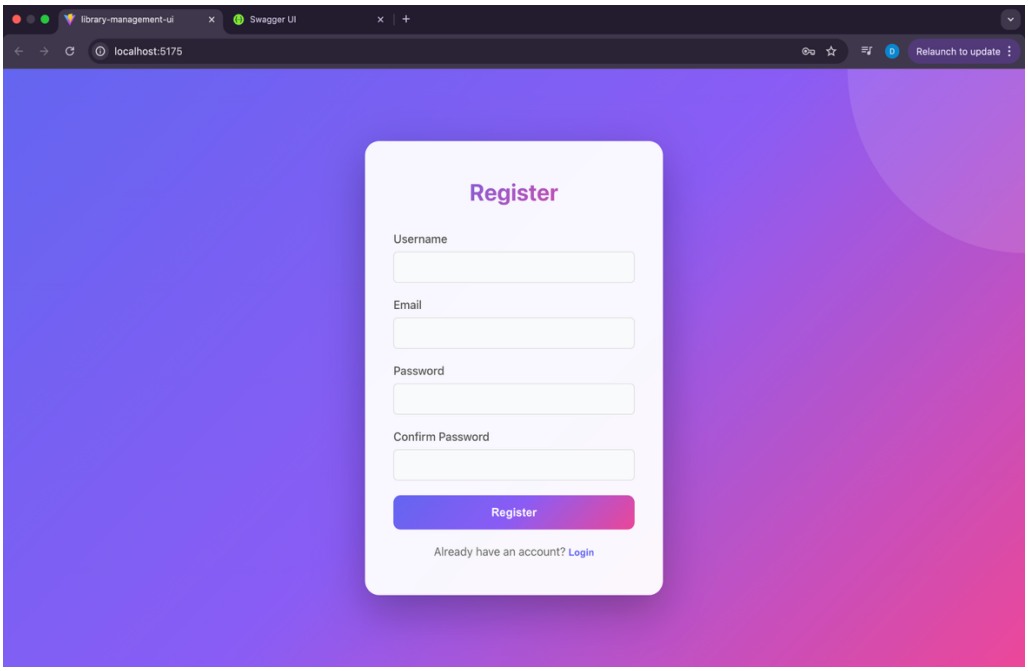

Figure 4 Register

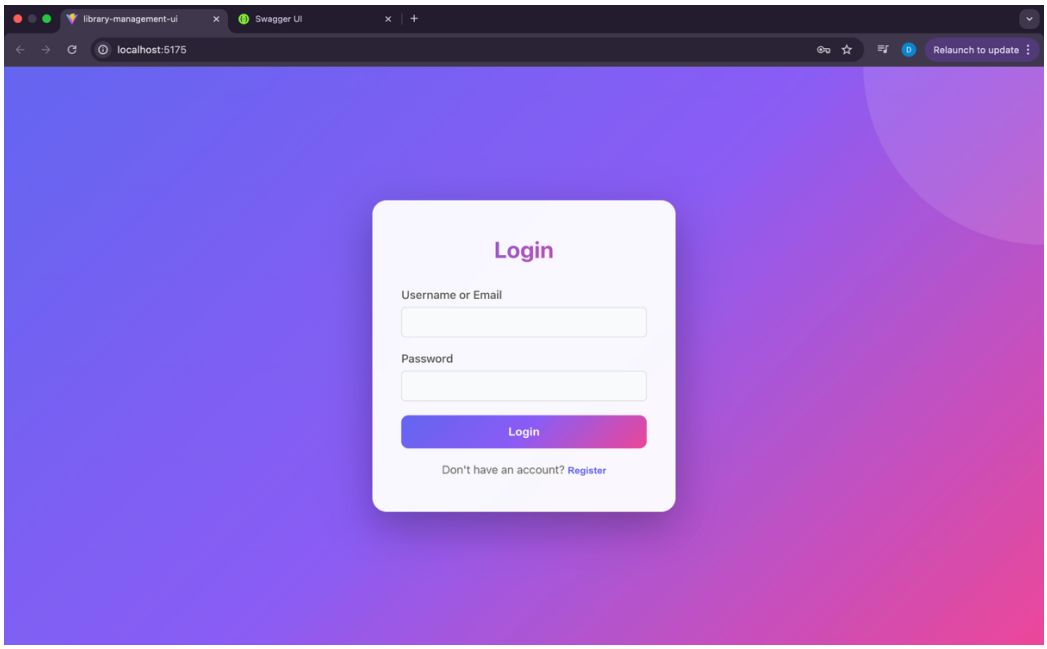

Figure 5 Login

The frontend was developed using **React** and **TypeScript**, with styling done using pure **CSS** to provide a responsive design that works across devices.

Database Design

The **Library Management System** uses an **SQLite** database to store data securely and efficiently. The database schema is designed to handle user information and book records, enabling seamless CRUD operations.

Tables in the Database:

1. **Users Table**:
    - Id (Primary Key): Unique identifier for each user (INTEGER, Auto-increment)
    - Username: User's login username (TEXT, max 50 characters, NOT NULL, UNIQUE)
    - Email: User's email address (TEXT, max 100 characters, NOT NULL, UNIQUE)
    - PasswordHash: Hashed password for secure authentication (TEXT, NOT NULL)
    - FullName: Full name of the user (TEXT, max 100 characters, NULLABLE)
    - CreatedAt: Timestamp when the user account was created (DATETIME, NOT NULL)
    - LastLoginAt: Timestamp of the user's last login (DATETIME, NULLABLE)
2. **Books Table**:
    - Id (Primary Key): Unique identifier for each book (INTEGER, Auto-increment)
    - Title: Title of the book (TEXT, max 200 characters, NOT NULL)
    - Author: Author name of the book (TEXT, max 100 characters, NOT NULL)
    - Description: Description of the book (TEXT, max 1000 characters, NULLABLE)
    - CreatedAt: Timestamp when the book record was created (DATETIME, NOT NULL)
    - UpdatedAt: Timestamp when the book record was last updated (DATETIME, NULLABLE)
    - UserId (Foreign Key): Reference to the user who added the book (INTEGER, NULLABLE, references Users.Id with ON DELETE SET NULL)

Relationships:

- A One-to-Many relationship exists between Users and Books tables. Each book record can be associated with a user through the UserId foreign key. Users can perform CRUD operations on books after logging in. When a user is deleted, their associated books remain in the database with UserId set to NULL.

Database Operations:

- **CRUD Operations on Books**: Handled by ASP.NET Core through Entity Framework Core, allowing easy interaction with the SQLite database. The backend ensures that the books table is updated based on user input, with proper validation and error handling. Users must be authenticated to create, update, or delete books.
- **User Authentication**: The user credentials (username, email and password) are stored securely in the Users table. Passwords are hashed using BCrypt hashing algorithm, ensuring secure login and authentication processes. JWT tokens are used for maintaining authenticated sessions.