Software Security
Assignment 2 – Report

MS21910272 – DLC Kalpana
MS21913488 – P.K.P. Marindagoda

Setting up instructions.

To connect to authentication providers like Google and Facebook via **public OAuth Server**, you will need to set up IIS-Express to use SSL. In here I have used **Google with .net web MVC Framework**. It's important to keep using SSL after login and not drop back to HTTP, your login cookie is just as secret as your username and password, and without using SSL you're sending it in clear-text across the wire. Besides, you've already taken the time to perform the handshake and secure the channel (which is the bulk of what makes HTTPS slower than HTTP) before the MVC pipeline is run, so redirecting back to HTTP after you're logged in won't make the current request or future requests much faster.

1. In **Solution Explorer**, click the **MvcAuth** project.
2. Hit the F4 key to show the project properties. Alternatively, from the **View** menu you can select **Properties Window**.
3. Change **SSL Enabled** to True.
4. Copy the SSL URL (which will be https://localhost:44346/ unless you've created other SSL projects).
5. In **Solution Explorer**, right click the **MvcAuth** project and select **Properties**.
6. Select the **Web** tab, and then paste the SSL URL into the **Project Url** box. Save the file (Ctl+S). You will need this URL to configure Facebook and Google authentication apps.
7. Add the RequireHttps attribute to the `Home` controller to require all requests must use HTTPS. A more secure approach is to add the RequireHttps filter to the application. See the section "Protect the Application with SSL and the Authorize Attribute" in my report of configuration and Create an ASP.NET MVC app with auth and SQL DB and deploy to Azure App Service. A portion of the Home controller is shown below.

```
[RequireHttps]
public class HomeController : Controller
{
   public ActionResult Index()
   {      return View(); }
```

8. Press CTRL+F5 to run the application. If you've installed the certificate in the past, you can skip the rest of this section and jump to Creating a Google app for OAuth 2 and connecting the app to the project, otherwise, follow the instructions to trust the self-signed certificate that IIS Express has generated.
9. Read the **Security Warning** dialog and then click **Yes** if you want to install the certificate representing localhost.
10. IE shows the *Home* page and there are no SSL warnings.
11. Google Chrome also accepts the certificate and will show HTTPS content without a warning. Firefox uses its own certificate store, so it will display a warning. For our application you can safely click **I Understand the Risks**.

Creating a Google app for OAuth 2 and connecting the app to the project

1. Navigate to the **Google Developers Console**.
2. If you haven't created a project before, select **Credentials** in the left tab, and then select **Create**.
3. In the left tab, click **Credentials**.
4. Click **Create credentials** then **OAuth client ID**.
    1. In the **Create Client ID** dialog, keep the default **Web application** for the application type.
    2. Set the **Authorized JavaScript** origins to the SSL URL you used above (`https://localhost:44300/` unless you've created other SSL projects)
    3. Set the **Authorized redirect URI** to: `https://localhost:44300/signin-google`
5. Click the OAuth Consent screen menu item, then set your email address and product name. When you have completed the form click **Save**.
6. Click the Library menu item, search **Google+ API**, click on it then press Enable.
7. From the Google APIs API Manager, visit the **Credentials** tab to obtain the **Client ID**. Download to save a JSON file with application secrets. Copied the **ClientId** and **ClientSecret** into the `UseGoogleAuthentication` method found in the *Startup.Auth.cs* file in the *App_Start* folder. The **ClientId** and **ClientSecret**.

```
public void ConfigureAuth(IAppBuilder app)
{
        app.CreatePerOwinContext(ApplicationDbContext.Create);
```

```
app.CreatePerOwinContext<ApplicationUserManager>(ApplicationUserManager.Create)
;



    app.UseCookieAuthentication(new CookieAuthenticationOptions
    {
        AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
        LoginPath = new PathString("/Account/Login"),
        Provider = new CookieAuthenticationProvider
        {
            OnValidateIdentity =
SecurityStampValidator.OnValidateIdentity<ApplicationUserManager,
ApplicationUser>(
                validateInterval: TimeSpan.FromMinutes(20),
                regenerateIdentity: (manager, user) =>
user.GenerateUserIdentityAsync(manager))
        }
    });

    app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);


    app.UseGoogleAuthentication(
         clientId: "275-356.apps.googleusercontent.com",
         clientSecret: "0807538970000");
}
```

8. Press **CTRL+F5** to build and run the application. Click the **Log in** link.
9. Under **Use another service to log in**, click **Google**.

   If you miss any of the steps above you will get a HTTP 401 error. Recheck your steps above. If you miss a required setting (for example **product name**), add the missing item and save; it can take a few minutes for authentication to work.

10. You will be redirected to the Google site where you will enter your credentials.
11. After you enter your credentials, you will be prompted to give permissions to the web application you just created:
12. Click **Accept**. You will now be redirected back to the **Register** page of the MvcAuth application where you can register your Google account. You have the option of changing the local email registration name used for your Gmail account, but you generally want to keep the default email alias (that is, the one you used for authentication). Click **Register**.