Library Management System

# Database Schema

1. **Books Table**:
   * BookID (Primary Key)
   * Title
   * AuthorID (Foreign Key referencing Authors table)
   * PublishedDate
   * Genre
2. **Authors Table**:
   * AuthorID (Primary Key)
   * Name
   * Biography
   * DateOfBirth
3. **Members Table**:
   * MemberID (Primary Key)
   * Name
   * Email
   * Phone
   * Address
   * DateJoined
4. **Staff Table**:
   * StaffID (Primary Key)
   * Name
   * Email
   * Phone
   * Position
5. **BorrowedBooks Table**:
   * BorrowID (Primary Key)
   * BookID (Foreign Key referencing Books table)
   * MemberID (Foreign Key referencing Members table)
   * BorrowedByStaffID (Foreign Key referencing Staff table)
   * ReturnedByStaffID (Foreign Key referencing Staff table, NULL if not yet returned)
   * BorrowDate
   * ReturnDate

# Use Cases

## **Staff Use Cases**

1. **Login**
   * Staff members can log in to the system using their credentials.
   * **Change Profile Details**
     + Staff members can update their own profile information, such as email, phone number, and address.
2. **Manage Books**
   * **Add a New Book**
     + Staff can add new books to the library's catalog by providing details such as title, author, published date, and genre.
   * **Edit Book Information**
     + Staff can update information about existing books in the catalog, such as correcting a title or updating the author information.
   * **Delete a Book**
     + Staff can remove books from the catalog if they are no longer available or needed.
   * **View Book List and Search**
     + Staff can view the entire book catalog and search for specific books based on various criteria (e.g., title, author, genre).
3. **Manage Borrowals**
   * **View All Borrowals**
     + Staff can view all borrowing and returning transactions to monitor the activity and ensure that books are being returned on time.
   * **Borrow for a Member**
     + When a member visits the library and requests to borrow a book, staff can assist by recording the borrowing transaction. This involves entering the member ID, book ID, and their own staff ID (BorrowedByStaffID) into the system.
   * **Return for a Member**
     + When a member visits the library to return a book, staff can assist by recording the return transaction. This involves entering the member ID, book ID, and their own staff ID (ReturnedByStaffID) into the system.
4. **Manage Staff**
   * **Add Staff**
     + Senior staff or admin can add new staff members to the system by collecting their details and adding them.
   * **Remove Staff**
     + Senior staff or admin can deactivate staff members from the system if they are no longer part of the team.
5. **Manage Members**
   * **Edit Member Information**
     + Staff can update the personal information of existing members, such as email address and phone number.
   * **Remove Member**
     + Staff can deactivate members from the system if they are no longer part of the library.

## **Member Use Cases**

1. **Register**
   * A new user can register to become a member of the library by providing their details (e.g., name, email, phone, address).
2. **View Borrowals**
   * Members can view their borrowing history to see the books they have borrowed and returned. This can be done either online or by visiting the library and requesting the information from a staff member.
3. **Manage Profile**
   * **Update Personal Information**
     + Members can update their personal information, such as email address, phone number, and address, either online or by visiting the library.
   * **Change Password**
     + Members can change their account password to ensure their information remains secure.

# Technologies Stack

* ASP.NET Core
* C#
* ASP.NET identity
* Entity Framework Code First
* Specification Pattern
* Repository Pattern
* SQL Server
* GitHub
* GitHub Actions
* Angular
* Typescript
* SCSS (use @Mixins, built in functions like lighted/darken…etc, programmer defined variables, nested styles which are not possible with CSS)
* Tailwind CSS - for styling
* Angular Material - Angular UI components
* Test Cases – Nunit
* Logging
* Azure Deployment

# Sprints

A detailed sprint plan with items for developing the Library Management System using ASP.NET Core and Angular:

**Sprint 1: Project Setup and Basic Framework**

* **Set up Git repository**
  + Initialize the repository
  + Create .gitignore file
  + Commit initial setup
* **Set up ASP.NET Core backend**
  + Create a new ASP.NET Core project
  + Install necessary Entity Framework Core packages
  + Create the database context
  + Configure the database connection
    - Example Controller
      * HTTP GET – all
      * HTTP GET – by Id
      * HTTP PUT
      * HTTP POST
      * HTTP DELETE
    - Unit of Work Pattern
    - Generic Repository
    - Specific Repositories
    - Auto mapper
* **Set up Angular frontend**
  + Install Angular CLI
  + Create a new Angular project
  + Commit initial setup

**Sprint 2: User Authentication and Profile Management**

* **Backend: User Authentication**
  + Implement member and staff registration endpoints
  + Implement login endpoints for members and staff
  + Set up authentication and session management
* **Frontend: User Authentication**
  + Create registration and login components
  + Implement forms for registration and login
  + Integrate authentication with the backend
* **Backend: Profile Management**
  + Implement endpoints for updating profile details
  + Implement password change functionality
* **Frontend: Profile Management**
  + Create profile management components
  + Implement forms for updating profile details
  + Integrate profile management with the backend

**Sprint 3: Book Management**

* **Backend: Book Management**
  + Create endpoints for adding new books
  + Create endpoints for updating existing book details
  + Create endpoints for deleting books
  + Create endpoints for viewing and searching the book catalog
* **Frontend: Book Management**
  + Create book list component
  + Create add/edit book components
  + Create delete book functionality
  + Implement search functionality
  + Integrate book management with the backend

**Sprint 4: Borrowing and Returning Books**

* **Backend: Borrowing Books**
  + Create endpoints for recording book borrowing transactions
  + Validate and store transaction information in the database
* **Backend: Returning Books**
  + Create endpoints for recording book return transactions
  + Update transaction information in the database
* **Frontend: Borrowing Books**
  + Create borrowing components for staff
  + Implement forms for recording borrowing transactions
  + Integrate borrowing functionality with the backend
* **Frontend: Returning Books**
  + Create returning components for staff
  + Implement forms for recording return transactions
  + Integrate return functionality with the backend
* **Backend & Frontend: Viewing Borrowing History**
  + Create endpoints for members to view their borrowing history
  + Implement history viewing components for members
  + Integrate borrowing history functionality with the backend

**Sprint 5: Member and Staff Management**

* **Backend: Manage Members**
  + Create endpoints for editing member information
  + Create endpoints for removing members
* **Backend: Manage Staff**
  + Create endpoints for adding staff members
  + Create endpoints for removing staff members
* **Frontend: Manage Members**
  + Create member list component
  + Create edit member component
  + Implement remove member functionality
  + Integrate member management with the backend
* **Frontend: Manage Staff**
  + Create staff list component
  + Create add/edit staff components
  + Implement remove staff functionality
  + Integrate staff management with the backend

**Sprint 6: Testing and Deployment**

* **Write unit tests for backend**
  + Ensure all endpoints are thoroughly tested
* **Write unit tests for frontend**
  + Ensure all components are thoroughly tested
* **Set up continuous integration and deployment (CI/CD)**
  + Automate testing processes
  + Automate deployment processes
* **Deploy the application**
  + Deploy the ASP.NET Core backend
  + Deploy the Angular frontend
  + Ensure both are seamlessly integrated

**Sprint 7: User Interface Enhancements (Optional)**

* **Develop user-friendly interface**
  + Improve UI/UX based on feedback and testing
  + Implement responsive design
* **Enhance user experience**
  + Add additional features based on user needs
  + Continuously iterate based on feedback

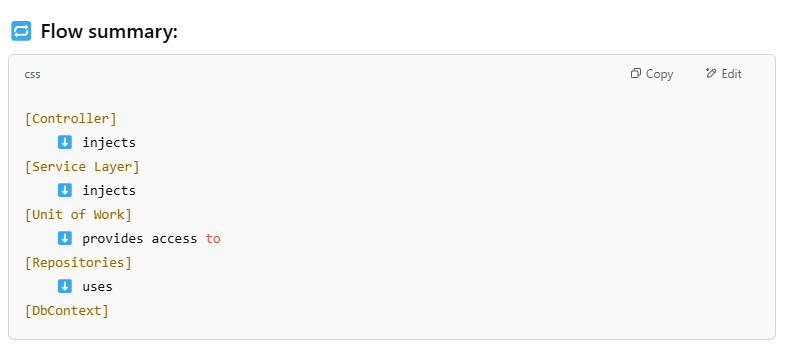
# Architecture

|  |
| --- |
| API |

|  |
| --- |
| Infrastructure |

|  |
| --- |
| Core |

## Project Data Flow



## UI