# ASP.NET Fundamentals Retake Exam – 17 April 2024

# 

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

**GameZone** is an online platform where the fans of console games will be able to exchange information about different games. Registered users will have their own zone with favourite games.

## Technological Requirements and Overview

* Use the provided skeleton – **GameZone-Skeleton.zip**
* 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 **Genre** 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**](#_Login_Page_(logged-out) **and** [**Register**](#_Register_Page_(logged-out) **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 **GameZone**:

### Game

* Has Id – a unique **integer, Primary Key**
* Has Title – a string with min length **2** and max length **50** (**required**)
* Has Description – string with min length **10** and max length **500** (**required**)
* Has **ImageUrl** – nullable string
* Has PublisherId – string (**required**)
* Has Publisher – IdentityUser (**required**)
* Has ReleasedOn– DateTime with format "**yyyy-MM-dd**" (**required**) (the **DateTime** format is recommended, if you are having troubles with this one, you are **free to use another one**)
* Has GenreId – **integer, foreign key (required)**
* Has Genre – Genre (**required**)
* Has **GamersGames** – a collection of type **GamerGame**

### Genre

* Has Id – a unique **integer, Primary Key**
* Has Name – a string with min length **3** and max length **25** (**required**)
* Has **Games** – a collection of type **Game**

### GamerGame

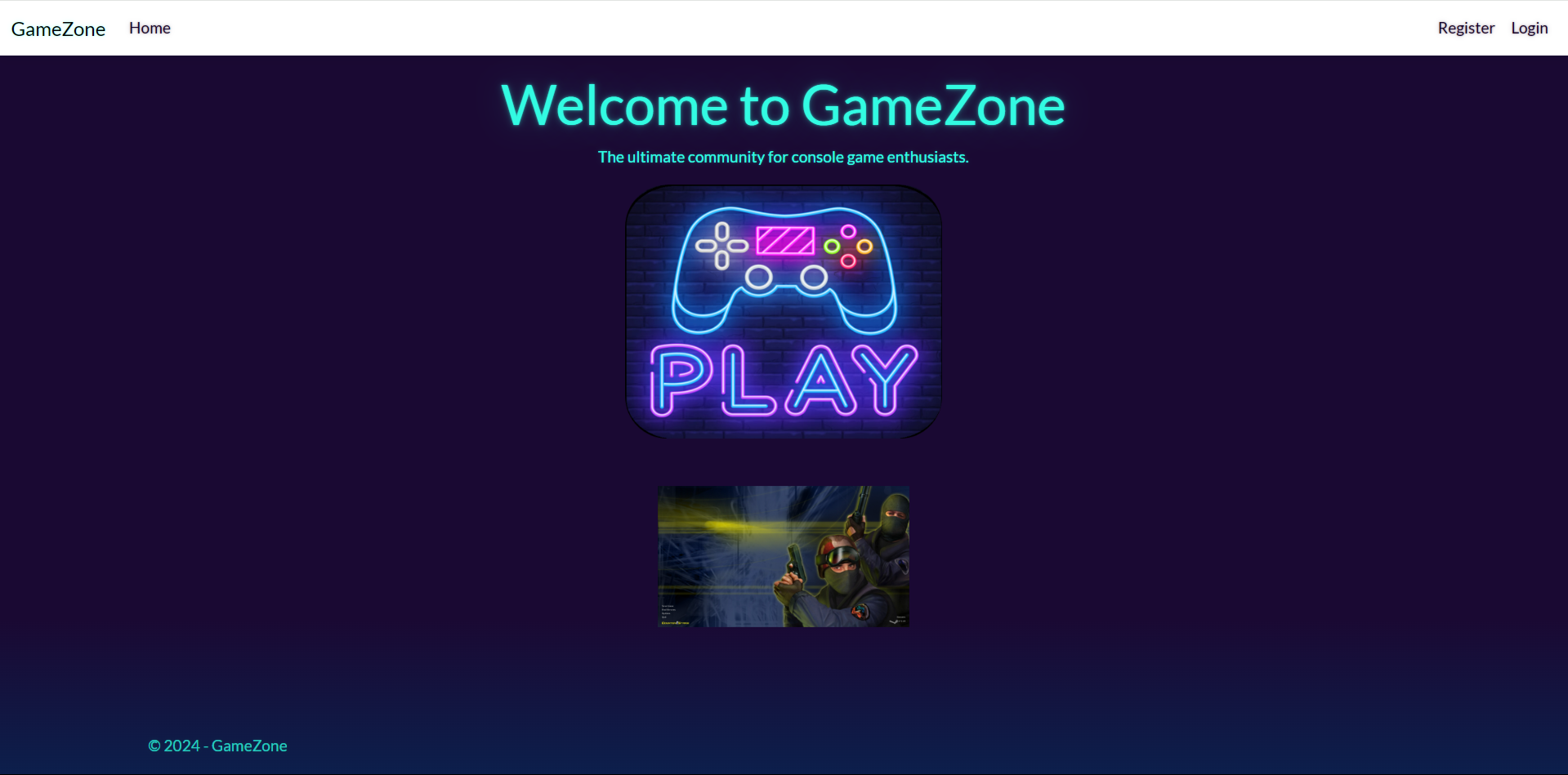
* Has GameId – integer, PrimaryKey, foreign key (**required**)
* Has Game – Game
* Has GamerId – string, PrimaryKey, foreign key (**required**)
* Has Gamer – IdentityUser

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

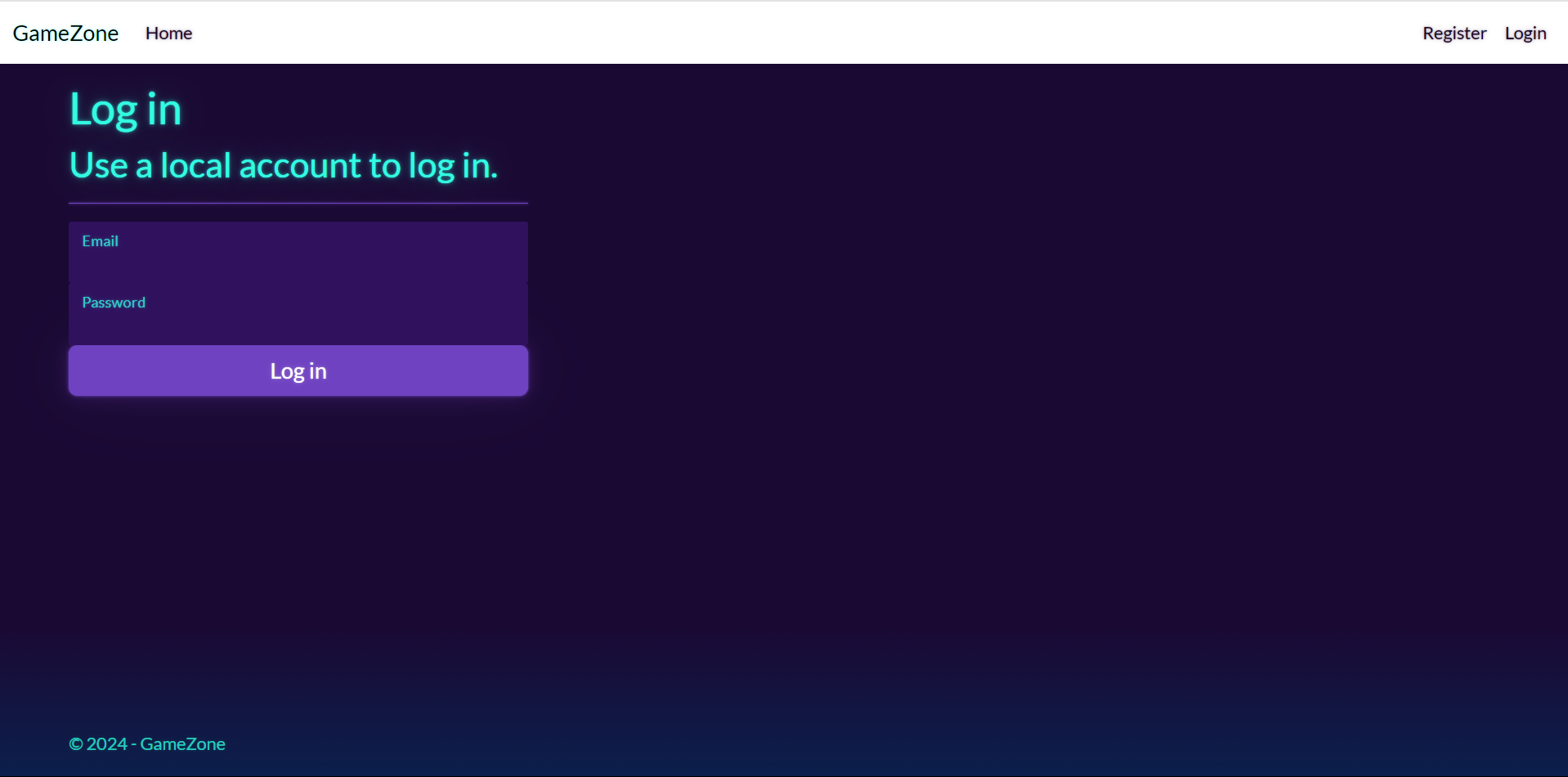
**Feel free to use the new syntax for realization of the many-to-many relation without a mapping table.**

## Page Requirements

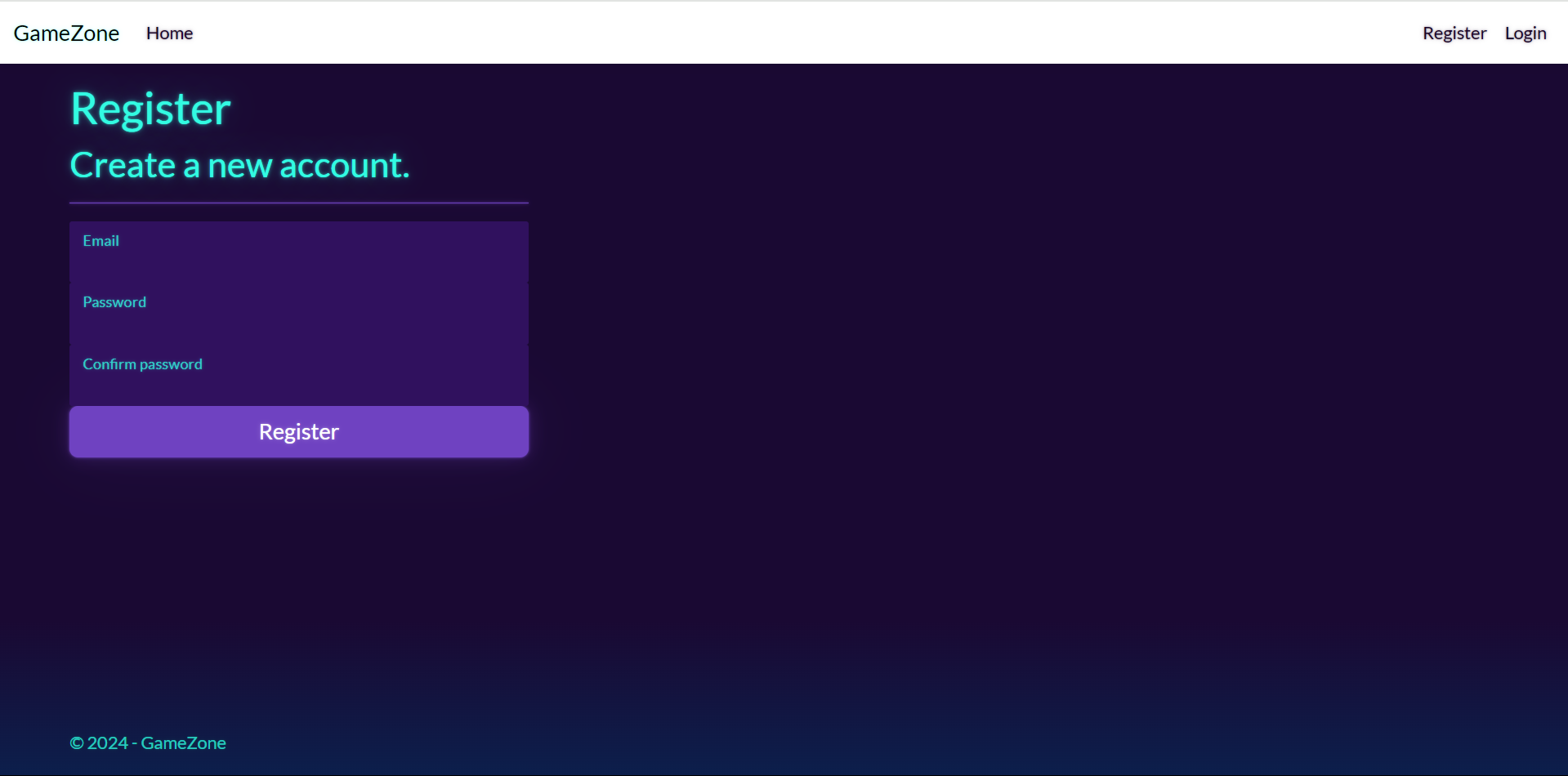
### Index Page (logged-out user)



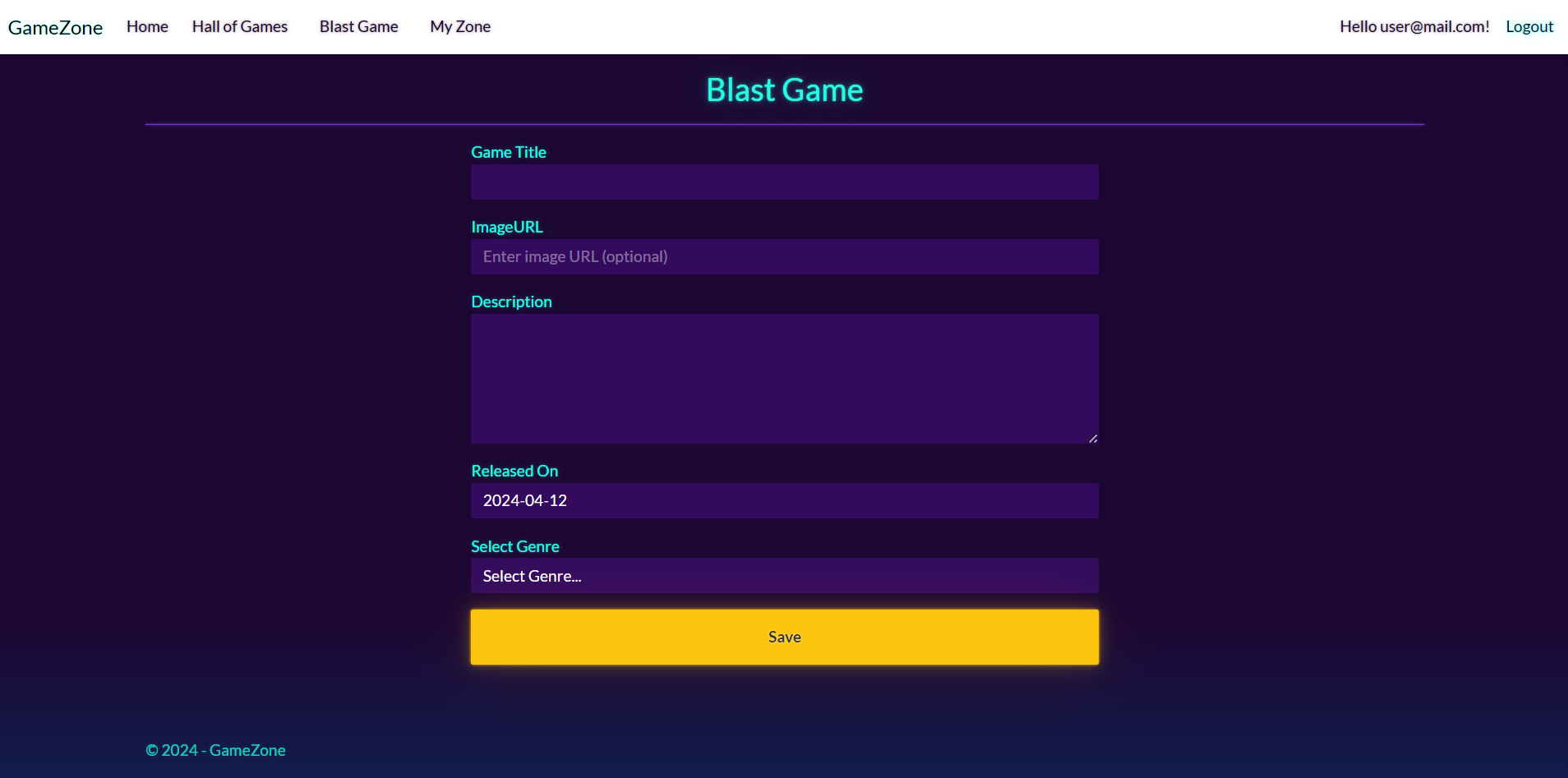
### Login Page (logged-out user)



### Register Page (logged-out user)

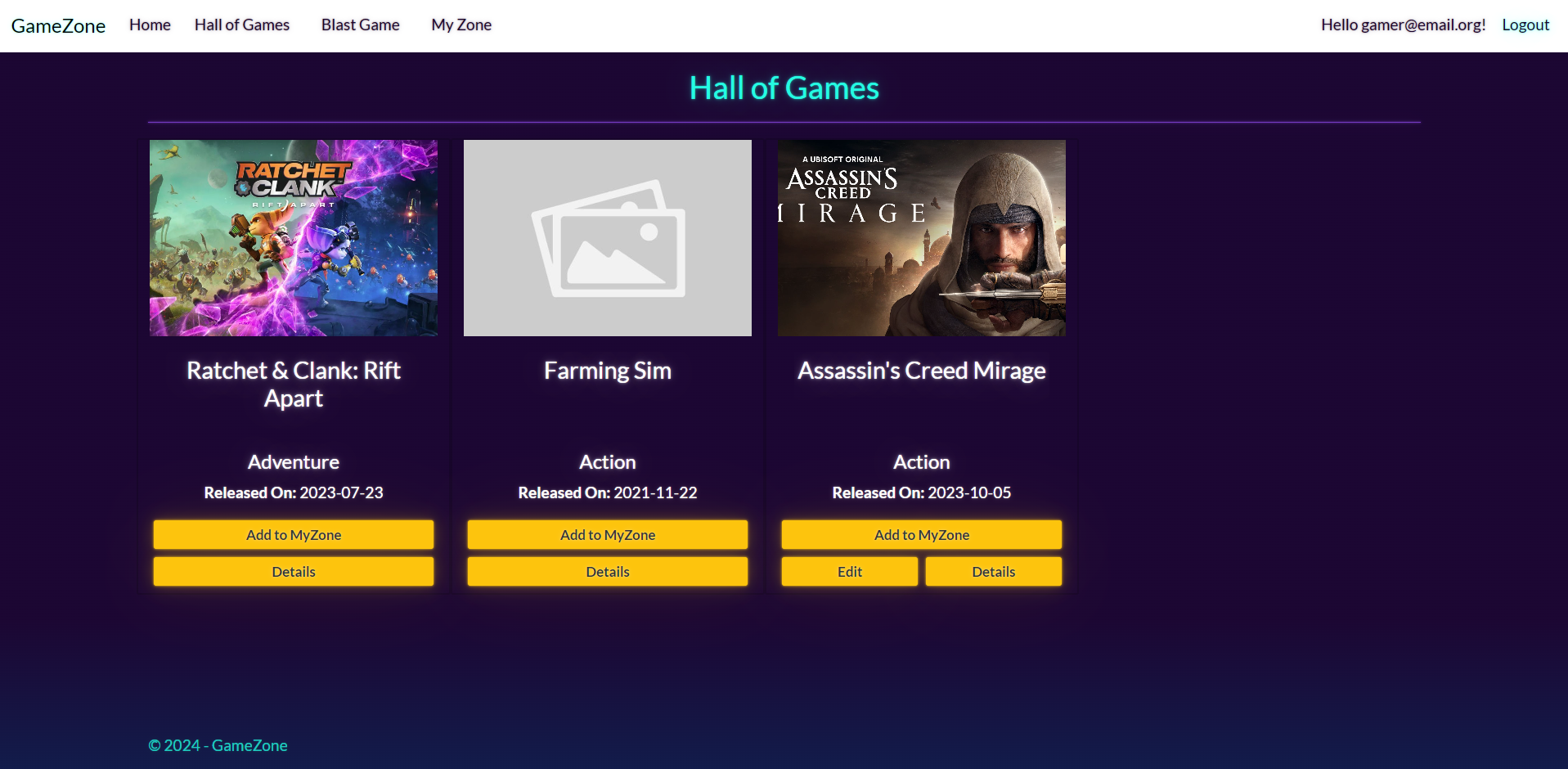


### /Game/Add (logged-in user)



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

### /Game/All (logged-in user, publisher of a specific game)



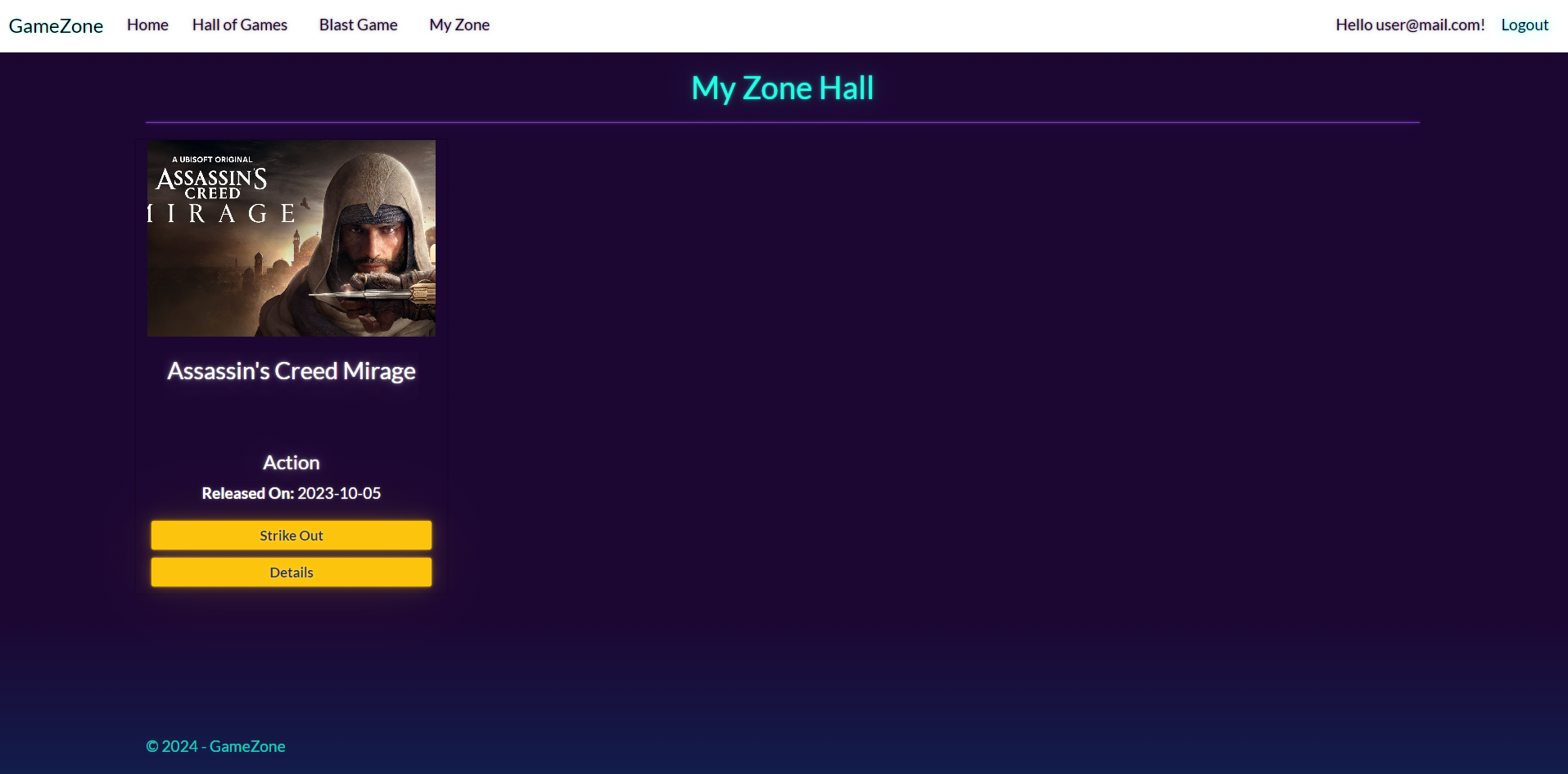
**NOTE**: If the user is **logged in** and tries to go to the **Home page**, the application must redirect them to the **/Game/All**.

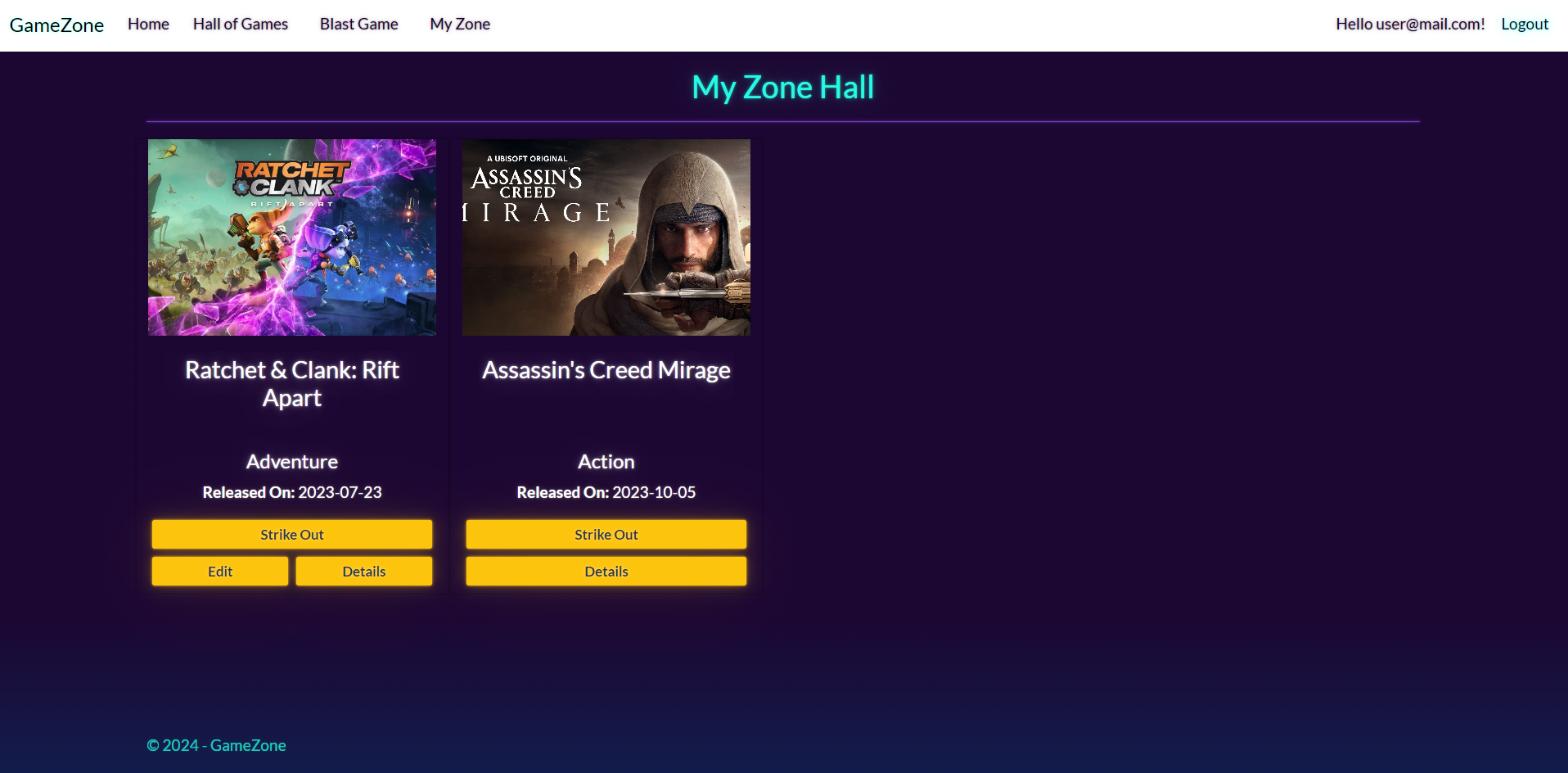
### /Game/All (logged-in user, not publisher of any game)



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

### /Game/MyZone (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.

### /Game/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.

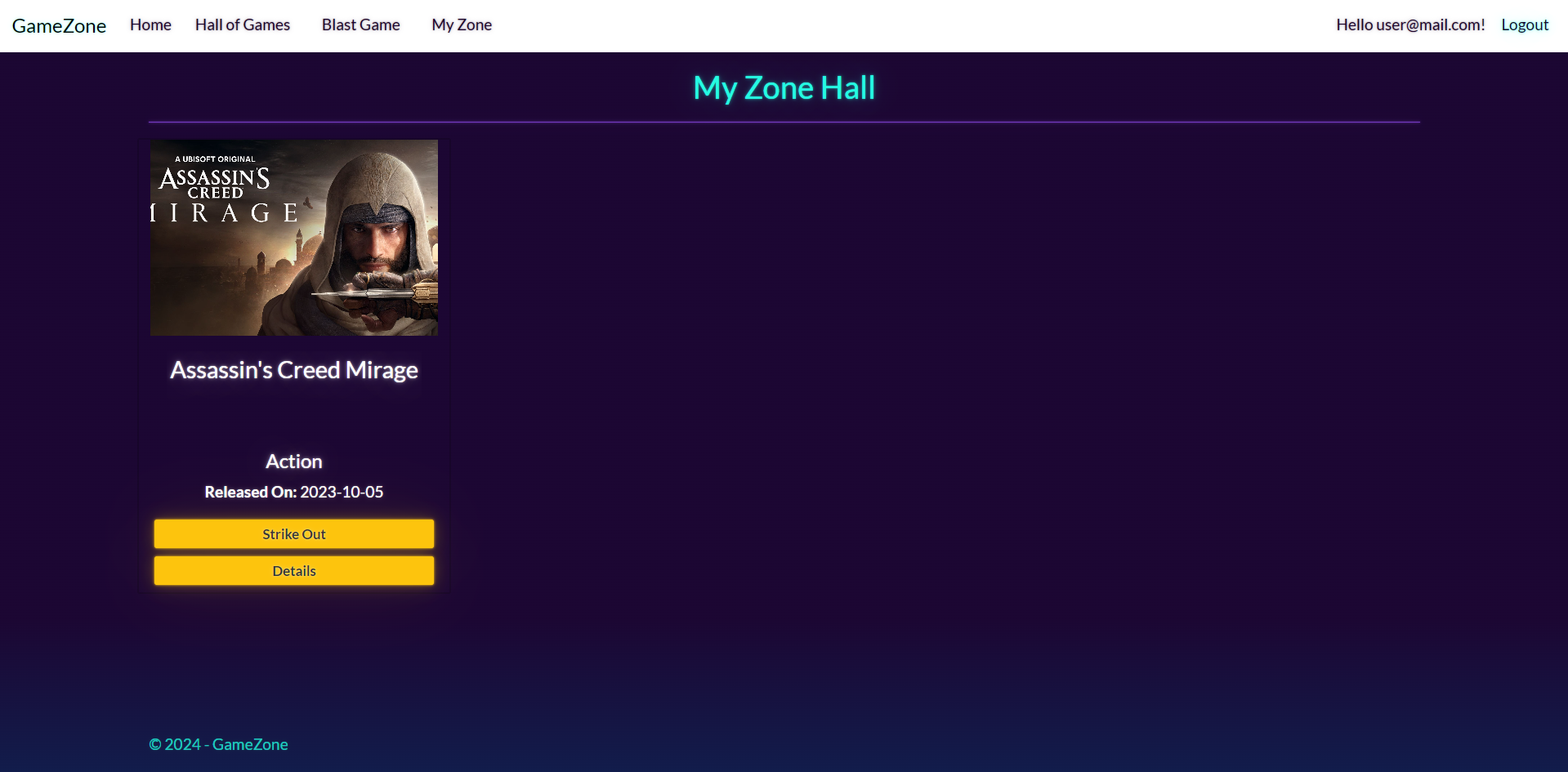
### /Game/Join?id={id} (logged-in user)

Adds the selected game to the user's collection of games. If the game is already in their collection, it shouldn't be added. If everything is successful, the user must be redirected to their collection "**/Game/MyZone**" page.

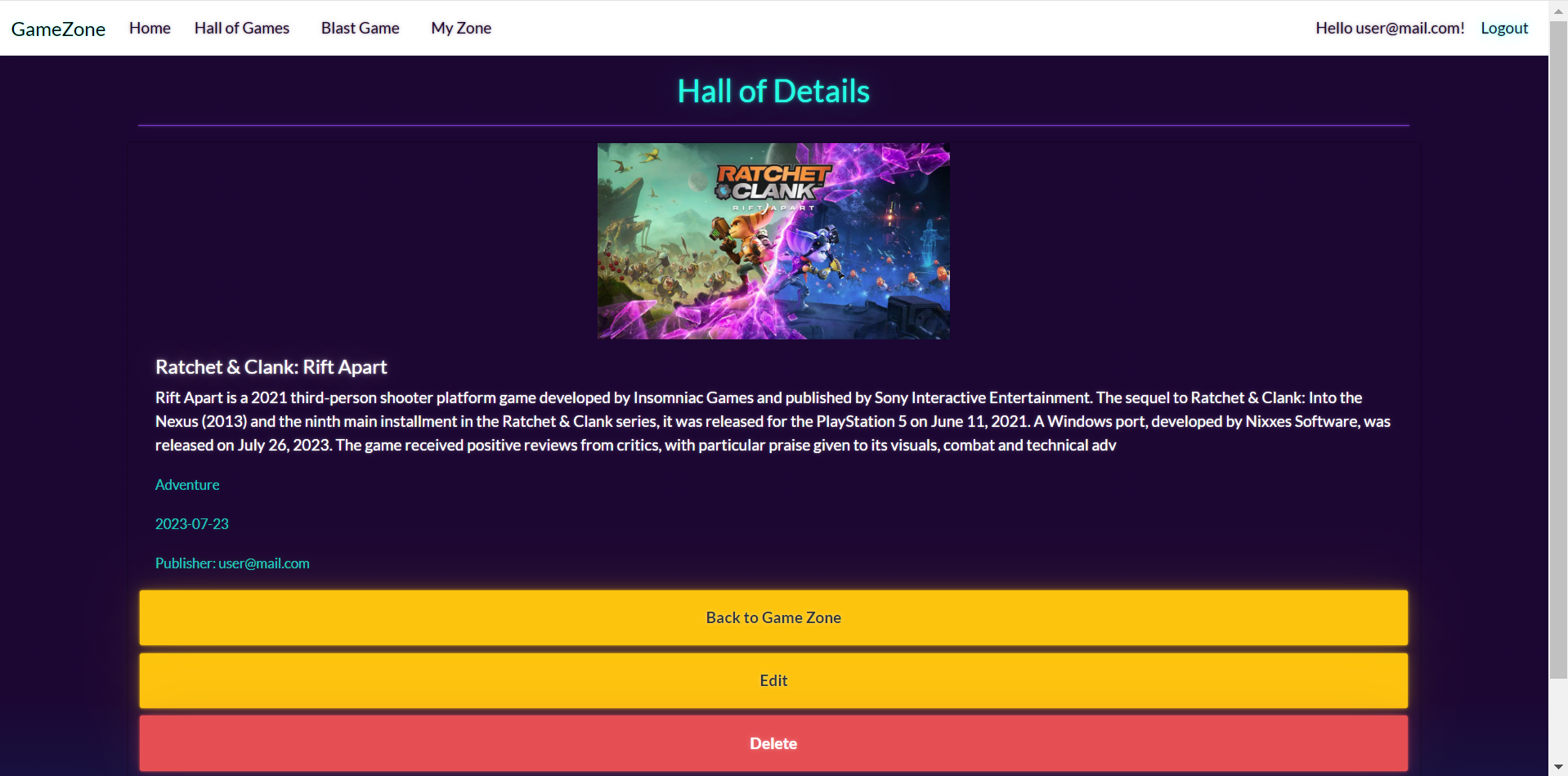
### /Game/StrikeOut?id={id} (logged-in user)

Removes the selected game from the user's collection of games. If everything is successful, the user must be redirected to "**/Game/MyZone**" page.

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



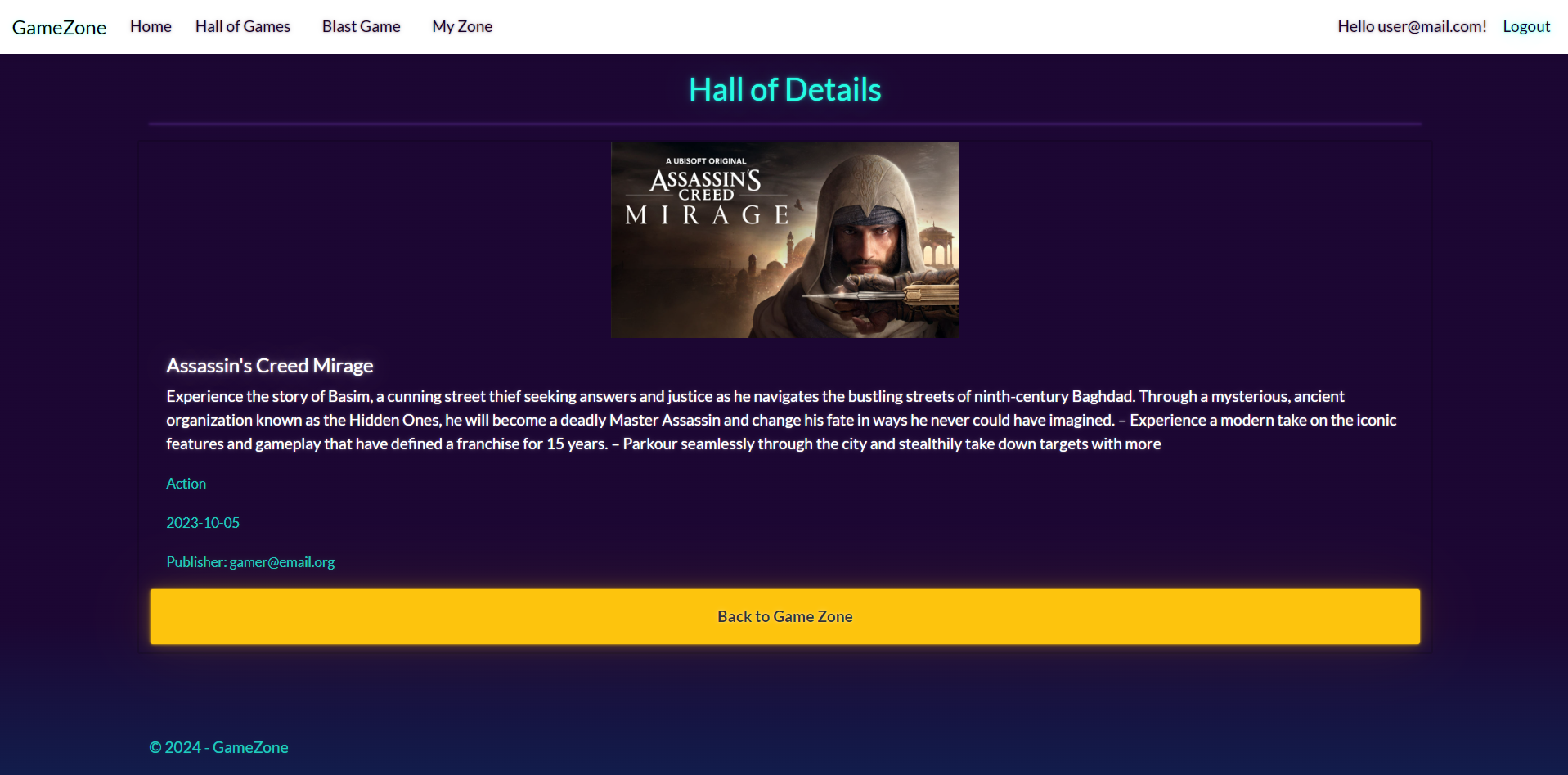
**Game/Details/{id} (logged-in user, publisher of a game)**



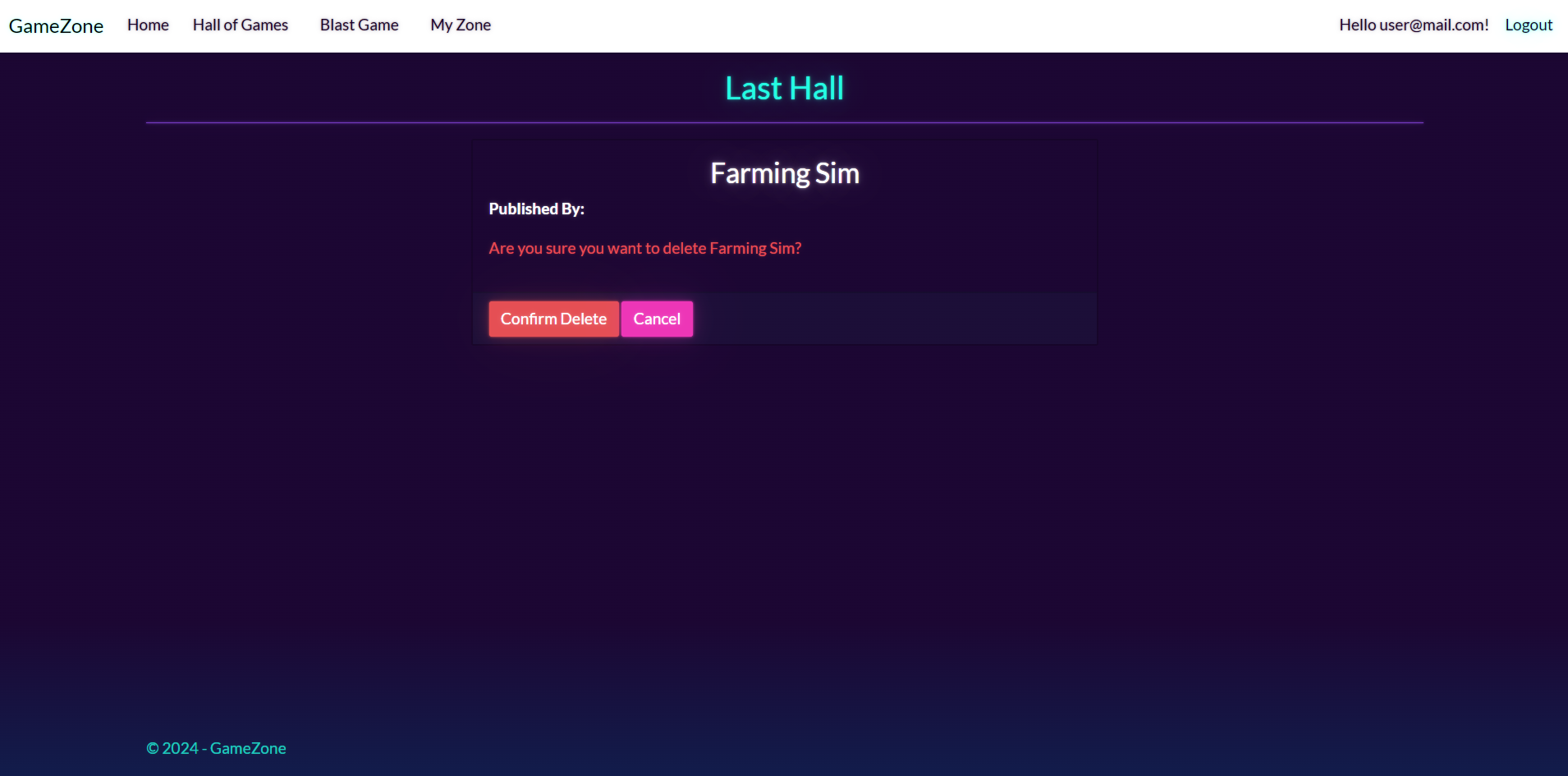
**NOTE**: If the **currently logged-in user is the publisher of the game**, the "Delete" and "Edit" buttons should be visible, allowing them full access to the game functionalities.

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

**Game/Details/{id} (logged-in user, not a publisher of a game)**

****

**Game/Delete/{id} (logged-in user, publisher of a game)**

****

## Functionality

The functionality of the **GameZone** Platform is a very simple application.

### Users

Guests can Register, Login and view the Index Page.

Users can **publish games** and **edit games** **only they have published**.

Users can see **published** **games** by all users on the Home Page (/Game/All).

If the user is the publisher of the game, they can see the **[Edit]** button. If the user is not the publisher of the game, they can add the game to their collection [**Add to MyZone**], , they can also see the **[Details]** button

### Game

Game can be **published** by users. All published games are visualized on the Home Page (/Games/All).

Games are **not automatically added to MyZone collection** by their publishers.

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

Games are visualized on the Home Page (/Game/All) with **two or three buttons**:

* If the user **IS NOT** the **publisher** of the game – **[Add to MyZone][Details]**
* If the user **IS** the **publisher** of the game – **[Add to MyZone][Edit][Details]**

The **[Add to MyZone]** button adds the game to the user's collection of games (MyZone), **unless it is already added**.

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

The **[Details]** button displays a new page with the full information for the selected game.

Usershave a MyZone page where only the gamesin their collection are visualized.

* The **[Strike Out]** button **removes** the **game** from the **user's** collection of **games**.

### Redirections

* Upon successful Login of an **IdentityUser**, you should be redirected to the /Game/All
* Upon successful Publishing (Blast) of a Game, you should be redirected to the /Game/All
* Upon successful Adding a **Game** to the User's collection, should be redirected to the /Game/MyZone
* Upon successful **Editing** of a **Game,** you should be redirected to the **/Game/All**
* Upon successful Removal of a **Game** from the User's collection, should be redirected to the /Game/MyZone
* If a User tries to **add** an **already added** game to their **collection**, they should be redirected to /Game/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 Game/Add page and functionality.
* Users (logged in) can access the Game/Edit page and functionality.
* Users (logged in) can access the Game/All page.
* Users (logged in) can access the **My** **Zone** page.
* Users (logged in) can access Logout functionality.
* Users (logged in) cannot access the Game/Edit page of a Game that have another user as a publisher.

## 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