# Software Engineering Internship Project Report

## Library Management System (Full-Stack Application)

Name: Raveen Adhikari
Project: Internship Assignment – Software Engineering
Technologies Used: .NET 8, C#, SQLite, React, TypeScript, CSS

Project Repository : https://github.com/raveenadhikari/Libarary-Management-System

## 1. Introduction

This report outlines the development and implementation of a full-stack Library Management System. The goal was to create a functional web application with user authentication and book management features using modern software development technologies and best practices. The Library management system is a very useful tool for many users to keep track of their books and also libraries.

## 2. Project Objectives

- Build a RESTful backend API using ASP.NET Core
- Create a React + TypeScript frontend that communicates with the API
- Implement user registration and login
- Provide book management features: Add, Edit, Delete, View
- Demonstrate good UI/UX design and usability

## 3. Tools and Technologies

| Layer | Technology |
|---|---|
| Backend | ASP.NET Core (.NET 8) |
| Database | SQLite with Entity Framework Core |
| Frontend | React(typescript) |
| Styling | CSS |
| Authentication | JSON WEB Tokens (JWT) |
| Version Control | Git/Github |

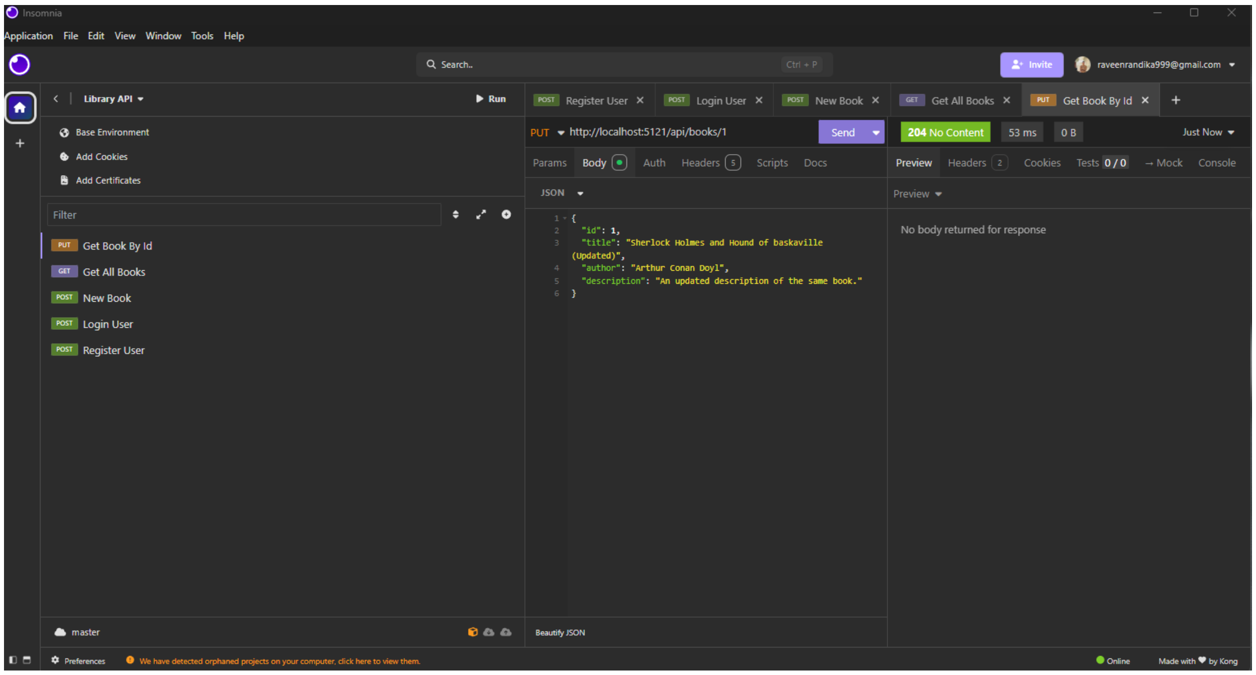
## 4. Methodology

### Backend Implementation:

The backend was developed using ASP.NET Core (.NET 8) with a focus on clean separation of concerns and RESTful design principles. I began by designing the User and Book models and setting up the Entity Framework Core with a SQLite database. I then implemented authentication using JWT, along with secure password hashing and salting.

During development, I tested all backend endpoints using Insomnia. I validated user registration, login, and all CRUD operations for books through JSON requests and inspected both responses and authentication headers. This approach allowed me to test the API independently of the frontend and ensure it functioned correctly before integration.

A screenshot of a sample API test in Insomnia is included below for reference.



PUT http://localhost:5121/api/books/1

```json
{
  "id": 1,
  "title": "Sherlock Holmes and Hound of baskaville (Updated)",
  "author": "Arthur Conan Doyl",
  "description": "An updated description of the same book."
}
```

204 No Content 53 ms 0 B

No body returned for response

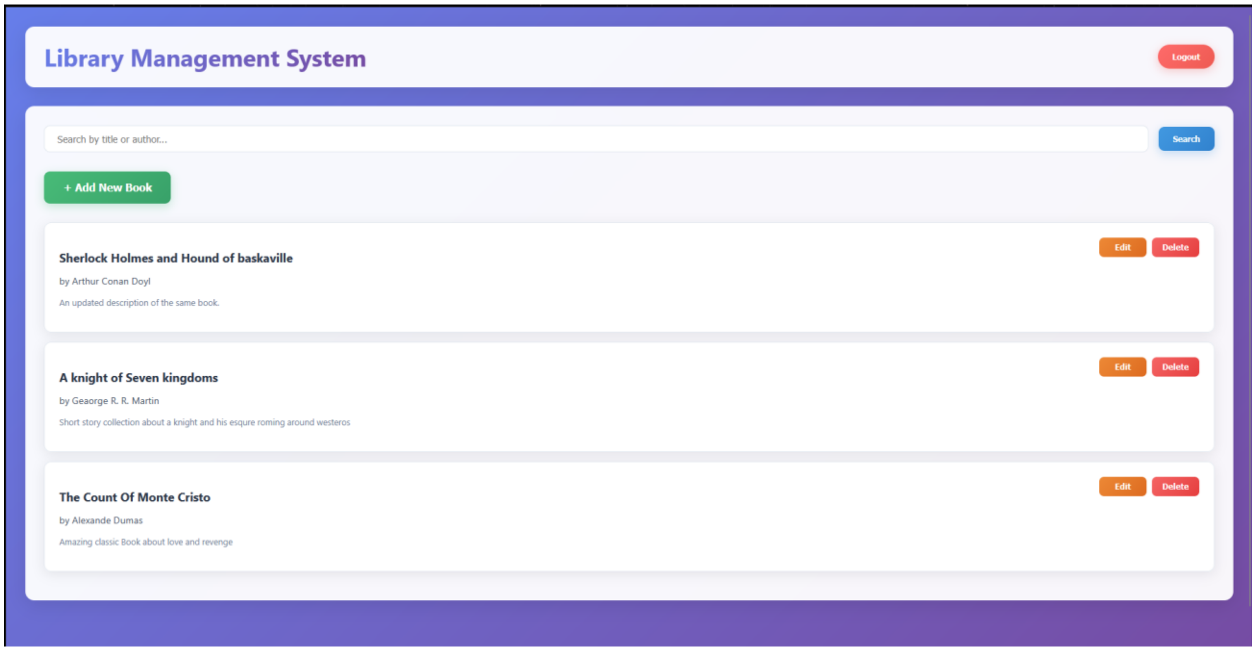
JWT (JSON Web Token) was used in this project to implement stateless, secure user authentication. When a user logs in successfully, the server issues a token that contains encoded information (user ID). This token is signed using a secret key and returned to the client.

### Frontend Implementation:

The frontend was developed using React with TypeScript to build a modular, maintainable, and type-safe interface. The application is structured using a component-based approach with separate pages for login, registration, and dashboard functionality.

All communication with the backend API is handled using a shared Axios instance, which includes the JWT token automatically for authenticated requests.

Below is a screenshot of the Homepage of the web app :



**Sherlock Holmes and Hound of baskaville**
by Arthur Conan Doyl
An updated description of the same book.

**A knight of Seven kingdoms**
by Geaorge R. R. Martin
Short story collection about a knight and his esqure roming around westeros

**The Count Of Monte Cristo**
by Alexande Dumas
Amazing classic Book about love and revenge

## 5. Application Features

### Authentication

- JWT-based token handling for secure login
- Tokens stored in browser local storage
- Protected routes ensure only authenticated users can access the dashboard

Book Management

- Add new books with title, author, and description
- Edit existing book records
- Delete books from the list
- Books fetched from the database using secure API endpoints

Search

- Client-side book search by title or author
- Instant filter with a “Search” and “Clear” button for improved usability

UX Enhancements

- Password confirmation and toggle visibility on register/login forms
- Logout button with token removal

## 10. Challenges Faced

- Handling CORS policies between React and ASP.NET
- Debugging token validation and key size errors when API testing.
- Managing consistent TypeScript types across components
- Fine tuning user experience for better usability

## 11. Learning Outcomes

- Gained hands on experience in building secure, full- stack applications
- Learned to integrate JWT authentication with frontend routing
- Improved TypeScript skills and modular React architecture
- Understood practical software engineering workflow including GitHub based collaboration and version control

## 12. Conclusion

The project was successfully implemented meeting all functional and technical requirements. Additional features such as search, password validation, and responsive UI were added to enhance usability and quality. The experience reinforced my full-stack development skills and prepared me for real world engineering roles.