# ASP.NET MVC – update Practical Exam – November 2014

You are assigned to improve 2 existing ASP.NET MVC web applications. Both systems are developed during the workshops in the course. The first one is the **Forum System** (<https://github.com/NikolayIT/ForumSystem>) and the second one is the **EasyPTC System** (<https://github.com/ivaylokenov/EasyPTC>).

Download the source codes, unzip them in 2 separate directories and work directly on the existing code.

Download link for the Forum System: <https://github.com/NikolayIT/ForumSystem/archive/master.zip>

Download link for the EasyPTC System: <https://github.com/ivaylokenov/EasyPTC/archive/master.zip>

### Do some improvements in the Forum System (6 points)

1. Connect **Post** model with **Tag** model (**many-to-many** relationship) (2 points)
2. **Seed** the database with at least **10 forum posts** (with at least **3 tags per post**). Use the Seed method of the EF code-first migrations class in the project (Configuration.cs) (1 point)
3. Fix **forum posts links** in Index view of the Home controller. See the RouteConfig.cs for more details. (1 point)
4. In the Index view of the Home controller **display proper tags for each Post** (2 points)

### Create Feedback Form in the Forum System (12 points)

Create a new database model (Feedback) with the following properties:

* **Author** (ApplicationUser), nullable
* **Title** (string, max length: 20 characters)
* **Content** (string)
* **CreatedOn** (datetime), **DeletedOn** (nullable datetime), **IsDeleted** (bool) and **ModifiedOn** (nullable datetime)

1.5 points

Implement a new **FeedbackController** class with two actions:

* [HttpGet] **Create()** for showing the form. Create the corresponding view for the form using Bootstrap.
  + Content **is allowed** to contain HTML tags. It is **not required** to implement HTML editor but the Content field should be multiline textbox field.
  + Create a view model for the form.

4 points

* [HttpPost] **Create(input data)** for accepting the submitted form and saving the data. When user is not logged-in the Author property of the Feedback model should be null.
  + **Redirect** to the index page when done and show notification message.

4.5 points

Use **Dependency Inversion** principle and **IDeletableRepository** for working with the data in the FeedbackController and protect the form from **XSRF**.

2 point

### Create pageable table that displays all feedbacks in the Forum System (12 points)

* Create **PageableFeedbackListController** that is accessible only for registered users.

1 point

* Implement view that displays table with 4 columns (with **Author** (username), **Title**, **Content** and **Created On**)

1 point

* + Implement **action** that gets the data from the database and prepares it for the view
  + Use **view model** for transferring data between action and controller

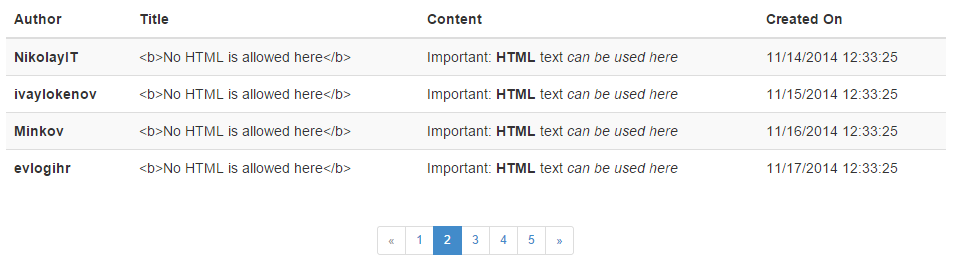
2 point

* Display **Content** as plain HTML
  + Be sure to protect the page from **XSS** (e.g. by **sanitizing** the content before displaying it)

2 points

* Make the table **pageable** (by accepting page parameter and displaying appropriate list of data )
  + Create **links to all pages** and links to **previous** and **next** pages (e.g. as in the picture below)
  + Display **4 items per page**

3 points



* **Cache** each page data. You are free to use **OutputCache** or **this.HttpContext.Cache**.Each page data should be cached separately. Note that OutputCache caches the full html of the page including the logged username.

2 points

* Use **Dependency Inversion** principle and **IDeletableRepository** for working with the data in the PageableFeedbackListController.

1 point

You are **not allowed** to use any libraries (like KendoUI, jqGrid, or any MVC-ready NuGet packages for simplifying paging). The idea here is to create everything on your own. The only thing you can use is Bootstrap for creating more beautiful result.

### AJAX voting (for logged users only) (11 points)

* Create proper database model(s) for saving votes
* Implement up and down voting for each question listed in the index page.
* One logged-in user can either up vote or down vote for each question
* Users should be able to un-vote the already voted questions
* Voting is only enabled for logged-in users. Use AJAX for voting.
* Display voting result along with the appropriate up and down vote button in the index page before each question.

The AJAX voting has the exact same behavior as in the academy forum (<http://forums.academy.telerik.com>)

### Create KendoUI Grid administration for the posts in the Forum System (15 points)

* Create **Administration area** and work in it
  + In order to simplify testing don’t introduce any special roles.

1 point

* Using KendoUI create pageable, filterable and sortable grid that **displays** forum posts with the following columns: Id, Author name, Title, Content, CreatedOn, ModifiedOn and IsDeleted
  + Create corresponding controller and actions that provides the data and views that display the grid
  + You are free to use pure JavaScript or any MVC helpers

4 points

* Implement **deleting** entities via the grid

2 points

* Implement **editing** entities via the grid (editable fields Title and Content) with popup window

2 points

* + Implement **TinyMCE** editor for the **Content**

2 points

* + Create **Editor Template** for the forum post and use it in the grid

2 points

* Implement **creating** entities via the grid (the Author of the post will be the currently logged in user)

2 points

### Your Forum System code should have 0 StyleCop warnings (2 points)

Documentation rules does not count. They are disabled.

### Deploy your Forum System code to <https://appharbor.com/> (7 points)

1. Create a free account in <https://appharbor.com/>
2. Create an application and name it the way you want (but don’t use your name or username)
3. Add free “SQL Server Yocto” add-on for the database of your application
4. Update your Web.config with the proper SQL connection string (don’t send this connection string when you submit your exam, use it temporary until you deploy the system)
5. Deploy your application to AppHarbor using the built-in Git repository:  
   <http://support.appharbor.com/kb/getting-started/deploying-your-first-application-using-git>
6. Save the link of the application hosted in AppHarbor in the **README.md** file of the forum system folder when submitting your work

### Create two different types of advertisements for the EasyPTC system (10 points)

* Banners and text ads
* Design the database by your choice
* Implement functionality for displaying the ads on various pages through the application (text ads should be viewable all on a single page and banners should be chosen randomly on each page)
* Each click on advertisement should be counted in your application
* Administrators should be able to create, edit and delete advertisements

### Create Shopping Cart module for the EasyPTC system (15 points)

* Create different pricing plans combining the two advertisements
* Administrators should be able to add, edit and delete different pricing plans
* Normal users should be able to purchase one or more pricing plans
* Design a shopping cart module where users can add the pricing plans, preview their cart, select the quantity of the pricing plans and remove them from the cart
* After purchase, if the data are correct, just visualize "Thanks for your order"

### Create User Profile module for the EasyPTC system (10 points)

* It should show the number of clicked ads for each type
* It should show all the ordered pricing plans
* Users should be able to change their avatar. The client should display the image before the user send it to the server

## Evaluation Criteria

The evaluation criteria include: correct and complete fulfillment of the requirements; good technical design and appropriate use of technologies; high-quality code (correctness, readability, maintainability) and good application security.

## Other Terms

During the exam you are allowed to use any teaching materials, lectures, books, existing source code, and other paper or Internet resources. Direct or indirect communication with anybody in class or outside is forbidden. This includes but does not limit to technical conversations with other students, using mobile phones, chat software (Skype, ICQ, etc.), email, forum posts, etc.

You are not allowed to commit or create pull request to the given repositories. You are only allowed to work locally.

Delete NuGet packages along with /bin/ and /obj/ folders when submitting your work.

## Exam Duration

Students are allowed to work up to **9 hours**.