**ASP.NET MVC Final Project**

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

**Project Description**

Design and implement an **ASP.NET MVC application**. It can be a discussion forum, blog system, e-commerce site, online gaming site, social network, or any other web application by your choice.

The application should have:

* **public part** (accessible without authentication)
* **private part** (available for registered users)
* **administrative part** (available for administrators only)

**Public Part**

The **public part** of your projects should be **visible without authentication**.

This public part could be the application start page, the user login and user registration forms, as well as the public data of the users, e.g. the blog posts in a blog system, the public offers in a bid system, the products in an e-commerce system, etc.

**Private Part (Users only)**

**Registered users** should have private part in the web application accessible after **successful login**.

This part could hold for example the user's profiles management functionality, the user's offers in a bid system, the user's posts in a blog system, the user's photos in a photo sharing system, the user's contacts in a social network, etc.

**Administration Part**

**System administrators** should have administrative access to the system and permissions to administer all major information objects in the system, e.g. to create/edit/delete users and other administrators, to edit/delete offers in a bid system, to edit/delete photos and album in a photo sharing system, to edit/delete posts in a blogging system, edit/delete products and categories in an e-commerce system, etc.

**General Requirements**

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

* Use **ASP.NET MVC** and **Visual Studio 2015**
* You should use **Razor** template engine for generating the UI
  + You may use any JavaScript library of your choice
    - For example Kendo UI widgets (with the ASP.NET MVC Wrappers), Chart.js for charts, etc.
  + ASP.NET WebForms is not allowed
  + Use **sections** and **partial views**
  + Use **editor** and/or **display** templates
* Use **MS SQL Server** as database back-end
  + Use **Entity Framework 6** 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 every model entity
  + You can use Kendo UI grid, jqGrid, 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
  + Your registered users should have at least one of the two roles: **user** and **administrator**
* Use **AJAX form and/or SignalR** 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>, <br />, etc.
* Create **unit tests** for your "business" functionality following the best practices for writing unit tests (**at least 80% code coverage**) - **~30% of the points for the project** (**IF YOU HAVE UNDER 50% CODE COVERAGE YOU WILL NOT PASS THE EXAM**)
* Use **Dependency Inversion** principle and **Dependency Injection** technique following the best practices - **~20% of the points for the project**
* 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 (**MANDATORY REQUIREMENT**)
* Use GitHub and take advantage of the **branches** for writing your features.
* **Documentation** of the project and project architecture (as .md file, including screenshots)

**Optional Requirements (bonus points)**

* Write integration tests and configure them to run as part of your CI build
* Originality of the idea (uniqueness)
* Using some king of machine learning (AI)
* Using external devices (e.g. Raspberry Pi)
* Host your application in Azure (or any other public hosting provider)

**Deliverables**

Put the following in a **ZIP archive** and submit it:

* The **source code** (everything except /bin/, /obj/, /packages/)
* The project documentation
* Screenshots of your application
* If hosted online - the URL of the application

**Public Project Defense**

Each student will have to make a **public defense** of its work to the trainers (~30-40 minutes). It includes:

* Live **demonstration** of the developed web application (please prepare sample data).
* Explain application structure and its **source code**
* Show the **commit logs** in the source control repository to prove a contribution from all team members.

23.03 – 26.03

23.03

- Add Reviews

24.03

- Add Cart

- Cart Page

25.03

- Create Paging

- Use Caching

- Server side paging

- Some sort of Ajax with paging

- Partial Views error handling

- Continious Integration and unit tests and final stuff.

- More Unit Tests

26.03

- Prepare and check everything