# ASP.NET Fundamentals Exam Preparation

# Homies

Problems for the exam for the ["ASP.NET Core Fundamentals"](https://softuni.bg/trainings/4367/asp-net-fundamentals-january-2024) course @ SoftUni

**Homies** is an online platform that the members of a neighbourhood use to announce different events like cleaning an area, important meeting, fun activities, etc.

## Technological Requirements and Overview

* Use the provided skeleton – **Homies\_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 **Type** and **Event** 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 and leave only the needed code in order for the app to be functioning correctly.

**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 **Homies**:

### Event

* Has Id – a unique **integer, Primary Key**
* Has Name – a string with min length **5** and max length **20** (**required**)
* Has Description – a string with min length **15** and max length **150** (**required**)
* Has OrganiserId – an string (**required**)
* Has Organiser – an IdentityUser (**required**)
* Has CreatedOn – a DateTime with format "**yyyy-MM-dd H:mm**" (**required**) (the **DateTime** format is recommended, if you are having troubles with this one, you are free to use another one)
* Has Start – a DateTime with format "**yyyy-MM-dd H:mm**" (**required**) (the **DateTime** format is recommended, if you are having troubles with this one, you are free to use another one)
* Has End – a DateTime with format "**yyyy-MM-dd H:mm**" (**required**) (the **DateTime** format is recommended, if you are having troubles with this one, you are free to use another one)
* Has **TypeId** – an **integer, foreign key (required)**
* Has Type – a Type (**required**)
* Has **EventsParticipants** – a collection of type **EventParticipant**

### Type

* Has Id – a unique **integer, Primary Key**
* Has Name – a string with min length **5** and max length **15** (**required**)
* Has **Events** – a collection of type **Event**

### EventParticipant

* HelperId– a string, Primary Key, foreign key (required)
* Helper– **IdentityUser**
* EventId– an integer, Primary Key, foreign key (required)
* Event – Event

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

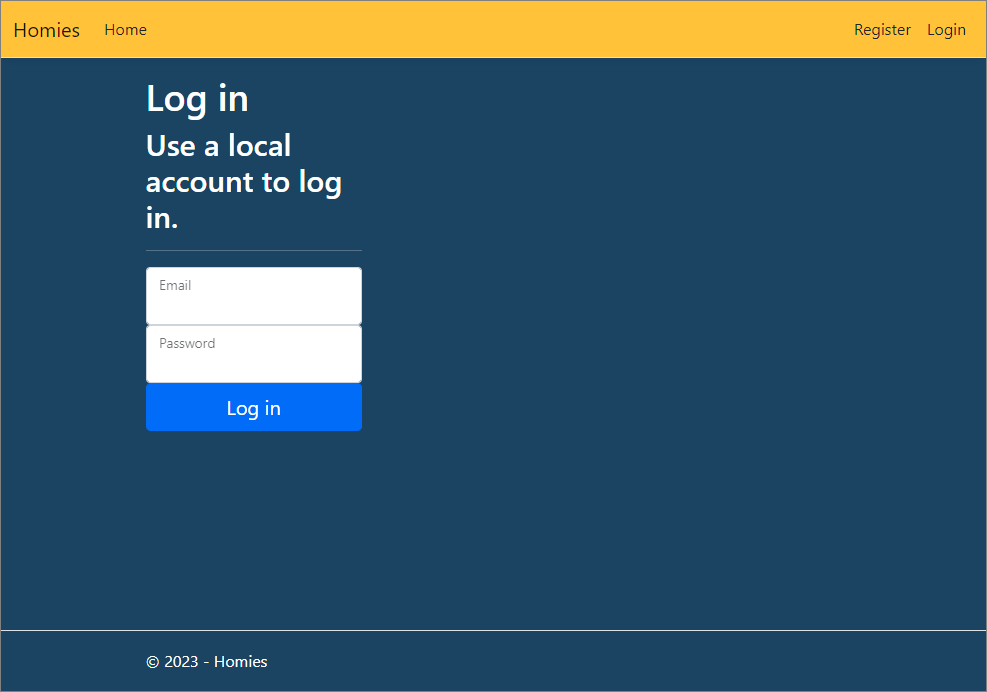
**Feel free to use the new syntax for realization of the many-to-many relation without a mapping table.**

## Page Requirements

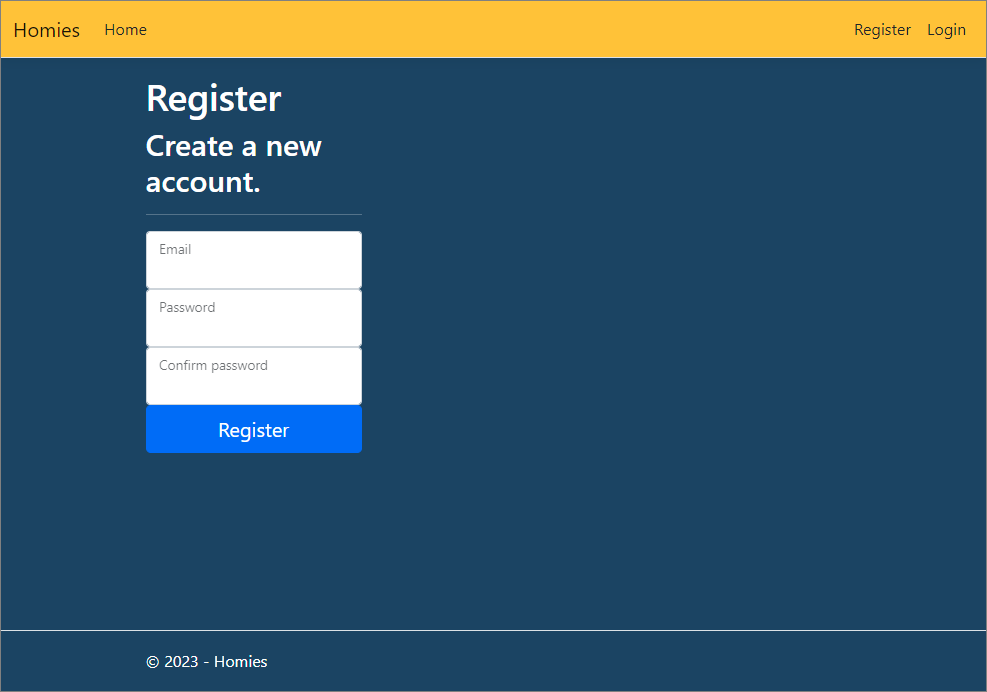
### Index Page (logged-out user)



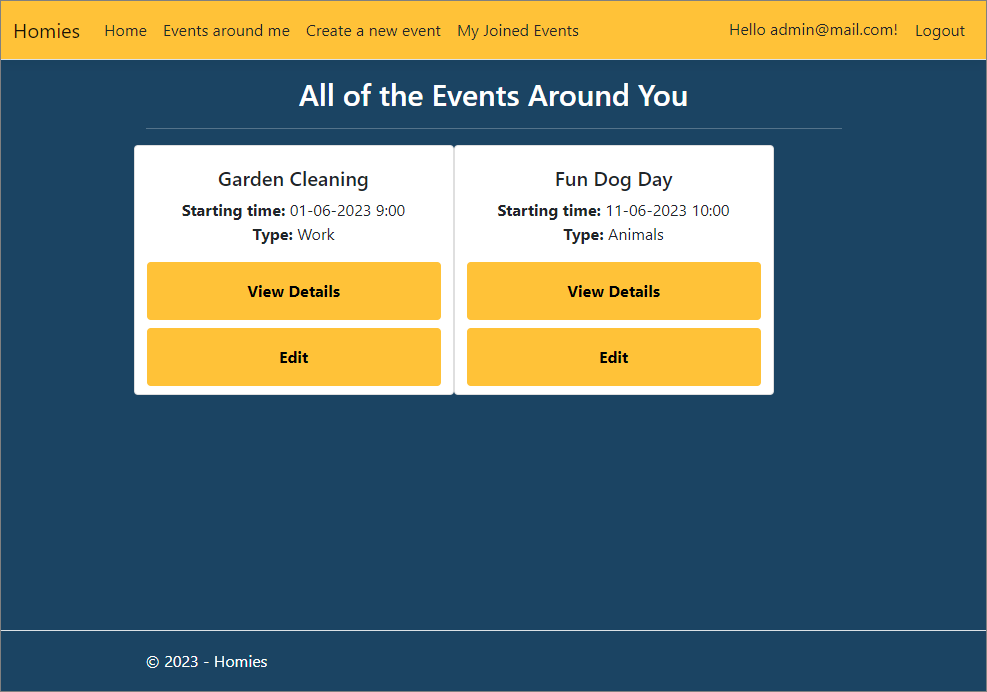
### Login Page (logged-out user)



### Register Page (logged-out user)

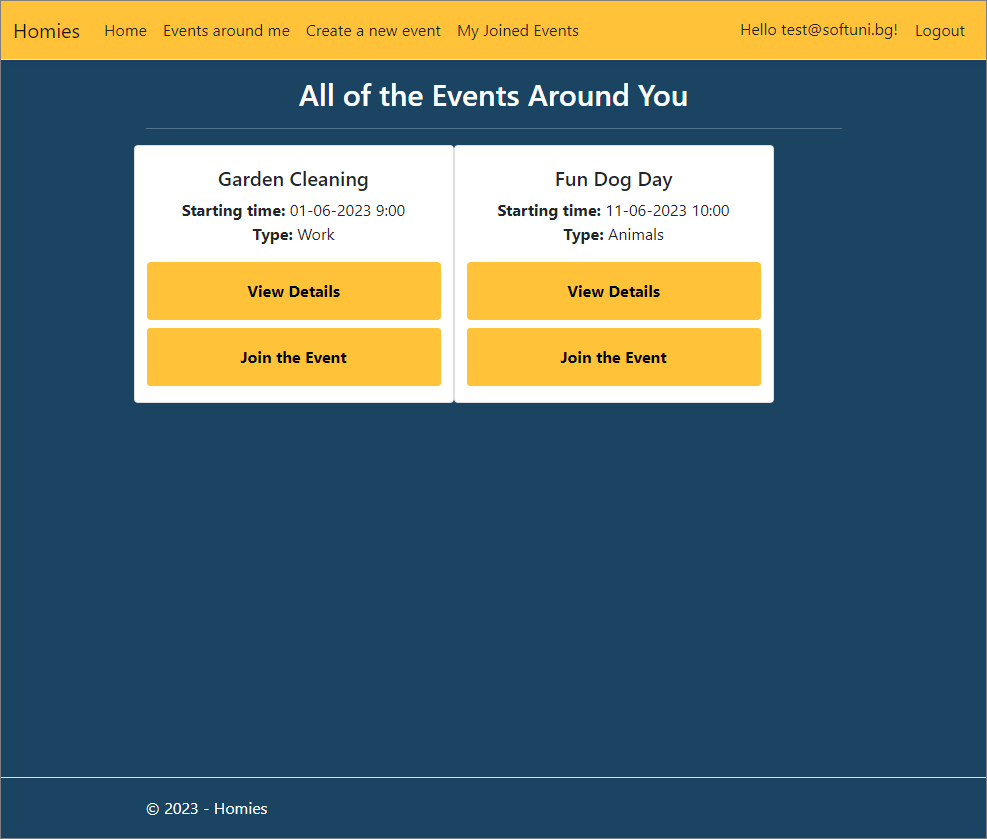


### /Event/All (logged-in user, creator of an event)



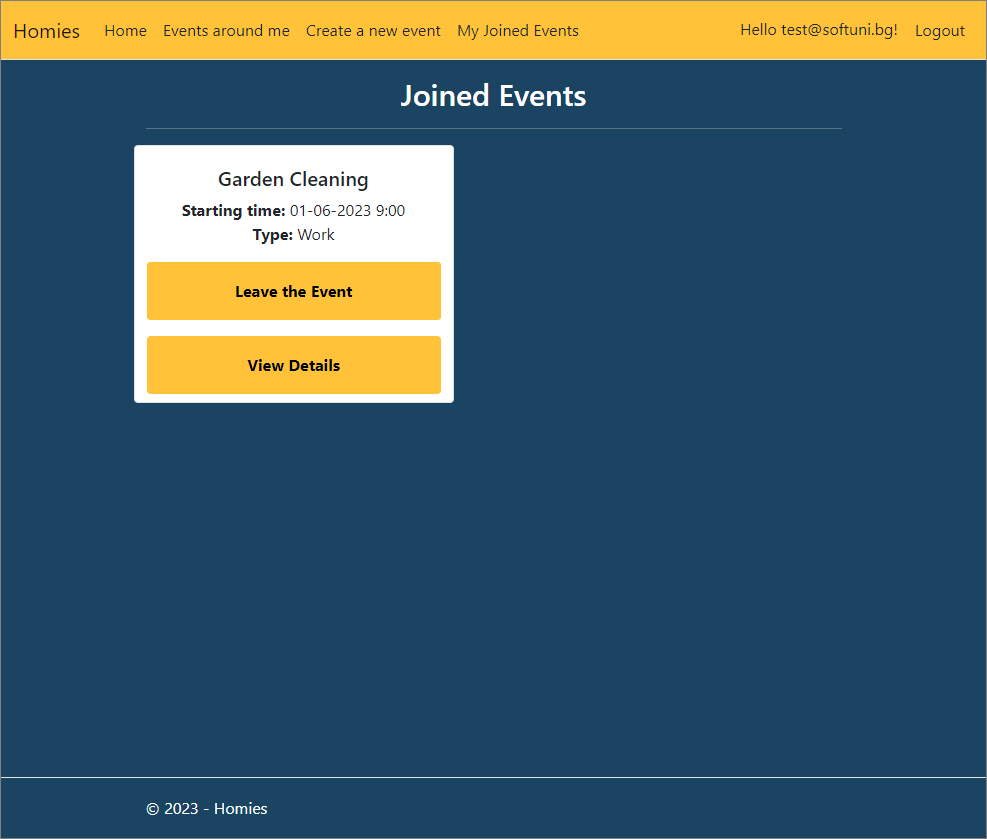
**NOTE**: If the user is **logged in** and tries to go to the **Home page**, the application must redirect them to the **/Event/All**.

### /Event/All (logged-in user, not creator of an event)



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

### /Event/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.

### /Event/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.

**NOTE: The little calendar icons in the example views are NOT required.**

### /Event/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.

**NOTE: The little calendar icons in the example views are NOT required.**

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

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

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

Removes the selected event from the user's collection of events. If everything is successful, the user must be redirected to home "**/Event/All**" page.

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

### \* BONUS: /Event/Details/{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.

## Functionality

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

### Users

Guests can Register, Login and view the Index Page.

Users can **add events**, **edit events** **only they have added** and **view the details of all events**.

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

If the user is the creator of the event, they can see the **[Edit]** button. If the user is not the creator of the event, they can join the event.

### Event

Events can be **a**dded by users. All created events are visualized on the Home Page (/Event/All).

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

Events are visualized on the Home Page (/Event/All) with two buttons:

* **First button:**
  + **[View Details]**
* **Second button:**
  + If the user **IS** the **creator** of the event – **[Edit]**;
  + If the user **IS NOT** the **creator** of the element – **[Join the Event]**

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

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

The **[Join the Event]** button adds the event to the user's collection of events, **unless it is already added**.

Usershave a My Joined Events page where only the eventsin their collection are visualized.

* The **[Leave]** button **removes** the **event** from the **user's** collection of **events**.

### Redirections

* Upon successful Login of an **IdentityUser**, you should be redirected to the /Event/All.
* Upon successful Creation of an Event, you should be redirected to the /Event/All.
* Upon successful Adding an **Event** to the User's collection, should be redirected to the /Event/Joined.
* Upon successful **Editing** of an **Event**, you should be redirected to the **/Event/All**.
* Upon successful Removal of an **Event** from the User's collection, should be redirected to the /Event/All.
* If a User tries to **add** an **already added** Event to their **collection**, they should be redirected to /Event/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).

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

## 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

### Bonus – 5 points