**ASP.NET Web Forms Teamwork**

This document describes the **teamwork assignment** for the **ASP.NET Web Forms** course at Telerik Academy.

**Project Description**

Design and implement a **data-driven ASP.NET Web Forms 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)
  + Task descriptions
  + Leaderboard
* **private part** (available for registered users)
  + Submit task
  + Profile management
* **administrative part** (available for administrators only)
  + Add tasks

**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 (User Area)**

**Registered users** should have personal area in the web application accessible after **successful login**. This area 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 Web Forms** and **Visual Studio 2015**
* Your UI should use **server-side Web Forms** UI rendering (ASPX pages and ASCX controls)
  + ASP.NET MVC and JavaScript UI controls are **not** allowed!
* Use **MS SQL Server** as database back-end
  + Use Entity Framework to access your database
* Use **data-binding** technique by choice
  + You are free to use data-source controls (like EntityDataSource and ObjectDataSource), model binding or manual binding in the C# code behind pages.
* Use at least **four data grids** (table-like data UI components) with **server-side paging and sorting**
  + Leader board
  + Task commits history
  + Task leaderboard
  + Tasks on tasks page
* 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 a **Master page** to define the common UI for the public, private and administrative parts
* Use **Sitemap** and navigational UI controls to implement site navigation
* Use the standard **ASP.NET Identity System** for managing **users** and **roles**
  + Your registered users should have at least two roles: **user** and **administrator**
* Use the standard ASP.NET Web Forms controls (from System.Web.UI)
  + External UI controls from Telerik / Infragistics / DevExpress / etc. are **not** allowed!
* Use UpdatePanels and **AJAX** where applicable to avoid full postbacks
* Use at least **three ASCX user controls** that encapsulate some functionality
* Use at least one **file upload** form to send files at the server side (e.g. profile photos for your users)
  + Profile photo
  + Script file
* 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
* 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 70% code coverage**) - **~30% of the points for the project**
* Use **MVP pattern** in collaboration with the **Dependency Inversion** principle and **Dependency Injection** technique - **~20% of the points for the project**
* Use GitHub and take advantage of the **branches** for team collaboration.
* Brief **documentation** of the project and project architecture (as .md file)

**Optional Requirements**

* Nice looking UI supporting of all modern and old Web browsers
* Good usability (easy to use UI)
* Code coverage **above 90%** - **bonus points**

**Deliverables**

* Record a short video showcasing your application
  + ~1-2 minutes, just show the interesting features
  + Do not record register/login functionality, this is not interesting...
* Your application should be hosted locally (on IIS)
* Register your application at [Our Showcase System](http://best.telerikacademy.com/)
  + Link to the video
  + Link to the GitHub repository

**Public Project Defense**

Each team will have to make a **public defense** of its work to the trainers (in 20-30 minutes per team member). It includes:

* Live **demonstration** of the developed web application (please prepare sample data).
* Explain application structure and its **source code**: ASPX pages, C# code, data-bindings, ASCX controls, etc.
* Each team member **must know everything about the project** (**what**, **why** and **how**) and will be presenting it separately.