# ASP.NET Fundamentals Exam – 18 February 2024

# SeminarHub

Problems for the exam for the ["ASP.NET Core Fundamentals"](https://softuni.bg/trainings/4105/asp-net-fundamentals-may-2023) course @ SoftUni

**Sellorama** is an online platform where people can easily find and sign up for various seminars and events. It's a place where organizers and participants come together to explore, learn, and share knowledge on a wide range of topics..

## Technological Requirements and Overview

* Use the provided skeleton – **SeminarHub\_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 logged-in/logged-out user
* **appsettings.json** – don't forget to change the your **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 **Semiar** and **Category** 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.

## Identity Requirements

You should **scaffold** **Identity** and use the **default** **IdentityUser**.

Remove the unnecessary code from the **Login.cshtml** and **Register.cshtml** files to unsure proper application functionality.

**NOTE: Don't worry about the views' style – once you scaffold Identity and remove the unnecessary code, the Login and Register pages should look like shown below. Don't add any classes to the views of those two pages!**

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**

## Database Requirements

The **Database** of **SeminarHub**:

### Seminar

* Has Id – a unique **integer, Primary Key**
* Has Topic – string with min length **3** and max length **100** (**required**)
* Has Lecturer – string with min length **5** and max length **60** (**required**)
* Has Details – string with min length **10** and max length **500** (**required**)
* Has OrganizerId – string (**required**)
* Has Organizer – IdentityUser (**required**)
* Has DateAndTime – DateTime with format "**dd/MM/yyyy HH:mm**" (**required**) (the **DateTime** format is recommended, if you are having troubles with this one, you are free to use another one)
* Has Duration – **integer** valuebetween **30** and **180**
* Has CategoryId – **integer, foreign key (required)**
* Has Category – Category (**required**)
* Has **SeminarsParticipants** – a collection of type **SeminarParticipant**

### Category

* Has Id – a unique **integer, Primary Key**
* Has Name – string with min length **3** and max length **50** (**required**)
* Has **Seminars** – a collection of type **Seminar**

### SeminarParticipant

* Has SeminarId – integer, PrimaryKey, foreign key (**required**)
* Has Seminar – Seminar
* Has ParticipantId – string, PrimaryKey, foreign key (**required**)
* Has Participant – IdentityUser

Implement the entities with the **correct datatypes** and their **relations**.

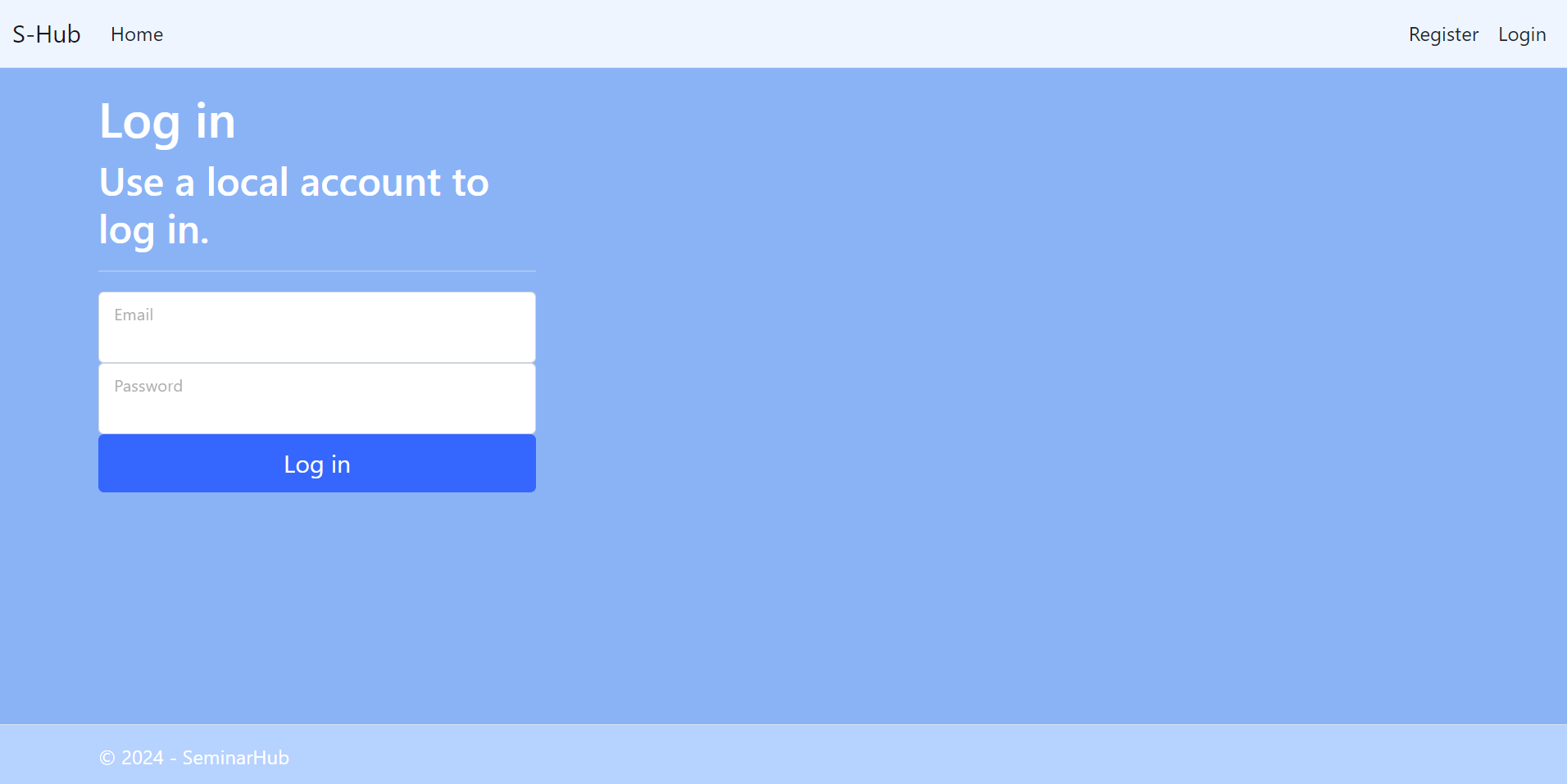
**Feel free to use the new syntax for realization of the many-to-many relation without a mapping table.**

## Page Requirements

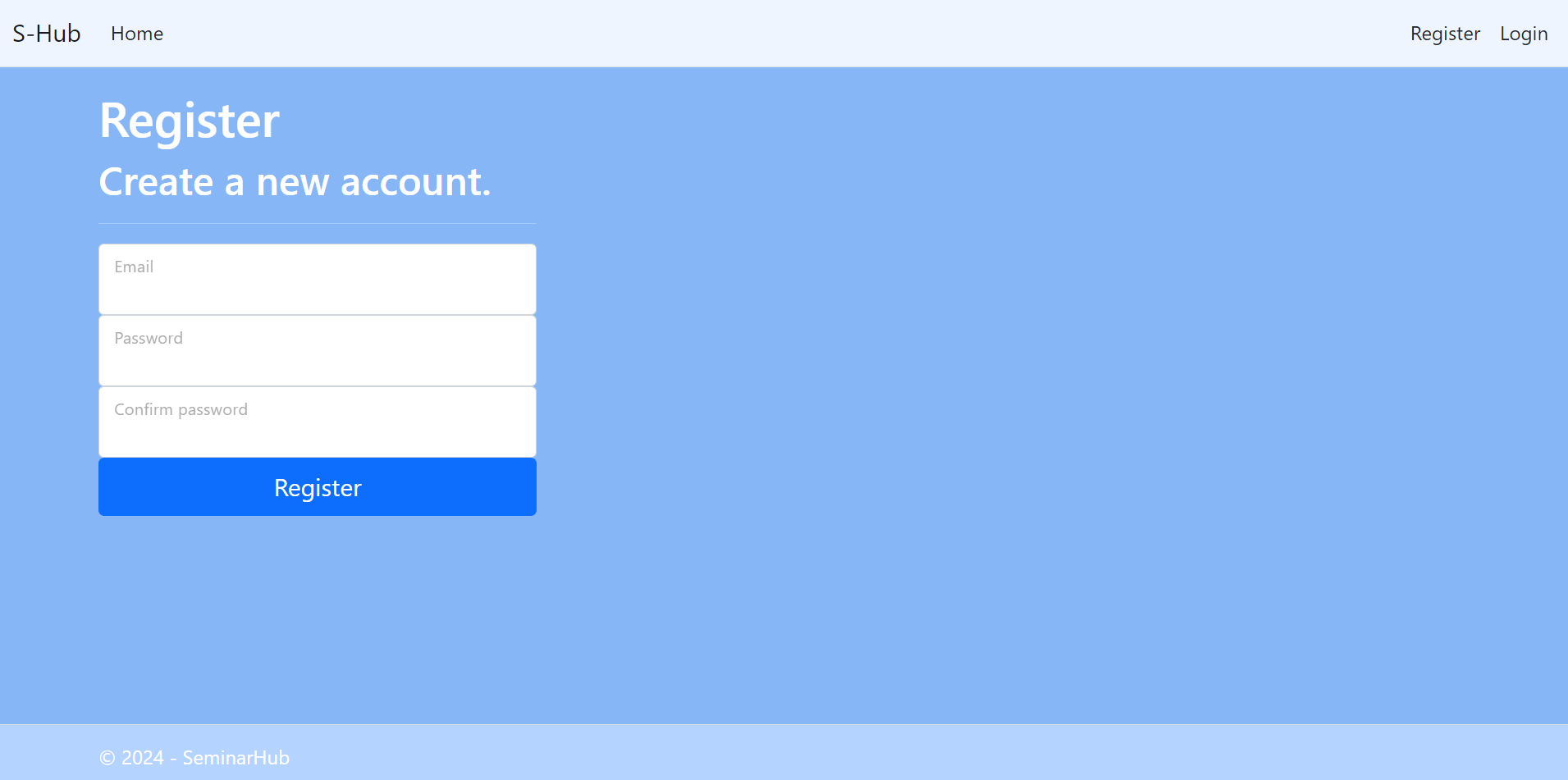
### Index Page (logged-out user)



### Login Page (logged-out user)



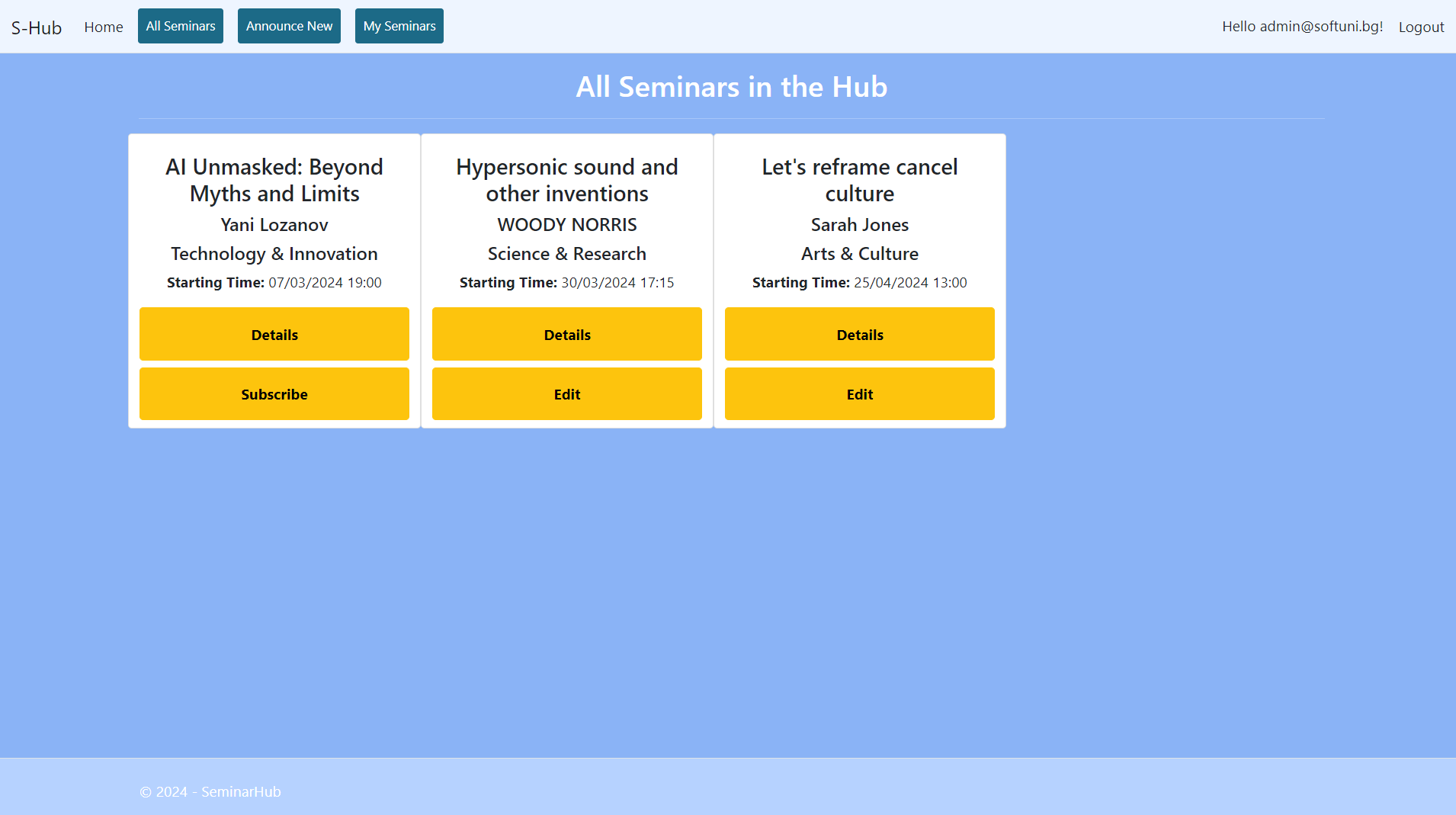
### Register Page (logged-out user)



### /Seminar/Add (logged-in user)

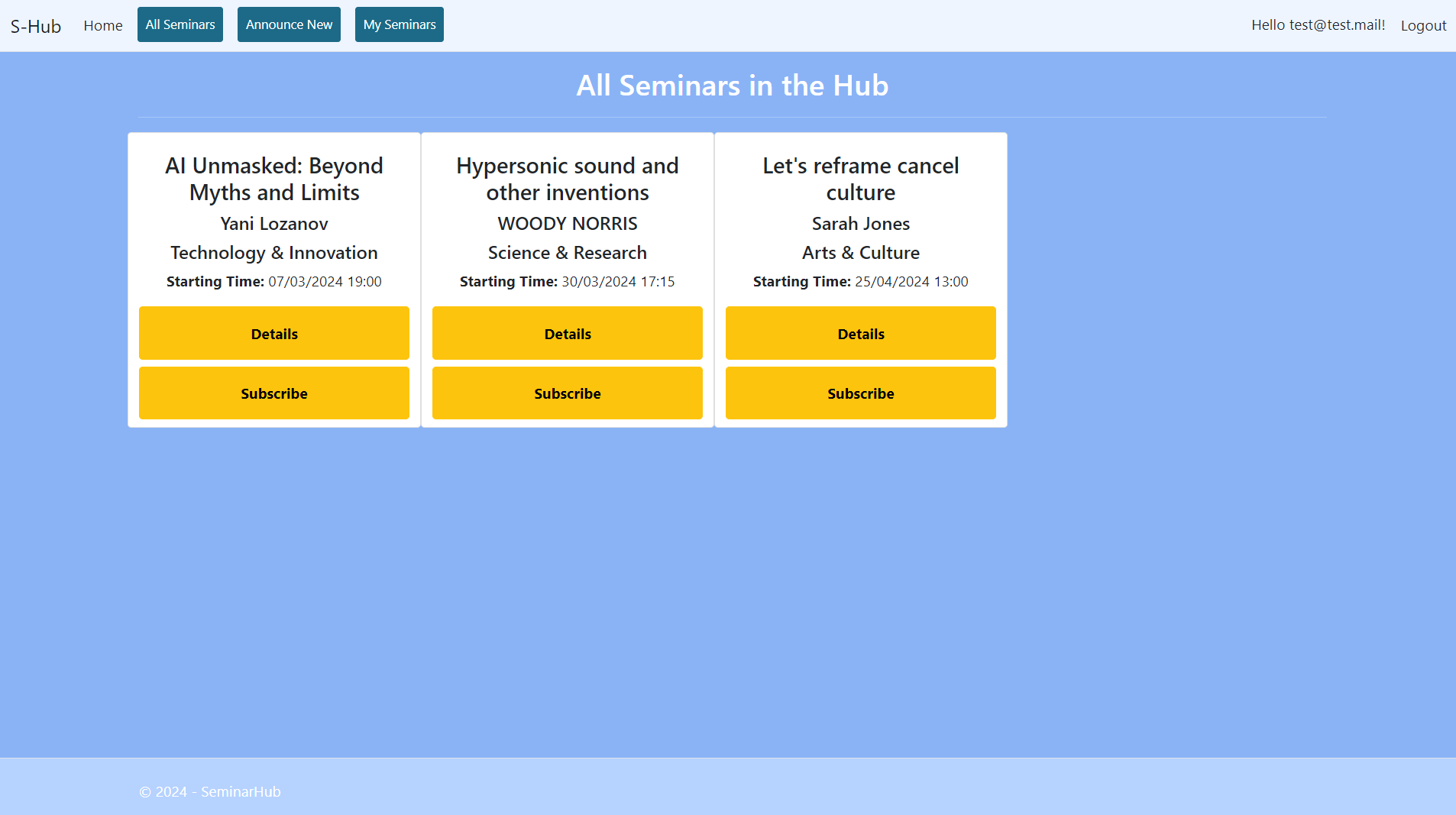
**NOTE:** the **DateTime** format is recommended, if you are having troubles with the one in the image, you are **free** to use **another** one.

### /Seminar/All (logged-in user, creator of a seminar)



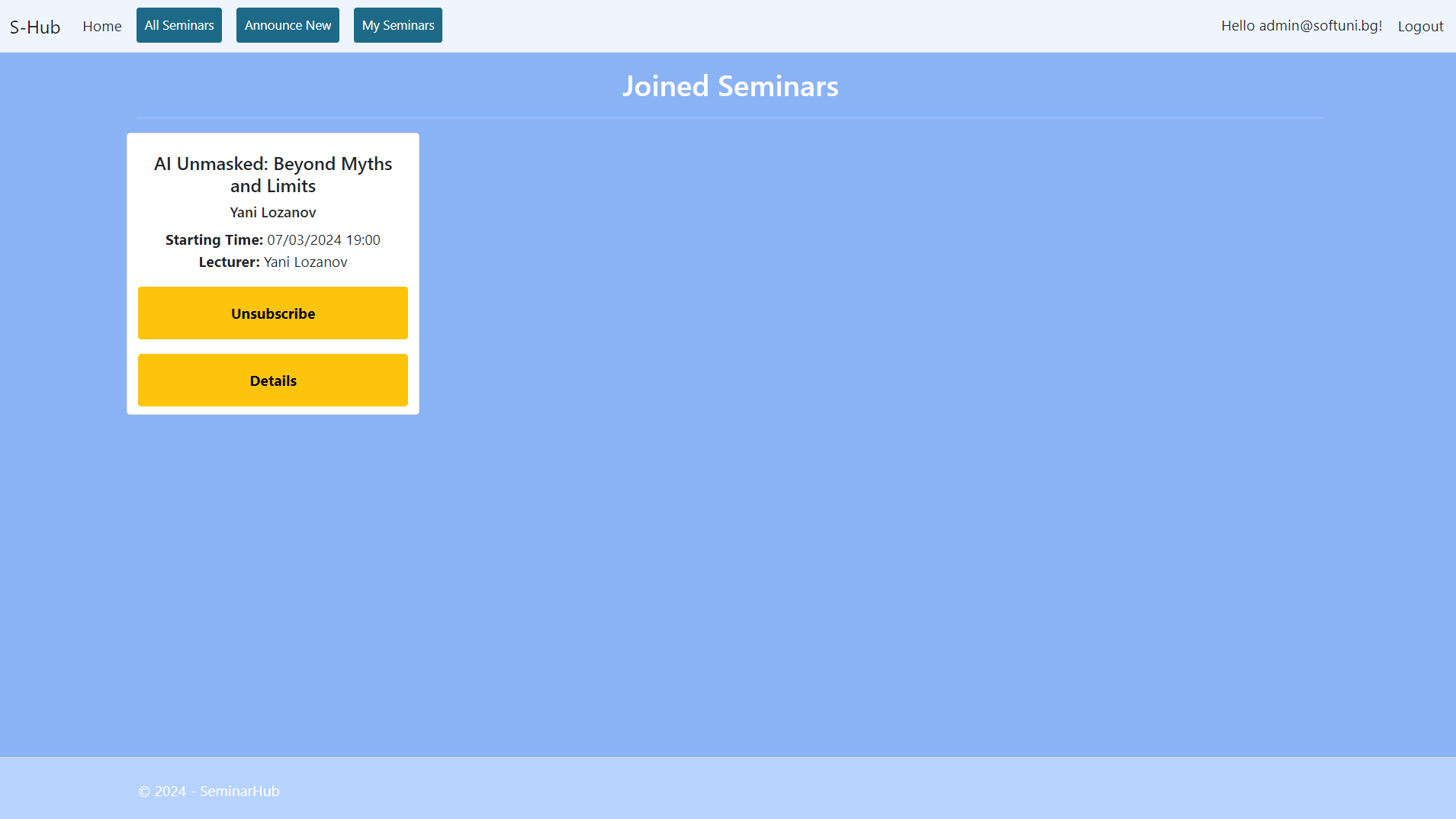
**NOTE**: If the user is **logged in** and tries to go to the **Home page**, the application must redirect them to the **/Seminar/All**.

### /Seminar/All (logged-in user, not creator of a seminar)



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

### /Seminar/Joined (logged-in user)



**NOTE:** the **DateTime** format is recommended, if you are having troubles with the one in the image, you are **free** to use **another** one.

### /Seminar/Edit/{id} (logged-in user)

**NOTE:** the **DateTime** format is recommended, if you are having troubles with the one in the image, you are **free** to use **another** one.

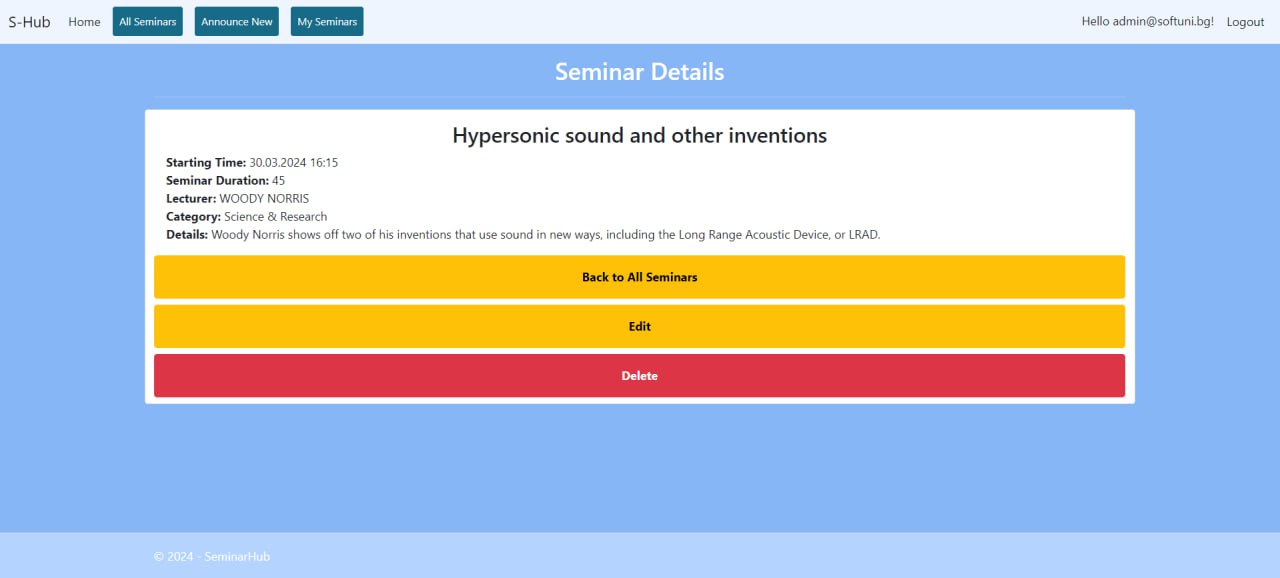
### /Seminar/Join?id={id} (logged-in user)

Adds the selected seminar to the user's collection of seminars. If the seminar is already in their collection, it shouldn't be added. If everything is successful, the user must be redirected to their collection "**/Seminar/Joined**" page.

### /Seminar/Leave?id={id} (logged-in user)

Removes the selected seminar from the user's collection of seminars. If everything is successful, the user must be redirected to "**/Seminar/Joined**" page.

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

**Seminar/Details/{id} (logged-in user, creator of a seminar)  
**

**Note:** If the **currently logged-in user is the creator of the seminar**, the "Delete" and "Edit" buttons should be visible, allowing them full access to the seminar functionalities.

However, if the **logged-in user is** **not the creator** of the seminar, the "Delete" and "Edit" buttons should not be visible. This ensures that only the seminar's creator can access its functionalities.

### Seminar/Details/{id} (logged-in user, not creator of a seminar)

### Seminar/Delete/{id} (logged-in user, creator of a seminar)

## Functionality

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

### Users

Guests can Register, Login and view the Index Page.

Users can **add seminars**, **edit and delete seminars** **only they have added**, **subscribe** for **seminars they haven’t added** and **view the details of all seminars**.

Users can see **added** **seminars** by all users on the Home Page (/Seminar/All).

If the user is the creator of the seminar, they can see the **[Edit]** and **[Delete]** buttons. If the user is not the creator of the seminar, they can join the seminar.

### Seminar

Seminar can be **a**dded by users. All created seminars are visualized on the Home Page (/Seminar/All).

Seminars are visualized on the Home Page (/Seminar/All) with **some** of their information.

Seminars are visualized on the Home Page (/Seminar/All) with two buttons:

* **First button:**
  + **[Details]**
* **Second button:**
  + **If** the user **IS** the **creator** of the seminar – **[Edit]**;
  + **If** the user **IS NOT** the **creator** of the seminar – **[Subscribe]**

The **[Details]** button displays a new page with **all** of the **info** for the **selected** **seminar**.

The **[Edit]** button displays a new page with a form, filled in with **all** of the **info** for the **selected** **seminar**. Users can change this info and save it.

The **[Subscribe]** button adds the seminar to the user's collection of seminars, **unless it is already added**.

Usershave a My Seminars page where only the seminarsin their collection are visualized.

* The **[Unsubscribe]** button **removes** the **seminar** from the **user's** collection of **seminars**.

### Redirections

* Upon successful Login of an **IdentityUser**, you should be redirected to the /Seminar/All.
* Upon successful Creation of a Seminar, you should be redirected to the /Seminar/All.
* Upon successful Adding a **Seminar** to the User's collection, should be redirected to the /Seminar/Joined.
* Upon successful **Editing** of a **Seminar**, you should be redirected to the **/Seminar/All**.
* Upon successful **Deletion** of a **Seminar**, you should be redirected to the **/Seminar/All**.
* Upon successful Removal of a **Seminar** from the User's collection, should be redirected to the /Seminar/Joined.
* If a User tries to **add** an **already added** Seminar to their **collection**, they should be redirected to /Seminar/All.
* Upon **successful deletion** or **abort deletion**, the user should be redirected to Home Page (/Seminar/All).
* 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).

## 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 Seminar/Add page and functionality.
* Users (logged in) can access the Seminar/Details page and functionality.
* Users (logged in) can access the Seminar/Edit page and functionality.
* Users (logged in) can access the Seminar/Delete page and functionality.
* Users (logged in) can access the Seminar/All page.
* Users (logged in) can access the **Joined** page.
* Users (logged in) can access Logout functionality.
* Users (logged in) cannot access the Seminar/Edit page of a Seminar that have another user as a organizer.
* Users (logged in) cannot access the Seminar/Delete page of a Seminar that have another user as a organizer.

## 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.

## 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