Library Website Project Structure

This document outlines the full project structure for a public library website with optional membership, book borrowing, and room booking. The project uses a **frontend in HTML/CSS/JS** and a **backend in ASP.NET Core Web API (C#)**.

📁Project Root

```
LibraryWebsite/
├── frontend/
├── backend/
```

📁Frontend (HTML/CSS/JS)

```
frontend/
├── index.html                 // Home page
├── books.html                 // Public book catalog
├── book-details.html         // Single book detail
├── login.html
├── register.html
├── borrow.html               // My borrowed books
├── rooms.html                // Available rooms
├── my-bookings.html          // Room booking history
├── membership.html           // Apply for optional membership
├── admin-dashboard.html      // Admin panel UI
│
├── css/
│   └── styles.css            // All styles
│
└── js/
    ├── main.js              // Common navigation & auth
    ├── books.js             // Book-related logic
    ├── auth.js              // Login/register logic
    ├── borrow.js            // Borrow/return logic
    ├── rooms.js             // Room booking logic
    └── admin.js             // Admin operations
```

📁Backend (ASP.NET Core Web API)

```
backend/
├── Controllers/
│   ├── AuthController.cs
│   ├── BooksController.cs
│   ├── BorrowController.cs
│   ├── RoomBookingController.cs
│   ├── MembershipController.cs
│   └── AdminController.cs
│
├── Models/
│   ├── User.cs
│   ├── Book.cs
│   ├── BorrowRecord.cs
│   ├── Room.cs
│   ├── RoomBooking.cs
│   └── Membership.cs
│
├── Data/
│   └── LibraryContext.cs     // Entity Framework DbContext
│
├── Program.cs
├── appsettings.json
└── backend.csproj
```

🔗Communication

- Frontend uses `fetch` to call backend APIs (e.g., `GET /api/books`)
- Backend returns JSON data for all operations

Roles

Role	Can Borrow	Can Book Room	Notes
Guest	✅(Limited)	✅(Limited)	Register required
Member	✅(Extended)	✅(Flexible)	Optional membership
Admin	✅	✅	Manage all data

📌Optional Features

- JWT authentication for login
- Role-based access control
- LocalStorage/sessionStorage for frontend state
- Admin dashboard with metrics