# Teamwork Project Assignment for the [ASP.NET MVC Course @ SoftUni](https://softuni.bg/courses/asp-net-mvc/)

This teamwork project assignment is designed to develop skills for creating dynamic data-driven Web applications using ASP.NET MVC and deployment in а cloud environment like Azure and AppHarbor.

## Project Description

Design and implement a **photo contest site**.

The application should allow users to create and participate in contests for images. Each user can be both an organizer and a participant in different contests. Organizers should have a variety of options for how to carry out the contest.

We have laid out some application design suggestions but feel free to implement your own. Get creative!

A **contest** must have a title, description, reward strategy, voting strategy, participation strategy and deadline strategy. A contest also keeps track of all **pictures** submitted to it and their **votes**.

The reward strategy is one of the following:

* Single winner – the owner of the picture with the most votes wins
* Top N prizes – list of prizes for the first N participants with the most votes

The voting strategy is one of the following:

* Open – everyone can vote
* Closed – only members of a committee (invited by the contest owner) can vote

The participation strategy is one of the following:

* Open – everyone can submit images
* Closed – participants are pre-selected by the contest owner

The deadline strategy is one of the following:

* By time – the contest closes automatically and may accept no new submissions at the given time
* By number of participants – the contest closes automatically after the selected number of participants have joined the contest

No matter what the deadline strategy is, the contest owner can dismiss (stop, selecting no winners) and end (finalize, selecting a winner / winners according to the contest reward strategy) a contest.

## General Requirements

Your Web application should use the following technologies, frameworks and development techniques:

* The application must be implemented using **ASP.NET MVC** framework.
* Use **Visual Studio 2015 or 2013** (Update 4 is recommended).
* Use **Razor** template engine for generating the UI.
  + Rendering with ASP.NET Web Forms is not allowed.
  + Use **sections** and **partial views**.
  + Use **editor** and **display templates**.
* Use **Microsoft SQL Server** as database back-end.
* Use **Entity Framework 6** to access you database.
  + Obligatorily use **Repository** and **Unit of Work** patterns.
* Use **MVC Areas** to separate different parts of your application (e.g. area for administration).
* Adapt the default **ASP.NET MVC site template** from Visual Studio 2013 or get another free theme.
  + Use responsive design based on **Twitter Bootstrap**.
* Use the standard **ASP.NET Identity System** for managing **users** and **roles**.
  + Your registered users should have at least one of these roles: **user** and **administrator**.
* Use **AJAX** request to asynchronously load and display datasomewhere in your application.
* Use **SignalR** communication somewhere in your application.
* Write **unit tests** for your logic, controllers, actions, helpers, etc.
* Implement **error handling** and **data validation** to avoid crashes when invalid data is entered (both **client-side** and **server-side**).
* Handle correctly the special HTML characters and tags like **<br />** and **<script> (escape special characters).**
* **Use Ninject (or any other dependency injection container).**
* **Use AutoМapper.**
* **Prevent from security vulnerabilities like SQL Injection, XSS, XSRF, parameter tampering, etc.**
* Host the application in a **cloud environment**, e.g. in **AppHarbor** or **Azure**.
* Obligatorily use a **file storage cloud API**, e.g. **Dropbox**, **Google Drive** or other for storing the files.

### Public Part

The **public part** of your application should be **visible** **without authentication**. All users can see the active contests and the past contests, ordered by date (from the soonest to the earliest).

**Design suggestion:**

All users can see all the active contests on the home page. Upon clicking a contest, a user can go to the contest details page (also public) and vote for the winners (in case the voting strategy for the contest permits). There can also be a "Past Contests" page where everyone can see the ended / dismissed contests **without** their winners.

### Private Part (User Part)

**Registered users** should be able to **login**. This can happen with a local account, and via Facebook and Google. You may also link the application to other external login providers.

Registered users can:

* Manage contests:
  + **Create** a contest
  + **Update** a contest (if they are the owners)
  + **Dismiss** a contest – stop the contest and select no winners
  + **Finalize** a contest – initialize the end of the contest and choose a winner / winners in accordance with its voting strategy
* Participate in contests:
  + **Register** **for** **a** **contest** – a user can enter an open contest freely, or be invited to a closed contest by its owner. It’s only after being invited that the user can participate in a closed contest
  + **Upload** **image** as an entry for a specific contest. A user may submit more than one image for a contest
  + **View** **contest** **results** – see the winner(s) of a contest

**Design suggestion:**

Once a user is logged in, display a menu (or some links) to the user’s contests. Create a page listing all the user’s contests and provide options for creating new contests and editing currently active contests. For example, you can display all contests and have an "Edit" button. When the user presses it, a page with the contest parameters opens and the user is free to edit the contest title, details and deadlines. There can also be a "Dismiss" and a "Finalize" button, along with the "Edit" button.

On the home page, the registered users can also see the active contests, but now they should have the option to participate in one. If an active contest is closed and the user is not invited, they should not be able to participate in it.

### Administration Part

**An administrator** should have access to all contests, as if he / she is the contest creator. He / She can also manage other users’ profiles (excluding their own profile, and excluding usernames). The admins can also delete pictures from contests if they think they are inappropriate.

A user can be an administrator and still be able to create and take part in contests (i. e., admins have all rights that non-admins have, plus some more). An administrator **cannot edit votes** for pictures.

**Design suggestion:**

Reuse the logic (and code, if possible) for the user’s contests section. This time, display all contests and allow editing them. For a specific contest, display a "Delete Picture" button next to every picture. Be sure to ask the administrator whether they really want to delete that picture.

For the users, you may display them in a grid (optionally, with the ability to search by username) and allow editing their personal details.

## Additional Requirements

* Follow the best practices for OO design and **high-quality code** for the Web application:
  + Use data encapsulation.
  + Use exception handling properly.
  + Use inheritance, abstraction and polymorphism properly.
  + Follow the principles of strong cohesion and loose coupling.
  + Correctly format and structure your code, name your identifiers and make the code readable.
* Well looking user interface (UI).
* Good usability (easy to use UI).
* Supporting of all modern Web browsers.
* Use caching where appropriate.
* Use a **source control system** by choice, e.g. Git, SVN, GitHub, CodePlex.
  + Submit a link to your public source code repository.

## Public Project Defense

Each team will have to deliver a **public defense** of its work in front of the other students, trainers and assistants. Teams will have **only 15 minutes** for the following:

* **Demonstrate** how the application works (very shortly).
* Show the **source code** and explain how it works.
* Explain how each team member has **contributed**: display the commit logs in the Source Control system you are using.
* Optionally you might prepare a **presentation** (3-4 slides).

Please be **strict in timing**! On the 15th minute you **will be interrupted**! It is good idea to leave **the last 2-3 minutes for questions** from the other students, trainers and assistants.

Be **well prepared** for presenting maximum of your work for minimum time. Bring your own laptop. Test it preliminary with the multimedia projector. Open the project assets beforehand to save time.

## Assessment Criteria

* **Functionality** – **0…20**
* **Implementing controllers correctly** (controllers should do only their work) **– 0...5**
* **Implementing views correctly** (using display and editor templates) **– 0…5**
* **Unit tests** (unit test for some of the controllers using mocking) **– 0…10**
* **Security** (prevent SQL injection, XSS, CSRF, parameter tampering, etc.) **– 0…5**
* **File storage** (cloud file storage) **– 0…5**
* **Data validation** (validation in the models and input models) **– 0…10**
* **Hosting the application in the cloud – 0…5**
* **Using auto mapper and inversion of control – 0…5**
* **Using areas with multiple layouts – 0…10**
* **Code quality** (well-structured code, following the MVC pattern, following SOLID principles, etc.) – **0…10**
* **Teamwork**\* (source control; each team member contributed in 5 different days; distribution of tasks) – **0…5**
* **Bonus** (bonus points are given for exceptional project) – **0..5**

\* If not all team members have contributed to the project, this **does not affect** the teamwork points.

## Give Feedback about Your Teammates

You will be invited to **provide feedback** about all your teammates, their attitude to this project, their technical skills, their team working skills, their contribution to the project, etc. The feedback is important part of the project evaluation so **take it seriously** and be honest.