# ASP.NET Core Teamwork Assignment

## Project Description

Design and implement a **Library Management System**, where the users can search, checkout, return books, etc.

Each requirement is categorized in one of three categories – **must**, **should** or **could**.

- Must requirements have the highest priority and should be addressed first.
- Should requirements have medium priority and should be addressed after all or most must requirements have been implemented and tested.
- Could requirements should be left for last.

The application should have:

- **public part** (accessible without authentication)
- **private part** (available for registered users)
- **administration part** (available for **librarian** users only)

## Public Part

The **public part** of your library system should be **accessible without authentication**.

This section **must** support the following functionalities:

### Homepage

The homepage must be the landing page for guest users. It should provide engaging content such as list of random books, most popular books, etc.

### Register

A register functionality must exist with at least a **username** field and a **password** field, which are both **client-side and server-side validated**. Two users with the same username cannot exist.

After registration, the user must be prompted to pay a membership fee. (1-month*(20 $)* or 1-year*(200 $)*)

### Login

A login functionality must exist with at least **username** field and a **password** field.

### Logout

A logout functionality must exist.

## Private Part

The **private part** of your library system should be **accessible for registered users**.

User with **invalid membership** (*have not paid book fee or membership expired*) can only **search** and **return**, but cannot **checkout**, **reserve**, or **renew**.

This section **must** support the following functionality:

---

## Search book

Provide a functionality to search a book by ISBN, title, author, genre, publisher or year. You can also add a checkbox which controls how multiple search criteria are applied:

- inclusive (e.g. title && author && …)
- exclusive (e.g. title || subject || …)

If the user inputs only one search criteria, the checkbox can be ignored

## Reserve book

A user has the option to reserve a book.

- If the book is reserved while its status is available – the user has 10 days to checkout this book. If he does not manage to check out the book in this 10 days period – the book is return back to status available automatically and a notification is sent to the user or is being reserved by the next user in line, who has also 10 days to check it out.
- If the book is reserved while its status is checked-out – after the previous user returns the book, the user, who wants to reserve it, must receive a notification that the book is available for him and has 10 days to checkout it out. If he does not manage to check out the book in this 10 days period – the book is return back to status available automatically and a notification is sent to the user or is being reserved by the next user in line, who has also 10 days to check it out.

Librarians must receive notification of every reservation state of the book.

## Renew book

A user has the option to extend the return date of a borrowed book. This action should cost him 5 $ per renewal and will last 5 days. A User can unlimitedly renew a book, as long as he has enough money in his Wallet.

The date cannot exceed the end date of a valid membership of a user.

Librarians must receive notification that the book is renewed.

## Return book

A user must return a book in 10 days. The user can manually return a book before that date.

If the user does not return the book on time, he goes into Restricted regime and will only has access to Search and Return book functionalities. Every overdue day will cost 1 $. The user remains restricted until he pays the fine and returns the overdue book.

Librarians must receive notification that a book is returned.

## Checkout book

A user has the option to checkout a book for a 10 days. If the book is currently taken, it must be redirected to the ‘Reserve book’ functionality.

Librarians should receive notification that a book is checked-out.

### Read notifications

The user should be able to see their notifications, sorted by date in descending order (most recent to the top)

This section **should** support the following functionality:

### Mark notification as 'seen'

The users have the option to mark a notification as read. Read notifications are not part of the notifications page, unless the user specifically goes to page such as "View read notifications"

### Review book

The user should be able to provide a rating (1-10) and a comment, giving a review of book. If this functionality is implemented, searching a book should return the books with their highest review and their lowest review (if any).

This section **could** support the following functionality:

### Cancel membership

A user has the option to cancel their membership. Any checked-out or reserved books are immediately returned. If the user has overdue books, he would not be able to cancel the membership.

Librarians receive a notification.

## Administration Part

The **administration part** of your library management system should be accessible only for **librarian** users.

This section **must** support the following functionality:

### Create/remove books.

Librarians have the option to create books. When creating a book, the librarian should also specify how many copies the library will have.

Librarians can remove books, but books that are currently checked-out cannot be removed – in this case a book is scheduled for deletion (it is removed from any reservation lists and the checkout period cannot be renewed)

### Ban users

A librarian should be able to ban users. The ban can have expiration time and description field. Banned users cannot login into the system

This section **could** support the following functionality:

### Lock book from checking-out/reserving

A librarian could be able to lock/unlock any book. All copies are affected - in this case it is removed from any reservation lists and the checkout period cannot be renewed.

# General Requirements

This section **must** support the following functionality:

- You **must** use **Git** to keep your source code and for team collaboration.
- You **must** use **Trello** for project management.
- You **must** follow the **SOLID principles** and the **OOP principles**. The lack of **SRP** or **DI** will be punished by **death**.
- You **must** use correct naming and write clean, **self-documenting code**.

## Backend Requirements

This section **must** support the following functionality:

- You **must** use **ASP.NET Core 2+**
- You **must** use **EF Core 2+**
- You **must** use **SQL Server**.
- You **must** use **services** for the business logic.
- You **must** have at least five types of **database entities.**
- You **must** provide at least two type of **relations in the database.**
- You **must** apply proper **server-side data validation.**
- You **must** apply proper **error handling.**
- You **must** write **unit tests** for the business layer.

## Frontend Requirements

This section **must** support the following functionality:

- You **must** use **Razor**
- You **must** create **usable and responsive UI** (use **Bootstrap/Materialize**).
- You **must** apply proper **client-side data validation.**
- You **must** apply proper **error handling.**