# C# Web Basics Exam – 28 April 2020

# Battle Cards

Exam problems for the [C# Web Basics course @ SoftUni](https://softuni.bg/courses/csharp-web-basics). Submit your solutions in the **SoftUni judge** system (delete all "**bin**"/"**obj**" folders).

**Battle Cards** is an online platform which is used to create and collect game cards.

## Technological Requirements

* Use the **SIS**
* Use **Entity Framework Core – 3.1**

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

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

## Database Requirements

The **Database** of **Battle Cards**:

### User – DONE!!!

* 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 (**required**)
* Has a Password – a string with **min length** **6** and **max length 20** - hashed in the database (**required**)
* Has **UserCard** collection

### Card – DONE!!!

* Has Id – an **int, Primary Key**
* Has Name – a string (**required**); min. length: 5, max. length: 15
* Has ImageUrl – a string (**required**)
* Has Keyword – a string (**required**)
* Has Attack – an int (**required**); cannot be negative
* Has Health – an int (**required**); cannot be negative
* Has a Description – a string with **max length 200** (**required**)
* Has **UserCard** collection

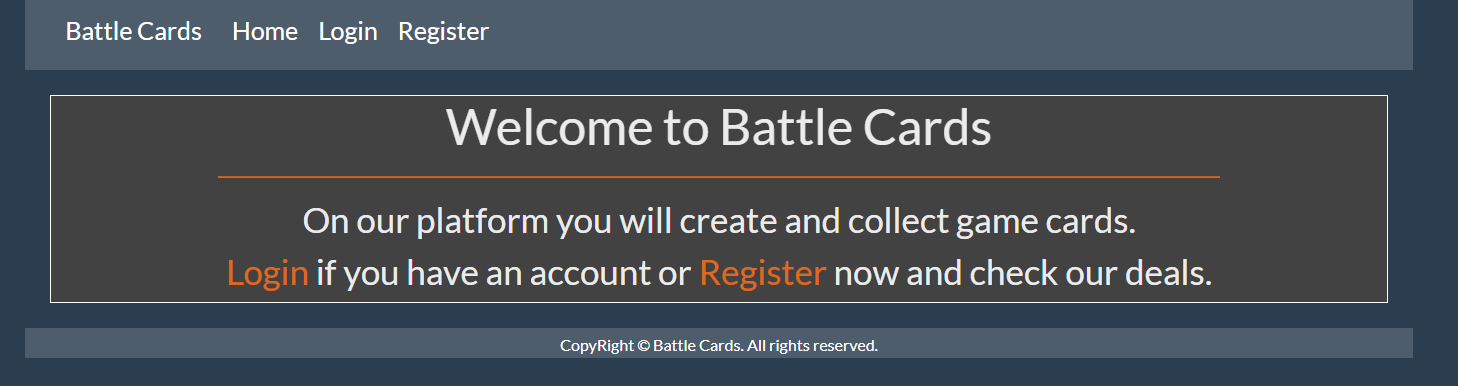
### UserCard – DONE!!!

* Has UserId – a **string**
* Has User – a User object
* Has **CardId** – an **int**
* Has Card – a Card object

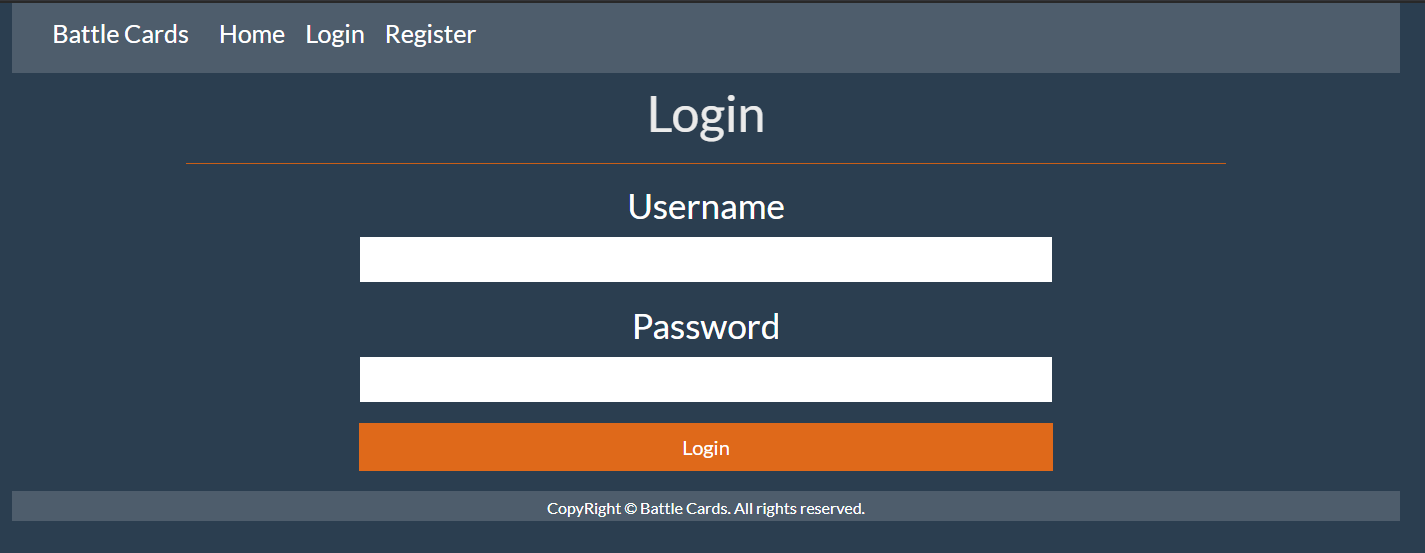
Implement the entities with the **correct datatypes** and their **relations**.

## Page Requirements

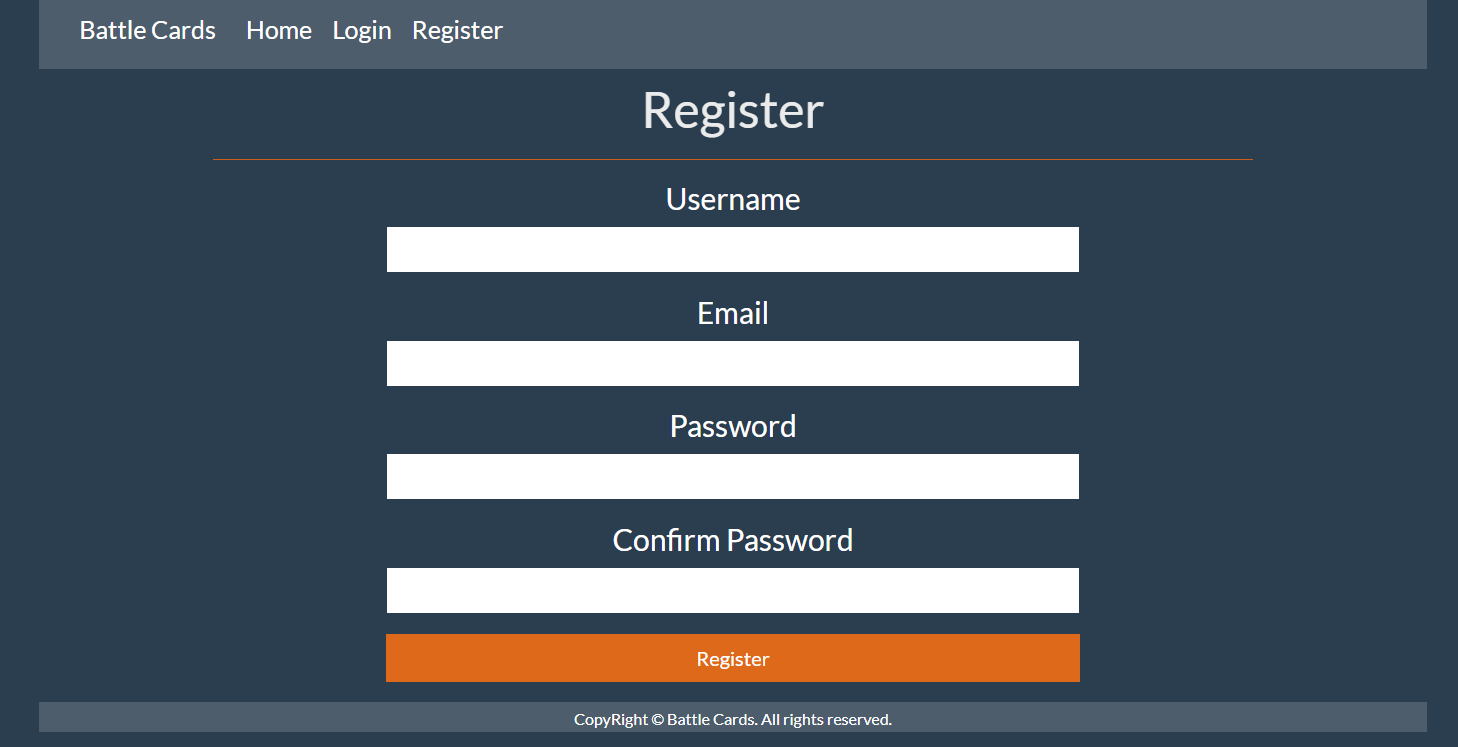
### Index Page (logged-out user) – DONE!!!



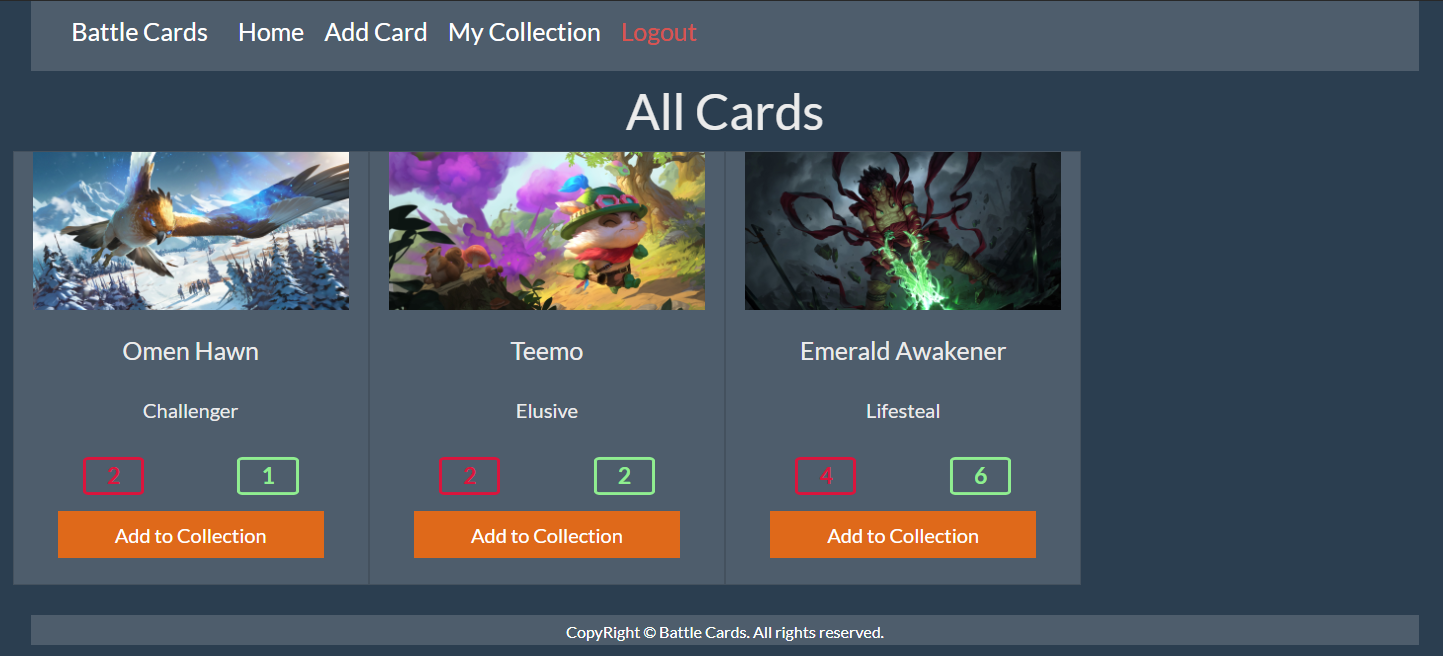
### Login Page (logged-out user) – DONE!!!



### Register Page (logged-out user) – DONE!!!

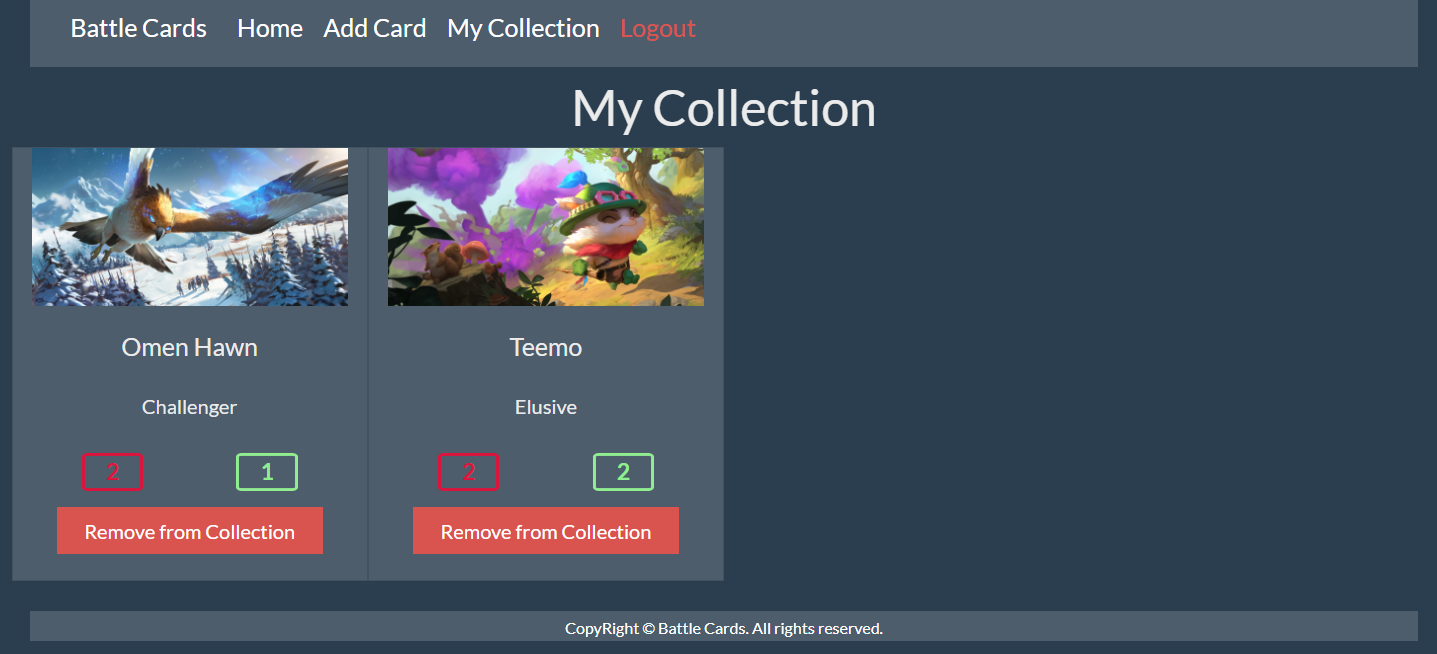


### /Cards/All (logged-in user) – DONE!!!

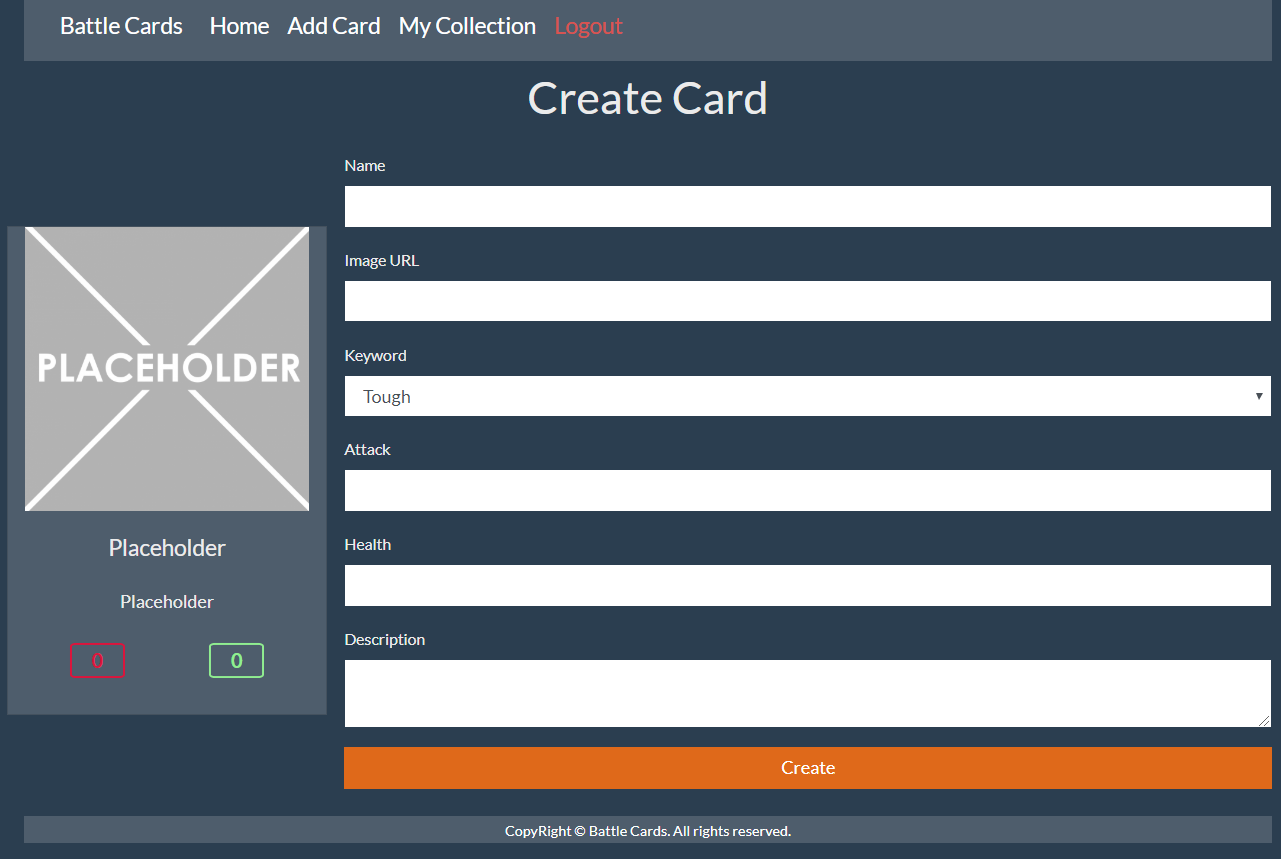


**NOTE**: If the user is logged in and he tries to go the home page, the application must redirect him to the **/Cards/All**

### /Cards/Collection (logged-in user) – DONE!!!!



### /Cards/Add (logged-in user) – DONE!!!



### /Cards/AddToCollection?cardId={cardId} (logged-in user)

Adds the selected card to the user’s collection of cards. If the card is already in their collection, it shouldn’t be added. If everything is successful, the user must be redirect to the home ("/Cards/All") page.

### /Cards/RemoveFromCollection?cardId={cardId} (logged-in user)

Removes the selected card from the user’s collection of cards. If everything is successful, the user must be redirect to their collection ("/Cards/Collection") page.

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

**NOTE**: The templates do **NOT** **require** **additional** **CSS** for you to write. Only **bootstrap** and the **given css** are enough.

## Functionality

The functionality of the **Battle Cards** Platform is very simple.

### Users – DONE!!!

Guests can Register, Login and view the Index Page.

Users can AddCards and see added Cards on the Home Page. From the Home Page they can also view Info about each one of those Cards and Add them to their collection.

### Cards

Cards can be Added by Users. When User adds a Card, it has to be added to their collection too. All created Cards are visualized on the Home Page, each one in its own separate rectangular element.

Cards are visualized on the Home Page with all their information. Description shows as a tooltip when their name is hovered.

Cards are visualized on the Home Page with button – [**Add to Collection**].

* The [**Add to Collection**] button adds the Card to the User’s collection of Cards **unless it is already contained**.

Usershave a Collection page where only the Cardsin their collection are visualized.

* The [**Remove from Collection**] button removes the Card from the User’s collection of Cards.

### 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 /Cards/All.
* Upon successful Creation of a Card, you should be redirected to the /Cards/All.
* Upon successful Adding a Card to the User’s collection, should be redirected to the /Cards/All.
* Upon successful Removal of a Cardfrom the User’s collection, should be redirected to the /Cards/Collection.
* If a User tries to **add** an **already contained** Card to their **collection**, they should be redirected to /Cards/All (or just a page refresh).
* If any of the **validations** in the POST forms **don’t pass**, **redirect** to the **same page** (**reload/refresh** it).

## Security – DONE!!!

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

* Guest (not logged in) users can access Index page.
* Guest (not logged in) users can access Login page.
* Guest (not logged in) users can access Register page.
* Guests (not logged in) cannot access Users-only pages.
* Users (logged in) cannot access Guest pages.
* Users (logged in) can access Cards Add page and functionality.
* Users (logged in) can access Cards All page.
* Users (logged in) can access Cards Collection 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.