Library Management System – Project Report

1. Introduction

This project is a simple Library Management System built with a .NET backend and a React frontend.
The goal of the system is to let users Add, View, Edit, and Delete books. Each book has a Title, Author, and Description.

2. Backend (ASP.NET Core + SQLite)

The backend was developed using ASP.NET Core Web API.

What it does:

- Stores book details in a SQLite database.
- Provides API endpoints to manage books.
- Uses Entity Framework Core for database operations.
- Has Swagger for testing the API easily.

API Endpoints:

- GET /api/books → Get all books
- GET /api/books/{id} → Get a book by ID
- POST /api/books → Add a new book
- PUT /api/books/{id} → Update a book
- DELETE /api/books/{id} → Delete a book

3. Frontend (React + TypeScript)

The frontend was built using React with TypeScript.

What it does:

- Shows a list of all books.
- Has a form to add new books.
- Lets the user edit or delete a book.
- Makes API calls to the backend using Axios.
- Uses simple CSS for styling to keep the UI clean.

Components:

- BookForm → Add a new book
- BookList → Show all books with edit/delete buttons
- BookEditForm → Edit an existing book

4. Challenges I Faced

- My biggest challenge was that I had no prior project experience with .NET.
- I had to learn the project structure (controllers, models, DbContext, etc.).
- Setting up SQLite with Entity Framework and running migrations was new to me.
- I do have some experience with Spring Boot, and I noticed the architecture is quite similar to .NET (controllers, database connection, dependency injection). This helped me learn faster, but I still had to get used to new commands and syntax.

5. Key Learnings

- Learned how to build a .NET Web API project step by step.
- Understood how Entity Framework Core makes it easy to work with databases.
- Practiced connecting a React frontend with a .NET backend.
- Saw how Swagger automatically creates API documentation and allows testing.

6. Conclusion

This project gave me good practice in full-stack development.
Even though I faced challenges learning .NET and SQLite, I was able to complete the project and make it work.

With this project I showed that I can:

- Build a backend with ASP.NET Core + SQLite
- Build a frontend with React + TypeScript
- Connect both sides together