## Ms Web Tech Assignment – MVC Users and Roles

For this assignment, you will be working with users and roles. Allow users to add themselves to the system through *AccountController*, either as a locally-verified user or an externally-verified user (via Google, Yahoo, etc.), but disable (lock out) all new accounts. Only a member of the administrator role can enable users, delete them as intruders, reset passwords, create/delete roles and add users to roles.

### Project

1. Take your current ***XXOEC*** project or start with ***a2OEC*** from eConestoga.ca:
   1. Put your name & section in the ***\_layout*** footer
   2. Ensure the message variable from TempData displays, bold & red, on ***\_layout*** before the View’s page
2. Modify the connection string for *DefaultConnection* in ***appsettings.json*** to use “Server=.\\sqlexpress;…” or whatever instance of SQL Server you use at home, so you can reload your security database there.
3. Create a drop-down menu item labelled ***Security*** on the main menu, with sub-menu items ***User Maintenance*** and ***Role Maintenance.*** Only permit users of the ***administrators*** role see this item.
4. Register local users named ***fred@fred.com***, ***mary@fred.com*** and ***gord@fred.com***, all with the password ***Fred@fred1***.

### XXCropController

1. Lock up this controller so only signed-on users can execute its actions.
   1. Exempt the *Index* action so the public (anonymous users) can access it.
2. Modify the Index view so that only members of the ***administrators*** or ***staff*** roles can see the *Create, Edit* and *Delete* hyperlinks. Leave the *Details* hyperlink visible … I’ll use this to confirm the controller lock-up.

### Account Controller

1. Modify the post-back *Registration* action:
   1. Set *LockoutEnabled* to true on new users. This just permits users to be locked out: it doesn’t lock them out.
   2. Set the *LockoutEnd* to 20 years in the future. This locks them out until it’s null or in the past.
   3. If the create succeeds, do not let the registration process log the user in … return to the home page with the message “*account created, awaiting activation by the administrator”.*
2. Modify the post-back *Login* action to increment the user’s AccessFailedCount on each failed login attempt.
   1. Confirm new users are locked out (for at least 20 years)
   2. Ensure 5 failed login attempts locks the user out.
   3. It’s annoying that a failed logon attempt causes the return-to-page to be forgotten. See if you can fix this.

### User Maintenance Controller

1. Create ***XXUserMaintenanceController*** as an empty MVC controller, or one with empty CRUD actions. Require anyone using this controller to be signed on. Once users can be assigned roles, require users to be in the *administrators* role to access this controller.
2. For the ***Index*** action, create a view to list all users.
   1. For each user, show their username, e-mail, whether they’re locked out and whether they’re authenticated by an external service (or are locally authenticated). Remember: the lockout end date is universal time. I recommend a view Model, because you should include whether each user is an administrator.
   2. Sort them so the locked-out users are listed first, ordered by username, then the unlocked ones, also ordered by username.
   3. On each line of the user listing, add hyperlinks to:
      1. Lock/unlock the user (the hyperlink text should be the opposite of their current status)
      2. Delete the user (don’t show this link for the current user’s entry in the list)
      3. Reset the user’s password.
3. For the delete action (it has no view of its own and only the post-back action):
   1. Remove the user from all roles, delete them from the system and then return to the user listing.
      1. If the user is successfully deleted, display a message to that effect via TempData.
      2. If not, display a TempData message saying so, with the *innermost* exception’s message.
4. To reset the user’s password:
   1. Only members of the ***administrators*** role can reset passwords … return them to the Index listing with a message to that effect if they aren’t in that role.
   2. Present a view showing the user’s name, password input boxes for the new & confirm passwords, and a *Change Password* button.
   3. If the passwords do not match, display a message to that effect on the *ResetPassword* view.
   4. If the password is reset correctly, display a message to that effect on the listing of all users.
   5. If the password reset fails, display a message to that effect on the *ResetPassword* view, with the innermost exception’s message.
5. The lock/unlock action doesn’t need its own view:
   1. Unlock the user if locked, and vice-versa by resetting the user’s *AccessFailedCount* to zero and their *LockoutTimeEnd* to now … or 20 years in the future.
   2. Return to the listing of all users with a message stating the *userName* and that she/he is now locked or unlocked. If the update failed, say so, and include the exception’s innermost message.

### Role Maintenance Controller

1. Create ***XXRoleMaintenanceController*** as an empty controller or with empty CRUD actions. Require anyone using this controller to be signed on. Later, when you have users in the *administrators* role, require them to be in the ***administrators*** role.
2. Create an *Index* action and view to list all the current roles, ordered by role name:
   1. Don’t just drop a Create hyperlink at the top of the view:
      1. Add a textbox and submit button above the list to create a new role. Wrap these in a <form> tag targeting your post-back action to add a new role.
   2. On each line in the list, add hyperlinks to:
      1. Delete the role (don’t show this for the ***administrators*** role)
      2. Manage users in the role.
3. Create an action to add a new role:
   1. Stop the insert and return an error message to the role listing if:
      1. The proposed role name is already on file.
      2. The role name is null, empty or just blanks.
   2. In a try-catch, trim leading and trailing spaces before writing the new role to file.
      1. If the insert succeeds, put a message saying so in TempData and return to the role listing.
      2. If the insert fails, put a message to that effect in TempData with the innermost exception message and return to the rolelisting.
4. Create a view and an action to delete an existing role:
   1. If the selected role is ***administrators***, stop the process, return to the view listing all roles and display an error message that *administrators* cannot be deleted.
   2. If there are no users in the role, just delete it and return to the list of all roles with a message to that effect. Again … if the delete fails, blah blah innermost blah blah … you know the drill.
   3. Otherwise, list the username of all members in the role along with a button to confirm the delete. A hyperlink back to the role listing would be nice.
      1. If the delete is successful, return to the roles listing with a message that the role was deleted (naming the role).
      2. If not successful, return to the delete view and display the innermost exception’s or IdentityResult’s message.
5. To manage users in a role:
   1. List all users in the given role: userName and e-mail at least
   2. Add a hyperlink & action on each line to remove individual users from the role
      1. For the ***administrators*** role, don’t show this hyperlink for the current user’s entry.
   3. Add a drop-down showing all users *not* in the role, with a button to add the selected user to the role.
   4. Display a message if the add or remove works or, if they fail, show a message with the innermost exception’s message or the IdentityResult’s message.

### Use it:

1. Create the roles ***staff*** and ***administrators***.
2. Add ***fred*** to the *administrators* and *staff* roles.
3. Add ***mary*** to the *staff* role.
4. Do not add ***gord*** to any roles.
5. Restrict all actions of ***User Maintenance*** and ***Role Maintenance*** to members of the ***administrators*** role.
6. Export your security database (schema and data) to an SQL script file inside your solution folder

### Hand In

1. Zip and upload your project folder to the [Drop-Box](https://eConestoga.ca/) (Course Tools 🡪 Assignments) for this assignment.
2. Hand in [the marking sheet](https://eConestoga.ca/) with your name on it, in class or through the slot in your instructor’s office door.