# Judge – 29 October 2017

The judge platform is a place in which users, can submit code and compile it. Your task is to create a web application that has back-end and front-end logic using the provided resources (**Web Server**, **MVC Framework**, **Bootstrap** and views).

## Data Models

Create the required **entities**. Use the appropriate **data types**.

* The system contains information about **users**, **contests** and **submissions**
* Users can **register** in the system. After successful registration, the user has **email, password, full name, list of contests** and **list of submissions**
* A **contest** only has **name**
* A submission has executable **code** and **contest**

## Functionality

* **Guests** (anonymous users) can register an account with their own email and password
* **Guests** can login by email and password.
* **Guests** can view the default home page including links to **login** and **register**
* **Logged-in users** can logout.
* **Logged-in users** can **create** contests
* **Logged-in users** can **edit** and **delete their** **contests**
* **Logged-in users** can **create** **submissions**
* **Logged-in users** can **view all contests**
* **Admins** can **create** contests
* **Admins** can **edit** and **delete** all contests
* When guest user tries to access a page that is allowed only for logged in user he should be **redirected to the login page**

## Design the Database

Design **entity classes** and create a **database** to hold the **users**, **contests** and **submissions**

Score: **10** **points**

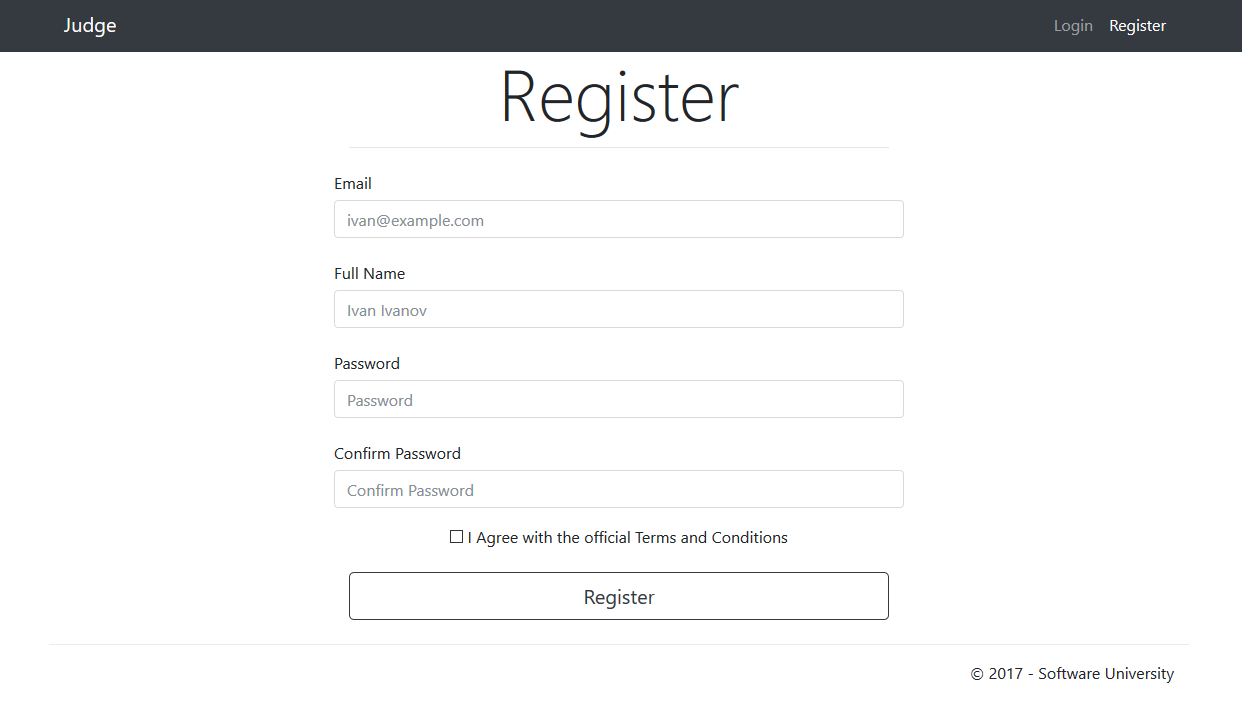
## Implement User Registration, Login and Logout - Done

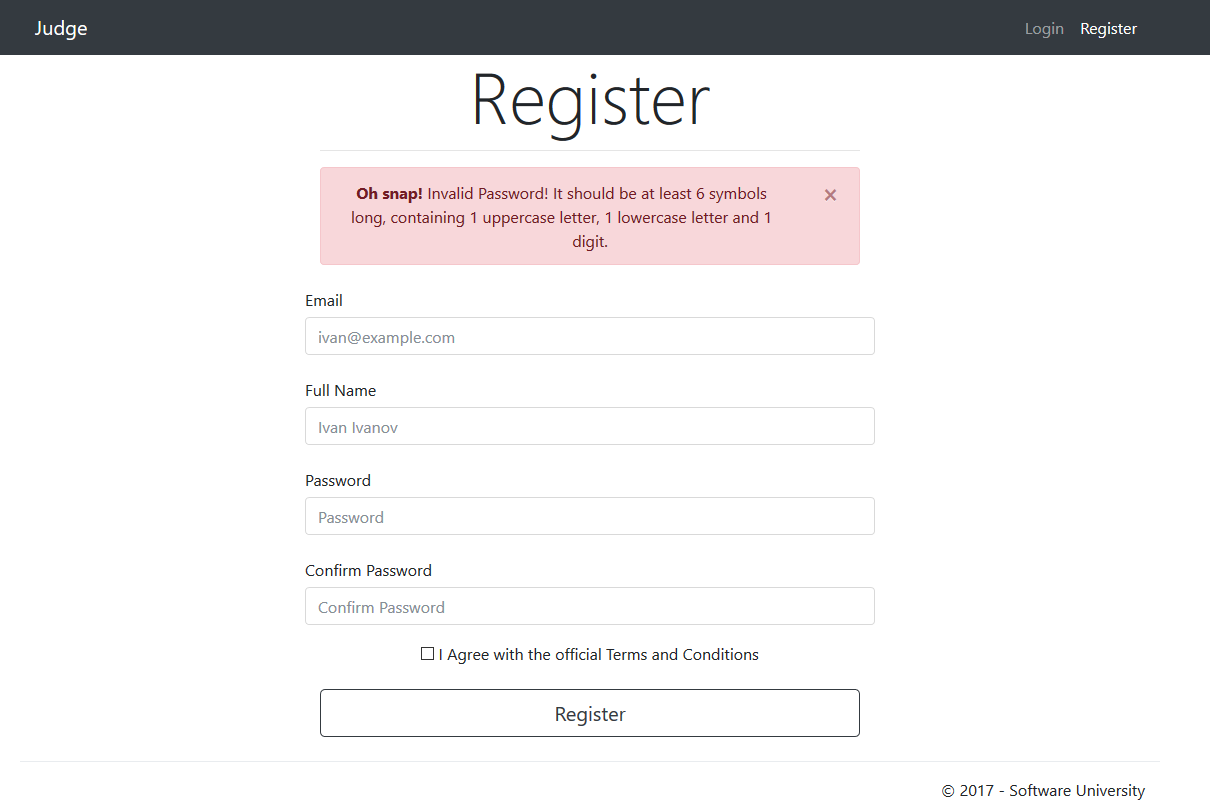
The guest users can register and login. He must enter the following:

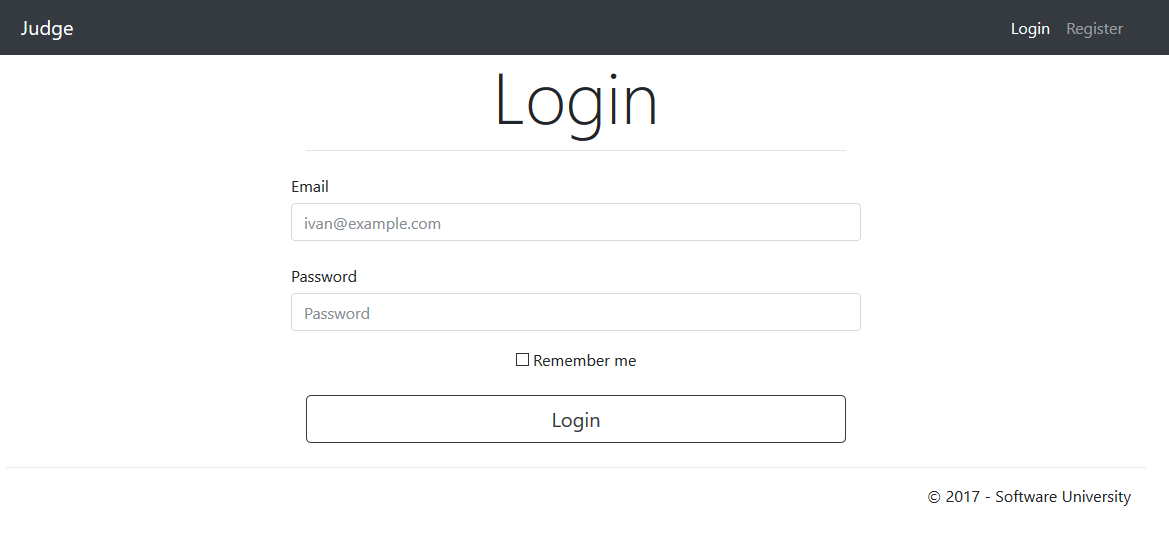
* **Register**
  + **Email** – must contain **@** **sign** and a **period**. It must be unique.
  + **Password** – length must be **at least 6 symbols** and must contain at least **1 uppercase**, **1 lowercase** letter and **1 digit**
  + **Confirm Password** – must **match** the provided password
  + **Full Name**
* **Login**
  + User can log in with **email** and **password**
* **Logout**
  + When logged in the user should have option to **log out**

After register, the user should be **redirected** to the login page. The **first user** to **register** becomes an **admin**. If there are **any errors** with the registration, they **should be** **displayed** in the register form.   
**Note**: You should display a **single error message**, for each error that occurs, e.g. If the password is **invalid**, you display an invalid password message as the one shown below.

All validations should be performed both on the back-end and front-end.







Each type of user must have **unique navigation bar**.

Guest:



Logged In User:



Score: **10** **points**

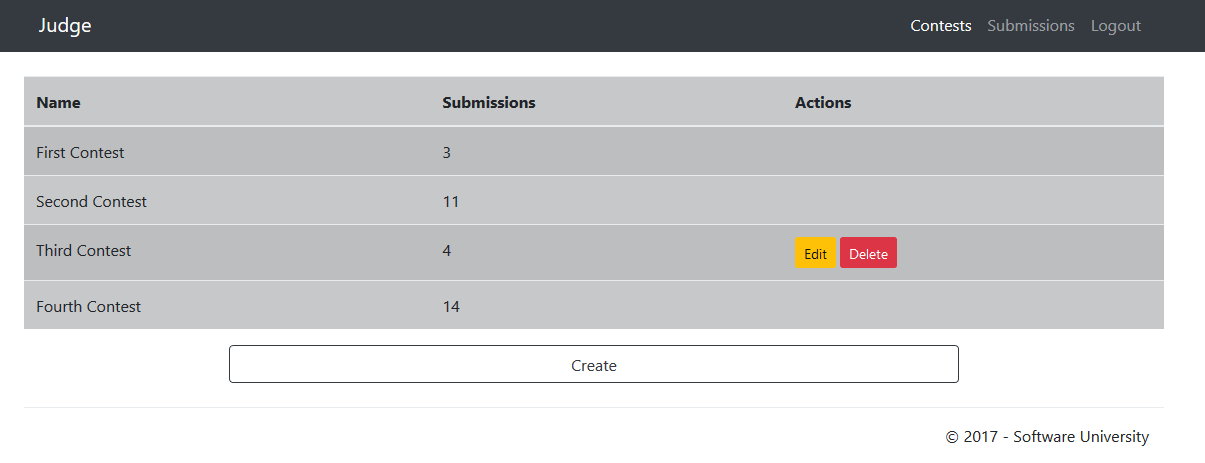
## Implement Managing Contests

As a user, you have the option to **add contests to the database**. A contest should be added only if it matches those criteria:

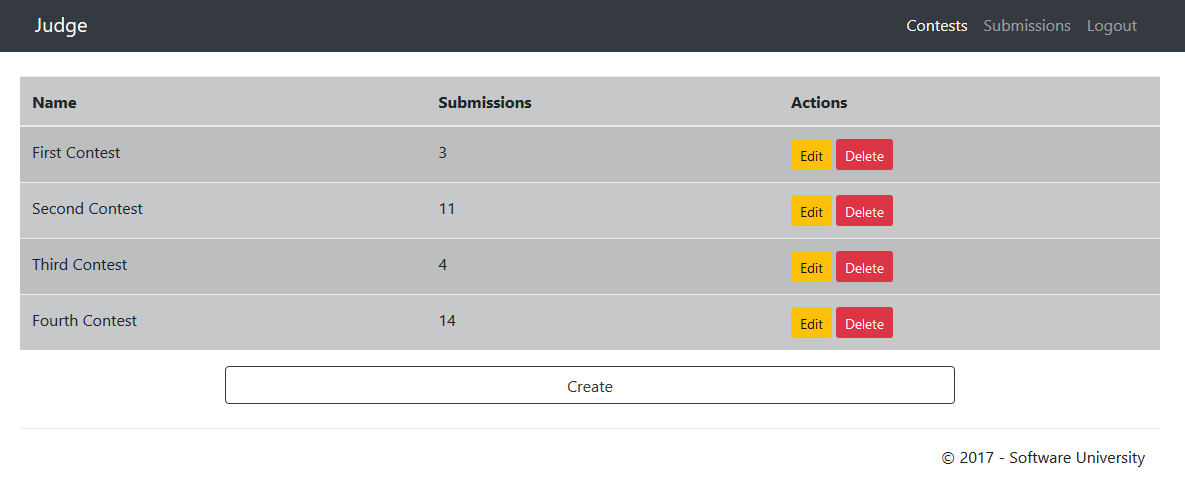
* **Name** – must begin with **uppercase letter** and has length between **3 and 100 symbols** (inclusive)

All validations must be performed on the back-end. If there are **any Errors**, they should be displayed.

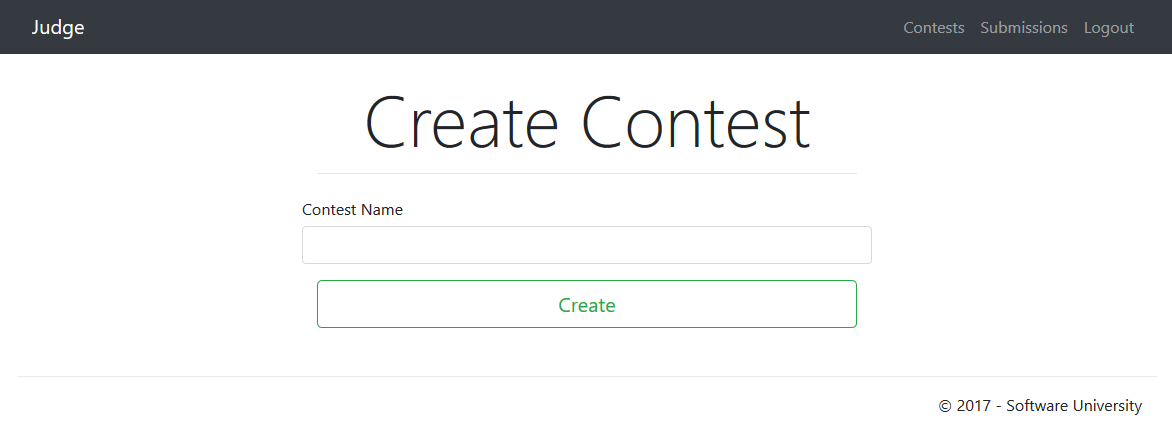
Listing all contests:



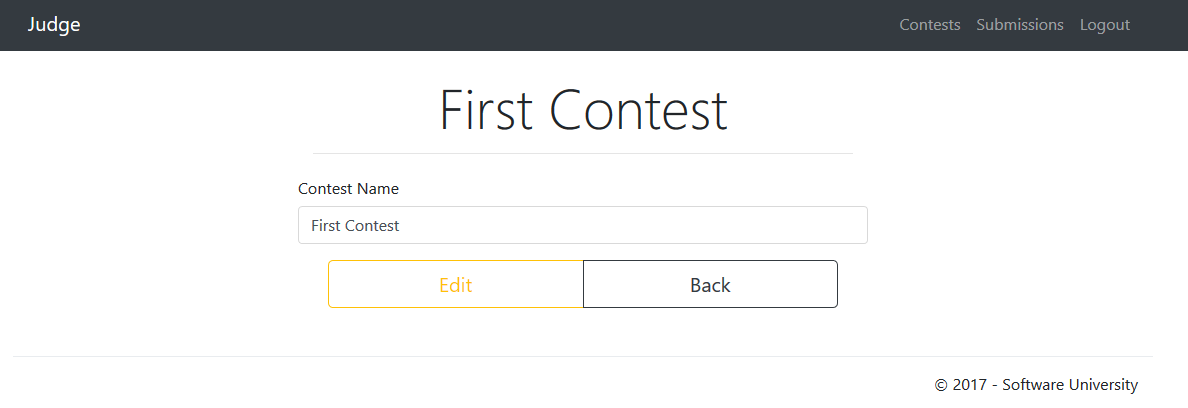
Admin Listing:



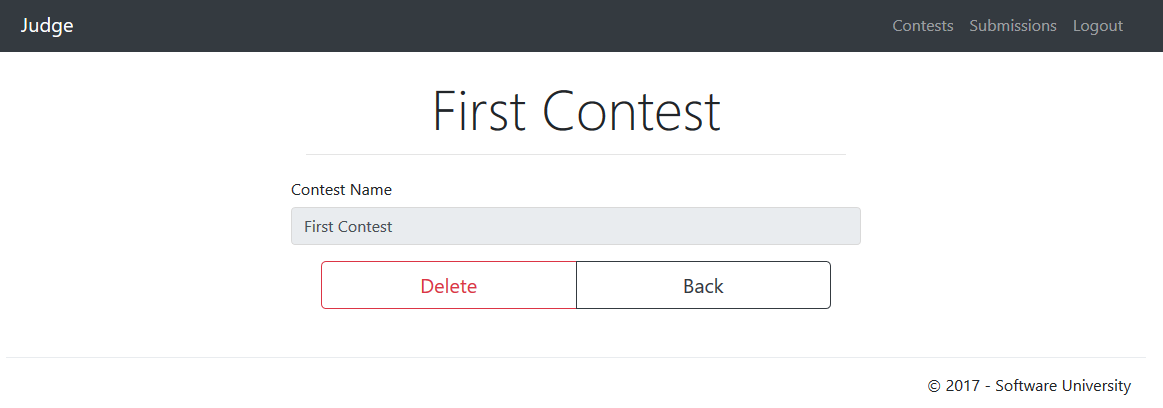
Adding a contest:



Editing a contest:



Deleting a contest:

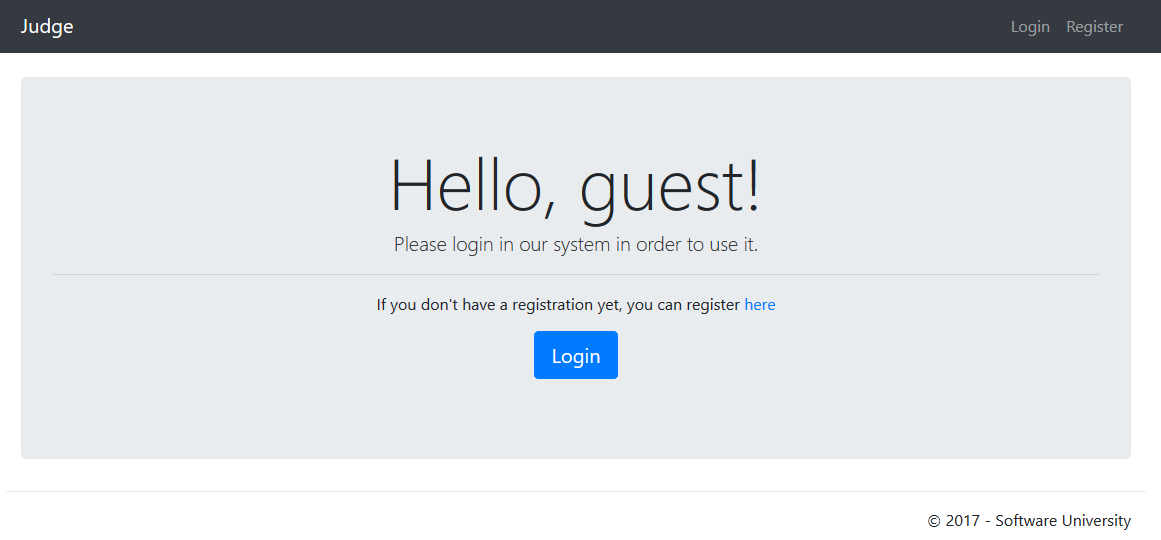


Score: **20** **points**

## Implement Guest Home Page

It should show a message telling the user to login or register.

**Guest** home page:

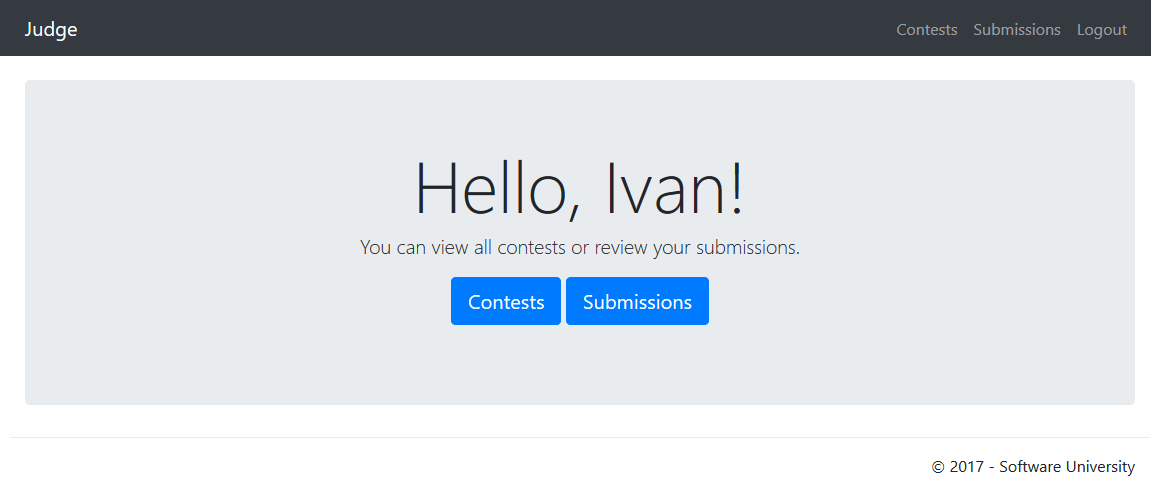


Score: **10** **points**

## Implement User Home Page

It should show a message telling the user that he can view all contests and submissions.

**User** home page:

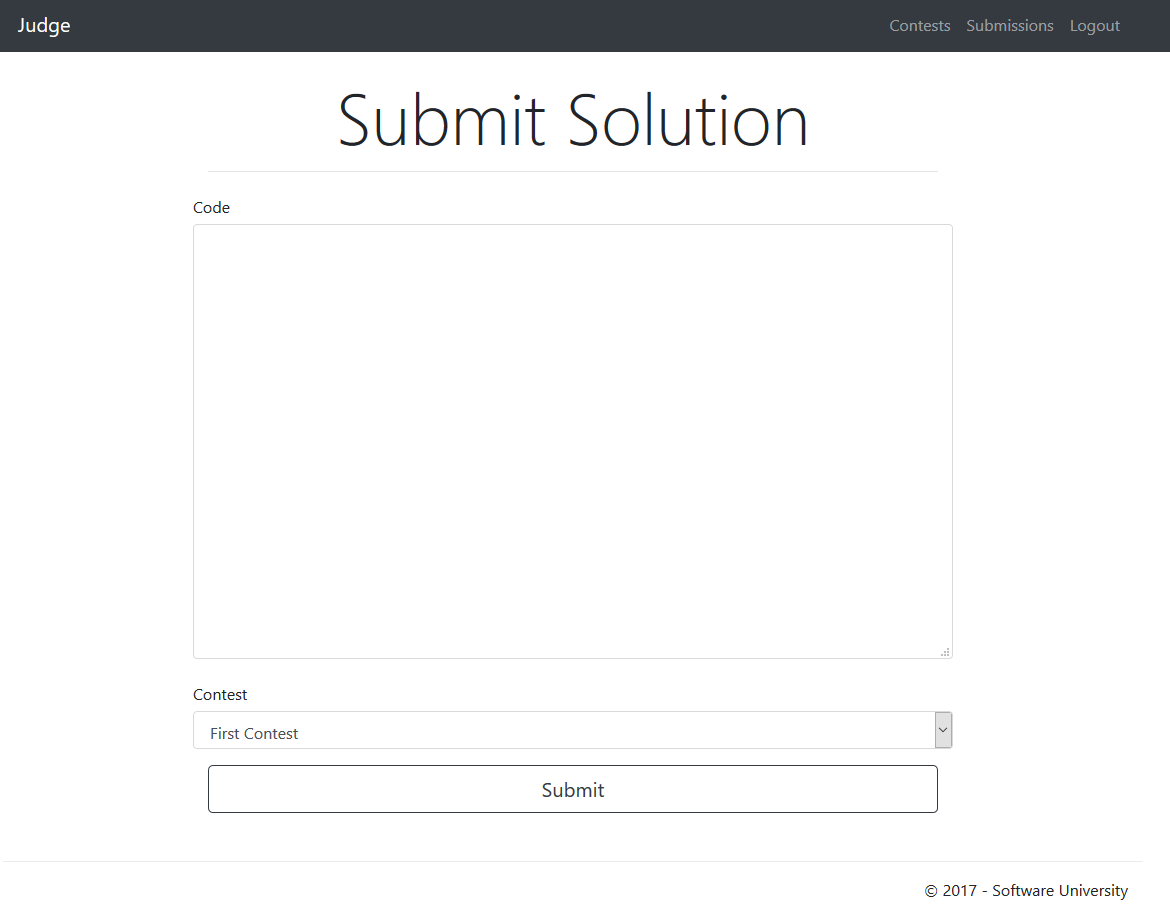


Score: **10** **points**

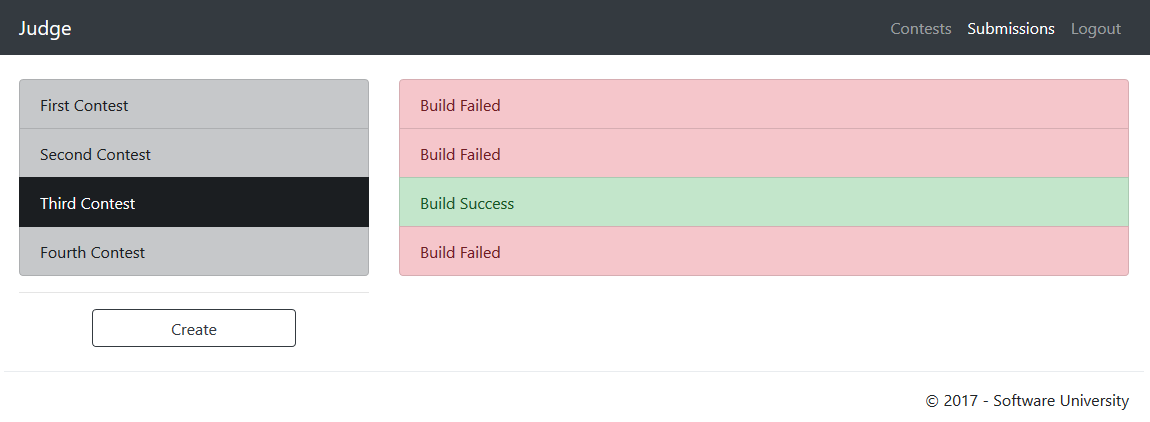
## Implement Managing Submissions

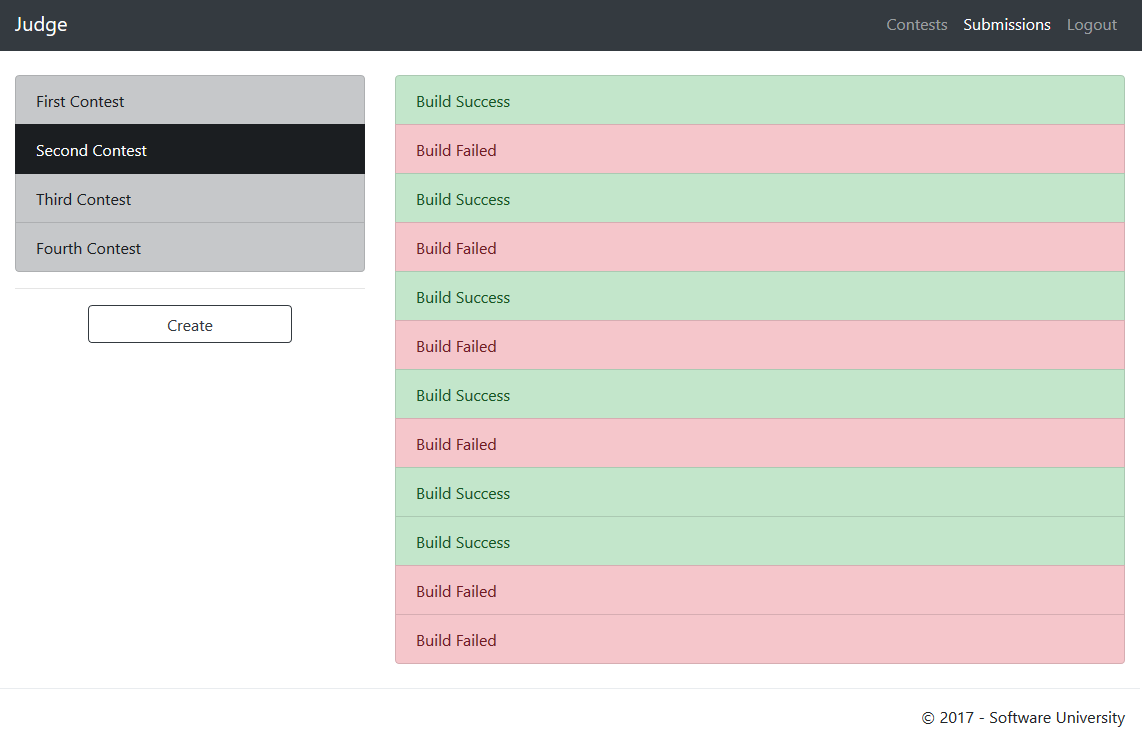
To create submission, you need a code and contest. Each submission should be "processed" at **random**, with a 70% chance to fail to compile.

**Create** **Submission** Examples:



**Listing** **Submissions** Examples:





**NOTE**: **All Users** can only see their own **submissions** (**Admins** too).

Score: **25** **points**

## Project Infrastructure

Points for code quality / good application structure / additional effort.

* Points for implementing **separate data layer**.
* Points for using **dependency inversion**.
* Points for using **services** for different jobs.

Score: **15 points**

## Bonus

* Points for using **AutoMapper**.

Score: **5 points**