Exam Preparation

Library

Problems for the exam preparation for the "ASP.NET Core Fundamentals" course @ SoftUni

Library is an online platform that is used to create and collect books.

1. Technological Requirements and Overview

Use the provided skeleton – **Library_Skeleton**. All of the needed packages have been installed.

The Technological Requirements are ABSOLUTE. If you do not follow them, you will NOT be scored for other Requirements.

The provided skeleton consists of:

- Areas/Identity/Pages you have to scaffold Identity here
- Controllers you should implement the controllers logic here
- Data you should hold the entities models here
- Models you should implement the models here
- Views you are provided with the needed views. Your task is to implement some logic regarding the loggedin/logged-out user
- Appsettings.json don't forget to change the Connection string
- **Program.cs** you should fulfil the security and password requirements here

NOTE: You should seed the database with provided in advance data regarding the Category and Book entity. In order to do this, remove the comments from the block of code in the protected override void OnModelCreating(ModelBuilder builder) method of the DbContext.

NOTE: Don't forget to uncomment the code inside the views while you implement your logic.

Now that you know the **Technological Requirements**, let us see what the **Functional Requirements** are.

2. Identity Requirements

You should scaffold Identity and use the default IdentityUser.

Remove the unnecessary code from the Login.cshtml and Register.cshtml files and leave only the needed code in order for the app to be functioning correctly.

The **password** requirements for the **IdentityUser** are the following:

Require confirmed account: false

Require digits: false

Require non-alphanumeric characters: false

• Required uppercase letters: false

3. Database Requirements

The **Database** of **Library**:



















Book

- Has **Id** a unique **integer**, **Primary Key**
- Has Title a string with min length 10 and max length 50 (required)
- Has Author a string with min length 5 and max length 50 (required)
- Has **Description** a **string** with min length **5** and max length **5000** (**required**)
- Has ImageUrl a string (required)
- Has Rating a decimal with min value 0.00 and max value 10.00 (required)
- Has CategoryId an integer, foreign key (required)
- Has Category a Category (required)
- Has **UsersBooks** a collection of type **IdentityUserBook**

NOTE: You are free to use your own Image URLs. If you don't want to use your own URLs, you can use the following two:

https://img.freepik.com/premium-photo/white-empty-books-mockups-template-3d-rendering 613230-382.jpg?w=1060

https://img.freepik.com/free-photo/concept-exams-tests-space-text 185193-79222.jpg?w=1380&t=st=1666039091~exp=1666039691~hmac=f17f061a73cc0d6055208ea2945dccd9ce2112420a5 52e7e0e9ff1ccbd9b1d52

Category

- Has **Id** a unique **integer**, **Primary Key**
- Has Name a string with min length 5 and max length 50 (required)
- Has Books a collection of type Books

IdentityUserBook

- CollectorId a string, Primary Key, foreign key (required)
- Collector IdentityUser
- BookId an integer, Primary Key, foreign key (required)
- Book Book

Implement the entities with the correct datatypes and their relations.

Feel free to use the new syntax for realization of the many-to-many relation without a mapping table.











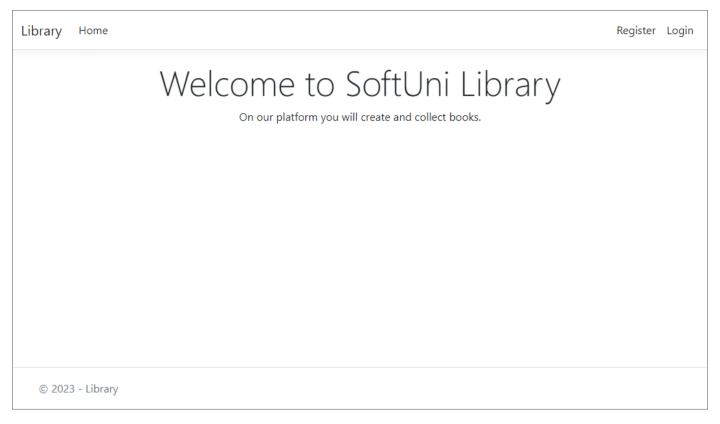




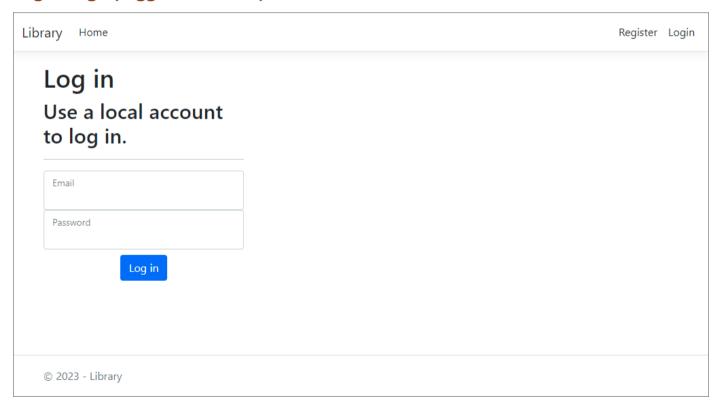


4. Page Requirements

Index Page (logged-out user)



Login Page (logged-out user)









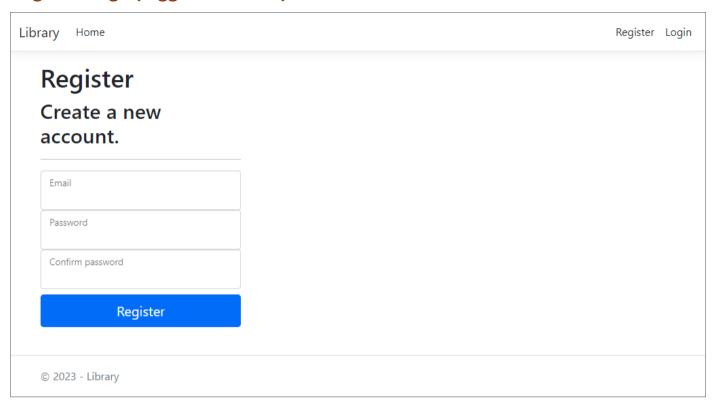




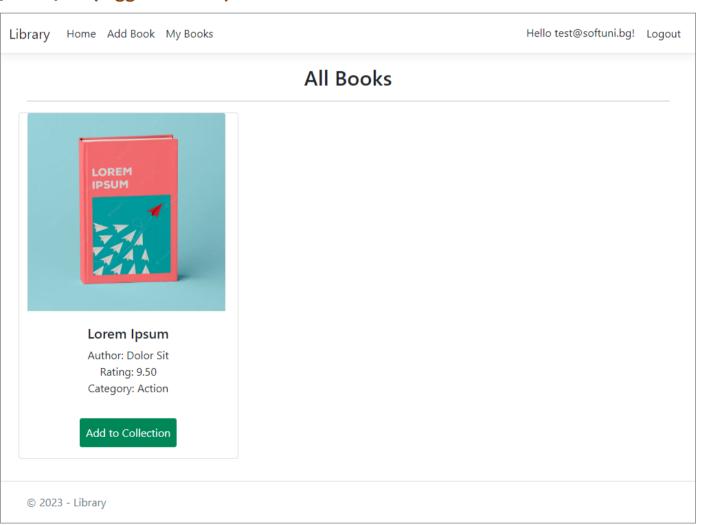




Register Page (logged-out user)



/Book/All (logged-in user)















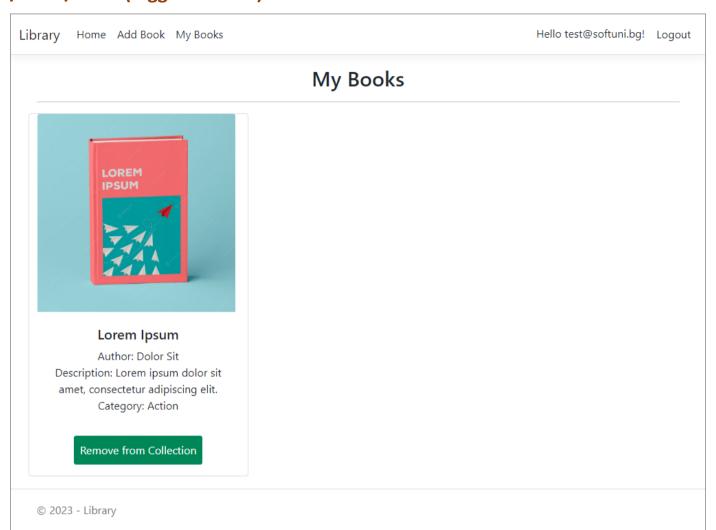






NOTE: If the user is logged in and tries to go to the Home page, the application must redirect them to the /Book/All.

/Book/Mine (logged-in user)









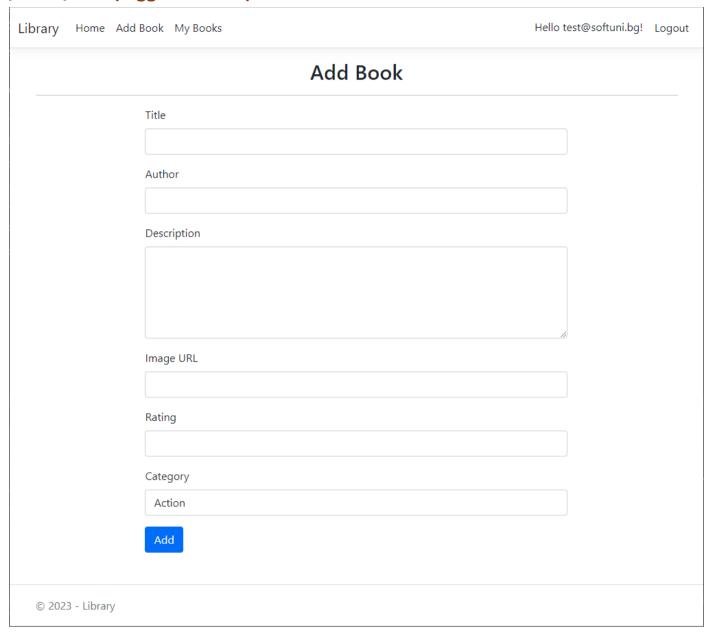








/Book/Add (logged-in user)



/Book/AddToCollection?id={bookId} (logged-in user)

Adds the selected book to the user's collection of books. If the book is already in their collection, it shouldn't be added. If everything is successful, the user must be redirected to the home "/Book/All" page.

/Book/RemoveFromCollection?id={bookId} (logged-in user)

Removes the selected book from the user's collection of books. If everything is successful, the user must be redirected to their collection "/Book/Mine" page.

NOTE: The templates should look **EXACTLY** as shown above.

5. Functionality

The functionality of the **BookLibrary** Platform is very simple.

Users

Guests can **Register**, **Login** and view the **Index Page**.



















IdentityUsers can Add Books and see added Books by all Users on the Home Page (/Book/All). From the Home Page (/Book/All), they can also view Info about each one of those Books and Add them to their collection.

Books

Books can be Added by Users. All created Books are visualized on the Home Page (/Book/All), each one in its separate rectangular element.

Books are visualized on the Home Page (/Book/All) with all their information.

Books are visualized on the Home Page (/Book/All) with a button – [Add to Collection].

The [Add to Collection] button adds the Book to the User's collection of Books, unless it is already added.

Users have a My Books page where only the Books in their collection are visualized.

The [Remove from Collection] button removes the Book from the User's collection of Books.

Redirections

- Upon successful **Login** of an **IdentityUser**, you should be redirected to the /Book/All.
- Upon successful **Creation** of a **Book**, you should be redirected to the /**Book/All**.
- Upon successful Adding a Book to the User's collection, should be redirected to the /Book/All.
- Upon successful **Removal** of a **Book** from the **User**'s collection, should be redirected to the **/Book/Mine**.
- If a User tries to add an already added Book to their collection, they should be redirected to /Book/All (or just a page refresh).
- Upon successful **Logout** of a **User**, you should be redirected to the **Index Page**.
- If any of the validations in the POST forms don't pass, redirect to the same page (reload/refresh it).

6. Security

The **Security** section mainly describes access requirements. Configurations about which users can access specific functionalities and pages:

- **Guest** (not logged in) users can access the **Index** page.
- Guest (not logged in) users can access the Login page.
- **Guest** (not logged in) users can access the **Register** page.
- Guests (not logged in) cannot access Users-only pages.
- Users (logged in) cannot access Guest pages.
- **Users** (logged in) can access the **Book/Add** page and functionality.
- **Users** (logged in) can access the **Book/All** page.
- **Users** (logged in) can access the My Books page.
- **Users** (logged in) can access **Logout** functionality.

7. Code Quality

Make sure you provide the best architecture possible. Structure your code into different classes, follow the principles of high-quality code (SOLID). You will be scored for the Code Quality and Architecture of your project.















8. Scoring

Identity Requirements – 5 points

Database Requirements – 10 points

Template Requirements – 10 points

Functionality – 50 points

Security – 5 points

Code Quality – 10 points

Data Validation – 10 points











