# ASP.NET Fundamentals Exam – 21 December 2022

# SoftUni Teams

Exam problems for the ["ASP.NET Core Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3853/asp-net-fundamentals-september-2022). Submit your solutions in the [**SoftUni Learning** **System**](https://softuni.bg/trainings/3853/asp-net-fundamentals-september-2022#lesson-44941) (delete all "**bin**"/"**obj**" folders).

**Contacts** is an online platform that is used to create and add contacts to your team.

## Technological Requirements and Overview

* Use the provided skeleton – **Contacts\_Slekenton\_6.0** 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/Account** – You are free to choose whether you'd like to use scaffolded identity or not
* **Controllers** – you should implement the controllers here
* **Data** – you should hold the entities 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 **Connection string**
* **Program.cs** – don't forget to set the **DefaultIdentity** options here

**NOTE:** You should seed the database with provided in advance data regarding the **Contact** 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.

## Database Requirements

The **Database** of **Contacts**:

### ApplicationUser

* Has an Id – a **string, Primary Key**
* Has a UserName – a string with **min length** **5** and **max length 20** (**required**)
* Has an Email – a string with **min length** **10** and **max length 60** (**required**)
* Has a Password – a string with **min length** **5** and **max length 20 (before hashed)** – no max length required for a hashed password in the database (**required**)
* Has **ApplicationUsersContacts** – a collection of type **ApplicationUserContact**

### Contact

* Has Id – a unique **integer, Primary Key**
* Has FirstName – a string with min length **2** and max length **50** (**required**)
* Has LastName – a string with min length **5** and max length **50** (**required**)
* Has Email – a string with min length **10** and max length **60** (**required**)
* Has PhoneNumber – a string with min length 10 and max length 13 (**required**). The phone number must start with "**+359**" or '**0**' (zero), followed by **four sets** of digits, **separated** by a **space**, '**-**' (dash) or nothing between the sets. The **first group** must have **exactly three digits** and the **others** exactly **two digits**. Valid examples: **0 875 23 45 15**, **+359-883-15-12-10**, **0889552217**.
* Has **Address** – a **string**
* Has **Website** – a **string**. First four characters are "**www.**"**,** followed by **letters**, **digits** or '**-**' and last three characters are "**.bg**" (**required**)
* Has **ApplicationUsersContacts** – a collection of type **ApplicationUserContact**

### ApplicationUserContact

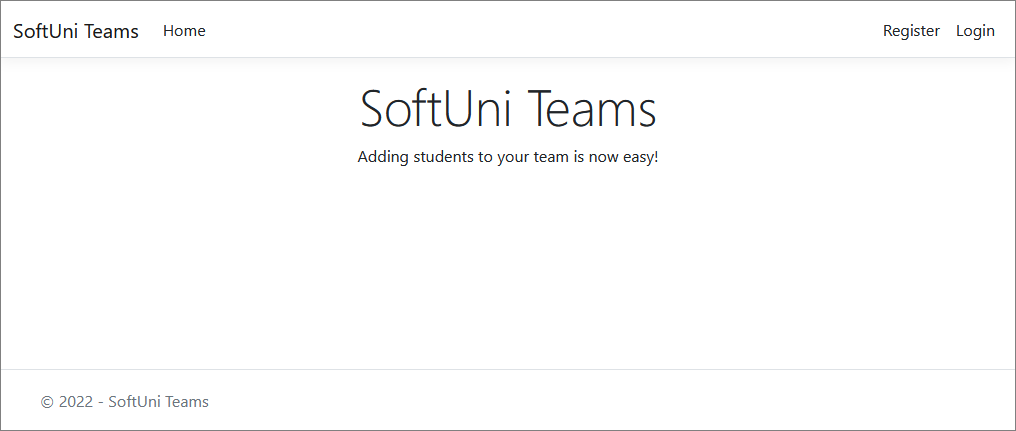
* ApplicationUserId– a string, Primary Key, foreign key (required)
* ApplicationUser– **Application**User
* ContactId– an integer, Primary Key, foreign key (required)
* Contact – Contact

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

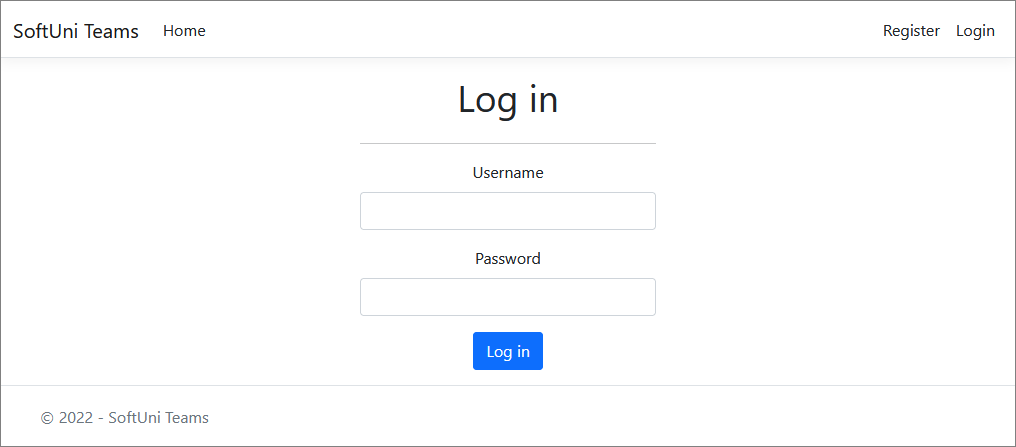
**Feel free to use the new syntax for realization of the many-to-many relation without a mapping table.**

## Page Requirements

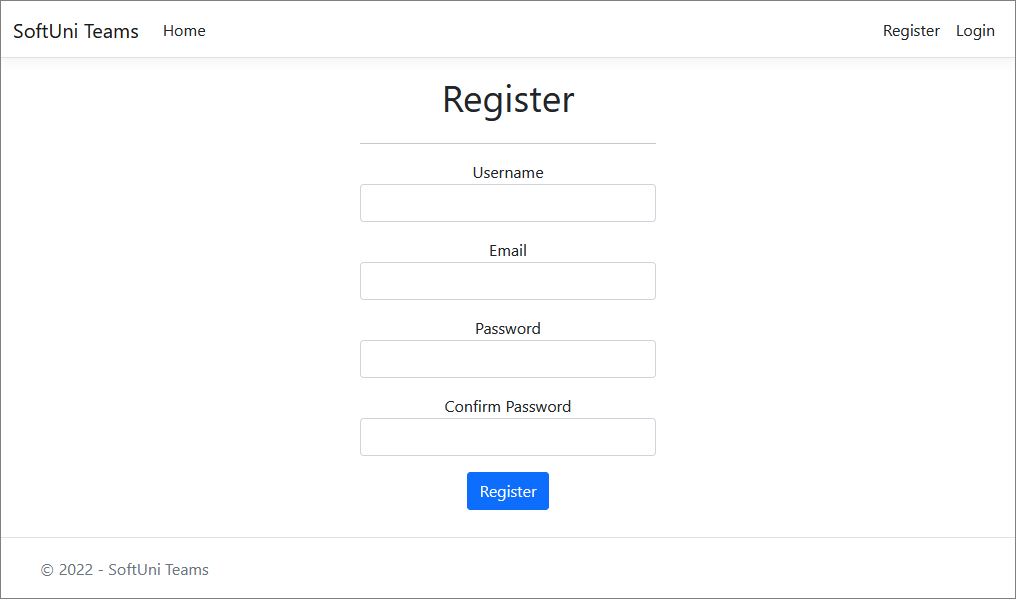
### Index Page (logged-out user)



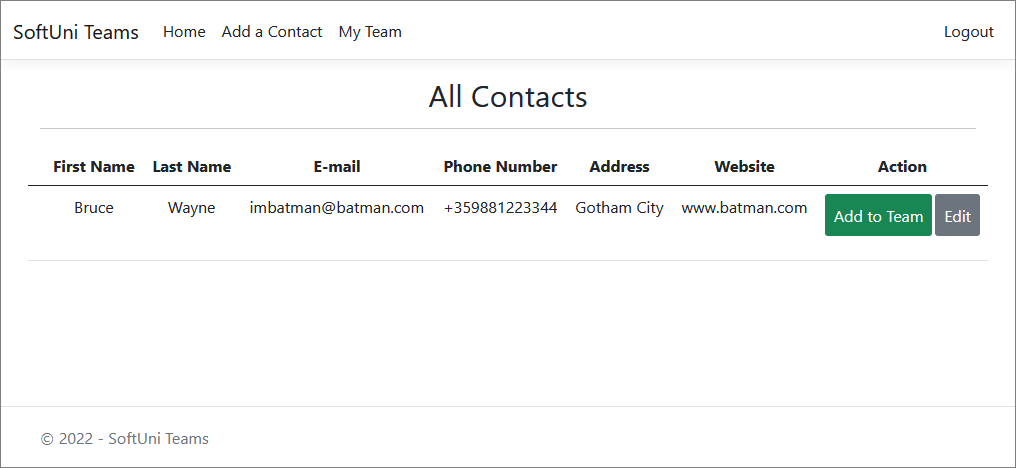
### Login Page (logged-out user)



### Register Page (logged-out user)

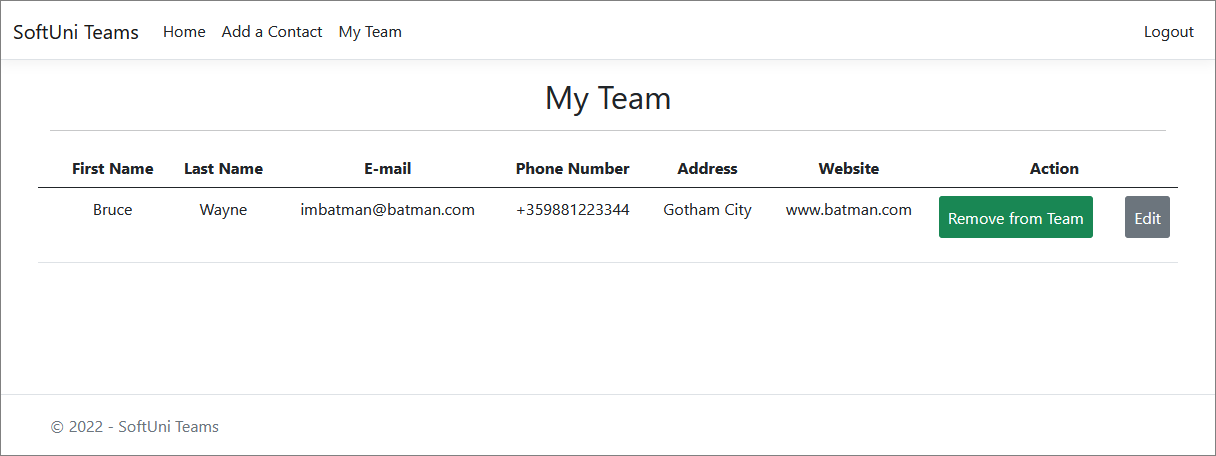


### /Contacts/All (logged-in user)



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

### /Contacts/Team (logged-in user)



**NOTE**: Displays a list of the contacts, added to the logged-in user's collection.

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

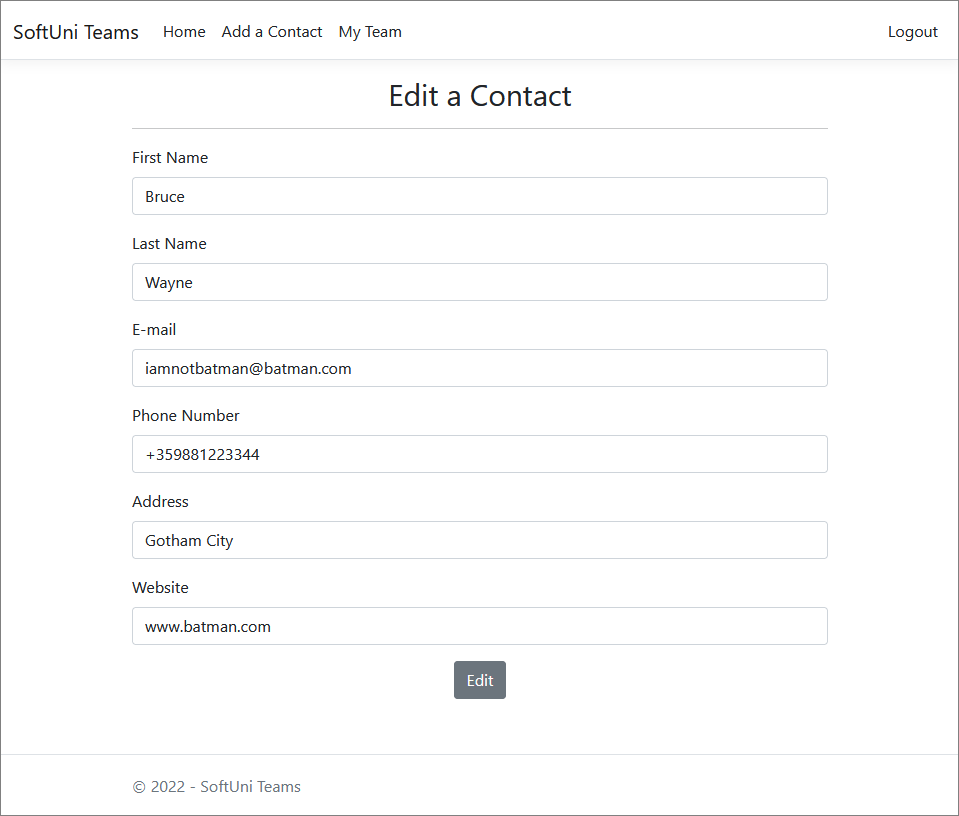
### /Contacts/AddToTeam?contactId={contactId} (logged-in user)

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

### /Contacts/RemoveFromTeam?contactId={contactId} (logged-in user)

Removes the selected contact from the user's collection of contacts. If everything is successful, the user must be redirected to their collection "**/Contacts/Team**" page.

### /Contacts/Edit/{contactId} (logged-in user)



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

## Functionality

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

### Users

Guests can Register, Login and view the Index Page.

Users can AddContacts and see added Contacts by all Users on the Home Page (/Contacts/All). From the Home Page (/Contacts/All),they can also view Info about each one of those Contacts, Add them to their team or edit their **Info**.

### Contacts

Contacts can be Added by Users. All created Contacts are visualized on the Home Page (/Contacts/All), each one on a separate row of a table.

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

Contacts are visualized on the Home Page (/Contacts/All) with two buttons – [**Add to Team**] and [**Edit**].

* The [**Add to Team**] button adds the Contact to the User's **Team**, **unless it is already added**.
* The [**Edit**] button redirects to a page where all of the **Contact**'s info can be edited.

Usershave a My Team page where only the Contactsin their collection are visualized.

* The [**Remove from Team**] button removes the Team from the User's **Team**.

### Redirections

* Upon successful Registration of a User, you should be redirected to the Login Page.
* Upon successful Login of a User, you should be redirected to the /Contacts/All.
* Upon successful Creation of a Contact, you should be redirected to the /Contacts/All.
* Upon successful Adding a Contact to the User's **Team**, should be redirected to the /Contacts/All.
* Upon successful Removal of a Contactfrom the User's **Team**, should be redirected to the /Contacts/Team.
* If a User tries to **add** an **already added** Contact to their **collection**, they should be redirected to /Contacts/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 Contacts/Add page and functionality.
* Users (logged in) can access the Contacts/All page.
* Users (logged in) can access the **My** Team page.
* Users (logged in) can access Logout functionality.

## 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

### Database Requirements – 10 points.

### Template Requirements – 10 points.

### Functionality – 50 points.

### Security – 10 points.

### Code Quality – 10 points.

### Data Validation – 10 points.