SYSTEMS DESIGN / CAPSTONE PROJECT
MIS 413

Client Checkpoint #7
Login / Cookies

A working demo of this project may be found at: **http://miscapstone.uncw.edu/413examples**

Goal: Provide a means for users to authenticate themselves and determine their roles and permissions.

Part A: SQL Database Stored Procedure

1. Write a new stored procedure, name it: xxx**personLogin (each team member should create this stored procedure, where xxx is your initials)**

2. If the user's email and password are correct - this stored procedure should return the personID, fn, ln, email, roleID. (sort by roleId)
   a. Open your validRolesTable (Edit 200 rows) and check/modify the admin role to be 1.

3. If the user's email and/or password are invalid, no data should be returned

4. We will build this stored procedure in class.

Part B: Fix any errors noted to your website from Project 6

5. Point value deductions will be doubled if you do not fix items previously noted as errors or suggestions for clearer pages.

Part C: Create a Log In Page.

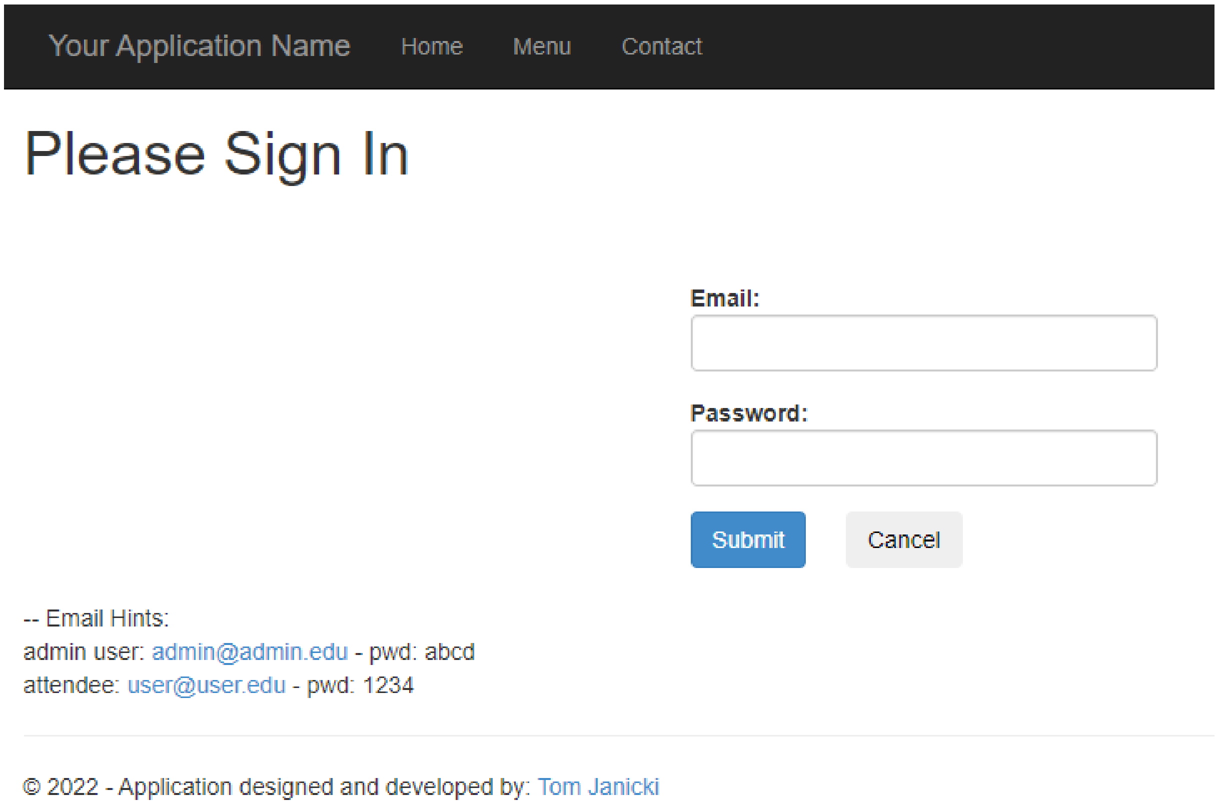
6. Create a login.aspx page at the root level (right click on the top project name in bold, and then add item– new webpage with master page.
   a. Name this page login.
   b. Use Site.Master
   c. Don't forget to change the title tag once the page is started

7. Modify the page as shown in the figure on the next page.
   a. Hint: Copy from your profile.aspx page from and including the H1 tag down to the </asp: panel> and place on your login page after the Content Tag and before the /Content Tag.
   b. Remove the firstName, lastName and role objects from each of their <form-group> to the ending </div> for that object.
   c. Revise the property to make the _pnlAdd to be visible
   d. Delete the Green Add a New Person Button
   e. For the two remaining buttons, removed the two onClick events (`OnClick="_submit_Click"`), found on the HTML view for those buttons.

8. Your remaining text boxes should be email and _password and two buttons

9. **To help with grading add two labels below your login information and change the text property to provide easy access during development.** The text properties should contain a valid email and password for the admin and for a typical student as in:

   Admin user: xxx1234@uncw.edu - pwd: abcd
   Student user: yyy2222@uncw.edu; pwd: 1234



Please Sign In

-- Email Hints:
admin user: admin@admin.edu - pwd: abcd
attendee: user@user.edu - pwd: 1234

© 2022 - Application designed and developed by: Tom Janicki

10. Double Click the Cancel button and write the code to clear the email text box and the _message label (you will not need to clear the password text box).

11. Save and test, especially the error checking, did you check for a valid email address?

Part D: Add a Login option to your top menu bar in your site.master.

12. On your site.master add the following code to put a login/logout option to your top menu bar after the *</ul> following the Contact Link* . The login/logout will toggle based on if we have signed into the system.

```
    <li><a runat="server" href="~/Contact">Contact</a></li>
    </ul>  **** do not type these two lines in purple they only are here to help
you find where to insert the new code)
```

```
(remember to copy and paste to notepad before pasting into your VS project)

    <ul class="nav navbar-nav navbar-right">
        <li>
            <asp:HyperLink ID="_login" runat="server" NavigateUrl
="~/login">Login</asp:HyperLink>
            <asp:HyperLink ID="_logout" runat="server" NavigateUrl ="~/logout"
Visible="false" >Logout</asp:HyperLink>
        </li>
     </ul>
```

13. Save and test your page, does the Login appear on the right side of the menu bar?

Part E: Move your Menu.aspx page to be only accessible to users who have been properly authenticated via the Login Page.

14. Slide the menu.aspx page into your USER folder

15. On your site.master change the link for the menu to be "~/user/menu.aspx" Hint: Look for the <li> commands for Home, Menu, Contact

16. Back in your USER folder, add a web.config file that will block users who have not from accessing anything in this folder.
    a. Right click on the USER folder
    b. Add a new Item
    c. Add a web.config file (keep the name)

    Modify the code as shown with the lines in bold

```
<configuration>
     <system.web>
          <authorization>
               <deny users="?"/>
          </authorization>
     </system.web>
</configuration>
```

Part F – Change the master web config to get ready to save a cookie on the client machine.

17. Open the root level web.config and find the tag </system.web> and immediately before it place this code (this will redirect the person to the login page if they have not signed in.

```
<authentication mode="Forms" >
   <forms loginUrl="~/LogIn.aspx" timeout="2880" cookieless="UseCookies" />
 </authentication>
 <customErrors mode="Off">
    </customErrors>
```

Part G: Back on your Login Page (C# option), we need to tell the page we want to open a SQL database without using a wizard, thus we will add some library items to the page that will give us access to SQL commands.

18. Add the following statements to the top of your C# page, after the other USING statements:

```
using System.Data.SqlClient;
using System.Data;
using System.Web.Security;
```

Part H: Coding under the Submit Button

19. Start your subroutine with checking that the validation controls worked and declaring a variable that will track if the user's info is correct or incorrect. Items shown in brown should be substituted with your particular project's storedprocedure, connection string.

```
  if (IsValid)
        {
            Boolean isOK = false;
            string dbConn =
System.Configuration.ConfigurationManager.ConnectionStrings["mis413ConnectionString
"].ConnectionString;
            //build a connection to the database
            SqlConnection conn = new SqlConnection(dbConn);
            //use the above connection to execute a particular stored procedure
//[substitue your personInsert stored procedure name]
            using (SqlCommand cmd = new SqlCommand("[tnjPersonLogin]", conn))
            {
                cmd.CommandType = CommandType.StoredProcedure;
         /build the parameters (input items) that the stored procedures needs
                cmd.Parameters.AddWithValue("@email", this._email.Text);
                cmd.Parameters.AddWithValue("@pwd", this._password.Text);
                // open the database and actually run the stored procedure, also
//catch any errors and display them in your _message label
                try
                {
                    conn.Open();
                    SqlDataReader dtrReader = cmd.ExecuteReader();
                    // see if any data was returned from the stored procedure, if
//no data than not a valid user or password
                    if (dtrReader.HasRows)
                    {
                        // read the first record
                        dtrReader.Read();

     //specialized cookie code goes here, what should you do if the login is valid?
// hold this area to insert cookie information
// *****


// *******

                        isOK = true;
                        //set authentication to true and save the user's name
                        string userName = (string)(dtrReader["firstname"]) + " " +
(string)(dtrReader["lastname"]);
                        FormsAuthentication.RedirectFromLoginPage(userName, true);
                    }
                    else
                    {
                        //no data was returned from the stored procedure
                        this._message.Text = "Invalid username or password.";
                    }
                }
                //if there are any errors with the store procedure, display them /
                catch (SqlException ex)
                {
                    this._message.Text = "Error from SQL " + ex.Message;
                }

                //this ends the using
            }

            if (isOK)
            {
                Response.Redirect("~/user/menu.aspx");
            }

        }

    }
```

20. At this point test good and bad combinations of passwords/emails, do you get the proper message. Don't forget to clear _message.Text if you click Cancel.

21. Once you have verified that the user email/pwd are valid; you want to build a cookie to 'store' that the person has logged into the system and to also save their ID for use on following pages. Place the following code immediately after the dtrReader.Read command inside the try/catch under the Submit(login) button

```
                    HttpCookie aCookie = new HttpCookie("userInfo");
                    aCookie.Values["firstname"] = (string) dtrReader["firstname"];
                    aCookie.Values["lastname"] = (string)dtrReader["lastname"];
                    aCookie.Values["email"] = (string)dtrReader["email"];
                    aCookie.Values["personId"] = Convert.ToString
(dtrReader["personId"]);
                     aCookie.Values["roleId"] = Convert.ToString (dtrReader["roleId"]);
                    aCookie.Expires = DateTime.Now.AddMinutes(20);
//this actually writes the cookie on the users machine on the postback to the client
                    Response.Cookies.Add(aCookie);
                    isOK = true;

//set authentication to true and save the user's name
                 string userName = (string)(dtrReader["firstname"]) + " " +
                 (string)(dtrReader["lastname"]);
                FormsAuthentication.RedirectFromLoginPage(userName, true);
```

Part I: Toggle the Login in the top Navigation to Logoff

22. Open the site.master page,

23. Add a using for the security authentication, to the top using statements (C# page)
    a. `using System.Web.Security;`

24. Based on if the user has logged in, toggle Login/Logout, we are checking the authentication 'ticket', place the following code in the page_load method on the C# page for site.master.

```
//test to see if the person is logged in, we will check
 if (Request.IsAuthenticated)
 {
     this._login.Visible = false;
     this._logout.Visible = true;
 }
 else
 {
     this._login.Visible = true;
     this._logout.Visible = false;
 }
```

25. Finally, build a **logout** page at root level, that will delete the authentication ticket and also kill the cookie on the user machine. This page will never appear to the user, but once the cookie is killed send back to the default page.

26. Add a: using System.Web.Security to the using statement group (on the C# page)

27. Here is the code to delete the login ticket and cookie, place in the page_load method

```
FormsAuthentication.SignOut();
Session.Clear();
Session.Abandon();

// Clear cookie
if (Request.Cookies["userInfo"] != null)
{
    HttpCookie myCookie = new HttpCookie("userInfo");
    myCookie.Expires = DateTime.Now.AddDays(-1d);
    Response.Cookies.Add(myCookie);
}
Response.Redirect("~/default");
```

Part J: Test

1. **Please TEST your project by entering the public link as in: http://misCapstone.uncw.edu/S23FolderName, the project should load, and click the MENU in the navigation menu to see your Menu page.**

2. When you are ready for grading, and you have tested your project by using a https://miscapston address, go to Canvas and post your full http://miscapstone...... Link in the Client Checkpoint #7 homework box. Make sure you include in the text box a valid admin and a valid student login credentials.