ASP.NET Core MVC Final Project

Cocktail Magician

This document describes the final project assignment for the ASP.NET Core MVC course at Telerik Academy.

Project Description

Design and implement an ASP.NET Core MVC application – Cocktail Magician.

Cocktail Magician allows creation of recipes for innovative, exotic, awesome cocktails and follows their distribution and success in amazing bars.

- The website visitors (unregistered users) can see the bars and the cocktails they offer
- Bar Crawlers (registered users) can rate cocktails and bars (using stars) and leave comments for them
- Cocktail Magicians (Cocktail Magician employees) can create cocktails (defining their ingredients) and define the bars which can offer them

Public Part

Visible for all website visitors - no authentication required.

The public part consists of a **home page** displaying top rated bars and cocktails in separate sections on the page.

It also includes searching possibility for:

- Bars: by Name, Address and Rating
- Cocktails: by Name, Ingredient/s and Rating

The result of the search should be displayed in a grid structure with paging (if needed).
The information shown in the grid contains

- For the Bars: image, name, address and rating
- For the Cocktails: image, name, ingredients (comma separated) and rating

Upon clicking a bar, the visitor can see details for the bar (image, name, rating, address, phone, and comments) and the cocktails it offers (with links to the cocktail).
Upon clicking a cocktail, the visitor can see details for the cocktail (image, name, rating, ingredients, and comments) and the bars this cocktail is offered in (with links to the bar).

The public part includes **login page** and **register page** as well.

Private Part (Bar Crawlers)

After login, Bar Crawlers see everything visible to website visitors and additionally they can:

- Rate bars (from 1 to 5 scale/stars)
- Rate cocktails (regardless of which bar offers them) (from 1 to 5 scale/stars)
- Leave a comment for a bar (maximum 500 characters)
- Leave a comment for a cocktail (maximum 500 characters)

Administration Part (Cocktail Magicians)

Cocktail Magicians can:

- Manage ingredients – CRUD operations for ingredients for cocktails (delete ingredient only if not used in any cocktail)
- Manage cocktails – CRUD operations for cocktails (never delete a cocktail, just hide it from the users)
- Manage bars – CRUD operations for bars (never delete a bar, just hide it from the users)
- Set cocktails as available in particular bars

** Notes:*

★ *A cocktail consists multiple ingredients*
★ *One ingredient can be present in multiple cocktails*
★ *A bar can have many cocktails*
★ *One cocktail can be present in multiple bars*

General Requirements

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

- Use **ASP.NET Core 2.0+ MVC** and **Visual Studio 2017+.**
- You should use **Razor** template engine for generating the UI.
    - You may use any JavaScript library of your choice
- Use **MS SQL Server** as database back-end.
    - Use **Entity Framework Core 2.0+** to access your database
    - Using **repositories and/or service layer** is a must
- Use at least **2 areas** in your project (e.g. for administration).
- Create **tables with data** with **server-side paging and sorting** for a model entity.
    - You can use Kendo UI grid, jqGrid, DataTables or any other library or generate your own HTML tables
- Create beautiful and responsive UI.
    - You may use **Bootstrap** or **Materialize**

    - You may change the standard theme and modify it to apply own web design and visual styles
- Use the standard **ASP.NET Identity System** for managing users and roles.
    - registered users should have at least one of the two roles: **user** and **administrator**
- **AJAX form** communication in some parts of your application.
- Use **caching** of data where it makes sense (e.g. starting page).
- Apply **error handling** and **data validation** to avoid crashes when invalid data is entered (both client-side and server-side).
- Prevent yourself from **security** holes (XSS, XSRF, Parameter Tampering, etc.)
    - Handle correctly the **special HTML characters** and tags like <script>,
, etc.
- Create **unit tests** for your "business" functionality following the best practices for writing unit tests (**at least 80% code coverage**).
    - Unit test framework: MSTest, XUnit or NUnit.
    - Mocking framework: Moq.
    - DB Provider: EF InMemory.
- Use Dependency Inversion principle and Dependency Injection technique following the best practices.
    - DI containers – Autofac and MS.Extensions.DependencyInjection.
- Integrate your app with a **Continuous Integration server** (Jenkins, AppVeyor or other) - configure your unit tests to run on each commit to your master branch.
- Use GitHub and take advantage of the **branches** for writing your features.
- **Documentation** of the project and project architecture (as .md file, including screenshots, diagrams, etc.)

Partner Evaluations and Criteria

Partner representatives should evaluate the project according to the following criteria:

1. Feature Completeness and Quality
    - Have all requirements been met? Have optional requirements been implemented?
    - Are there any issues with performance, bugs, or malfunction?
2. Code Architecture and Design Quality
    - Loose coupling
    - Modularity
    - Extensibility
3. User Experience and Creativity

The final grade of the project is determined by the following rules.

Each of these 3 criteria are awarded points from 0-5, where 0 is the lowest score and 5 is the highest score:

0	1	2	3	4	5
Missing	Poor	Fair	Good	Very Good	Excellent

Then the final project grade is determined by the following formula:

[Feature Completeness] * 3 + [Code Architecture Quality] * 2 + [User Experience and Creativity] * 1.

Maximum number of points is 5*3 + 5*2 + 5 = 30.

This final score should be awarded by partner representative on **June 11th.**