# C# Web Basics Exam – 16 June 2019

# SULS

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).

**SULS** is an online judge platform which manages problems and submissions of users. You have been tasked to implement a platform in which the users can create problems or submit solutions to other users' problems.

## Technological Requirements

* Use the **SIS.WebServer**
* Use the **SIS.MvcFramework**
* Use **Entity Framework Core**

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 **SULS** needs to support **3 entities**:

### User

* 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, which holds only **valid** email (**required**)
* Has a Password – a string with **min length** **6** and **max length 20** - hashed in the database (**required**)

### Problem

* Has an Id – a **string, Primary Key**
* Has a Name – a string with **min length** **5** and **max length 20** (**required**)
* Has Points– an **integer** between **50** and **300** (**required**)

### Submission

* Has an Id – a **string, Primary Key**
* Has Code – a string with **min length** **30** and **max length 800** (**required**)
* Has Achieved Result – an **integer** between **0** and **300** (**required**)
* Has a Created On – a DateTime object (**required**)
* Has Problem – a Problem object
* Has User – a User object

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

## Page Requirements

### Index Page (logged-out user)



### Login Page (logged-out user)



### Register Page (logged-out user)

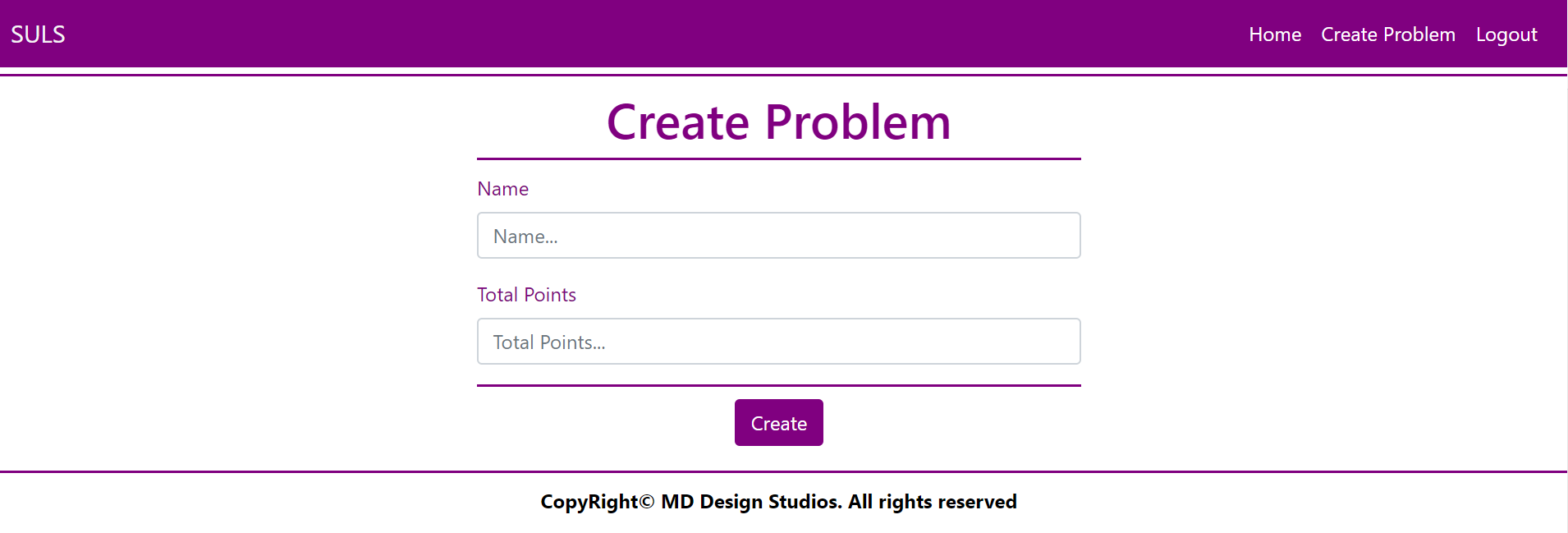


### Home Page (logged-in user)



**NOTE**: The [Submit] and [Details] buttons lead to the Create Submission Page and Details Problem Page respectively.

### Problem Create Page (logged-in user)



### Problem Details Page (logged-in user)



**NOTE**: All Submissions for the current Problem are rendered below in the Problem Submissions section.

**NOTE**: Submissions are visualized with percentage Achieved Result / Problem Total Points, **rounded** to the **closest integer**.

### Create Submission Page (logged-in user)



**NOTE**: The Submission Code will be sent from a <textarea>, as a **whole string**.

**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 SULS Platform is very simple.

### Users

Guests can Register, Login and view the Index Page.

Users can Create Problems and see Created Problems on the Home Page. From the Home Page they can also view Details about each one of those Problems or Submit a Submission to that Problem. Users can also delete submissions from the table with the Problem Submissions on the Problem Details Page.

### Problems

Problems can be Created by Users. All created Problems are visualized on the Home Page, each one in its own separate rectangular element.

Problems are visualized on the Home Page as a table with **Name** of the **Problem**, **Count** of **Submissions** and actions like **Submit** and **Problem** **Details**.

Problems are visualized on the Home Page with 2 buttons – [Submit] and [Details].

* The [Submit] button leads to the Create Submission Page
* The [Details] button leads to the Details Page

### Submissions

Submissions can be created by Users for a Problem. All created Submissions, for a Problem, are visualized on that Problem’s Details Page.

Submissions’ Code is created from a <textarea>, which submits a giant text string.

* Upon creation, the Submission’s Achieved Result property should be set to a **random value**   
  between **0** and the Problem’s Total Points.
* Upon creation, the Submission’s Created On property should be set to the **current** Date and Time.
* Upon creation, the Submission’s User property should be set to the **currently logged-in user**.

### 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 Home Page.
* Upon successful Creation of a Problem, you should be redirected to the Home Page.
* Upon successful Creation of a Submission, you should be redirected to the Hom**e** Page.
* Upon successful Deletion of a Submission, you should be redirected to the Hom**e** Page.
* Upon successful Logout of a User, you should be redirected to the Index Page.

## 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 Index page.
* Guest (not logged in) users can access Login page.
* Guest (not logged in) users can access Register page.
* Users (logged in) cannot access Guest pages.
* Users (logged in) can access Home page.
* Users (logged in) can access Problem Create page and functionality.
* Users (logged in) can access Problem Details page.
* Users (logged in) can access Submission Create page and functionality.
* Users (logged in) can delete Submissions.
* 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.

## Data Validation

* In case of invalid data during the **login**, redirect to "**/Users/Login**"
* In case of invalid data during the **registration**, redirect to "**/Users/Register**"
* In case of invalid data during the **submissions creation**, redirect to "**/Submissions/Create**"
* In case of invalid data during the **problems creation**, redirect to "**/Problems/Create**"

**NOTE:** Use the ValidationSisAttributes

## Scoring

### Database Requirements – 10 points.

### Template Requirements – 10 points.

### Functionality – 40 points.

### Security – 10 points.

### Application Structure – 10 points.

### Data Validation – 20 points.