**Preparation**

Exercises will refer to my week 9 GitHub repo which is at [**https://github.com/oit-gaden/Web-Development-2020-Winter/tree/master/Week9**](https://github.com/oit-gaden/Web-Development-2020-Winter/tree/master/Week9)

Copy the contents of your week 8 lab folder to your week 9 lab folder.

Note: Remove the .vscode folders under week 9 root, webapi and webapp folders.  Students have had issues with VSCode not knowing where code is.  This will mean VSCode will create those folders again along with the launch.json and tasks.json files when you run the webapi.

Go to [**https://auth0.com/ (Links to an external site.)**](https://auth0.com/) and create an account.  You can use your GitHub account to login or one of the other options if you want.  If you are logged in to Auth0 while viewing the tutorials you can see settings belonging to your account rather than generic ones.  There is a copy symbol in the upper right corner of the code/config listings that provide a simple way to copy the contents of the listings.

Include screenshots required in a single Word/Wordpad document located in your week 9 lab folder.

**Exercise 1 - Secure your frontend webapp**

1. Follow the tutorial directions at: [**https://auth0.com/docs/quickstart/spa/vuejs/01-login (Links to an external site.)**](https://auth0.com/docs/quickstart/spa/vuejs/01-login)with the modifications listed in the following steps.
2. Create a new "application" in auth0 rather than rely upon the default one created when you created your auth0 account.
3. Use port 8080 rather than 3000 since 8080 is the default port when you run the command **npm run serve.**
4. Skip the section "Create a Sample Application" since you already have a Vue application.
5. Add the login and logout buttons/code to NavBar.vue and not Home.vue.  Also, like the login and logout buttons, hide and show the other menu items based upon whether or not a user is logged in.
6. Since you already have the student list page you can skip the part of the tutorial that adds a profile page.  You can do this if you want.
7. Secure your student list page as described under the section: "Secure the Profile Page".
8. Use the usual command (no port number) to start the Web frontend application:  
     
   **npm run serve**
9. When you start your Web application and click on the login button in your webapp you should be redirected to the auth0 login page.  You can create a new test user by "signing up" or using a Google login.  Once logged in you should be able to use your app as before.  You might encounter some issues with your app working completely correct.  I ran into some.  Don't worry about that.  As long as you can login and logout and see the menu bar items hidden/shown based upon if you are logged in or not.

**Exercise 2 - Secure your backend webapi**

1. Follow the tutorial directions at: [**https://auth0.com/blog/how-to-build-and-secure-web-apis-with-aspnet-core-3/  (Links to an external site.)**](https://auth0.com/blog/how-to-build-and-secure-web-apis-with-aspnet-core-3/)with the modifications listed in the following steps.
2. You can run your webapi in a console/terminal window rather than in VSCode if you want by running  the command (in the webapi folder):  
     
   **dotnet run**
3. Skip all the way down to the section titled: "Securing the API with Auth0".  If you want to learn more about building APIs using .NET Core then I would recommend reading and perhaps trying the  part of the tutorial prior to this section to learn some of the basics we did not cover in class/labs.  While not complete, it does give a basic intro to building APIs using .NET Core.
4. In the sub-section titled "Configure the Web API" it says to replace the contents.  You have important configuration (e.g. database) so you don't want to replace it.  Just add the auth0 related configuration as in their example.  
     
   "Auth0": {"Domain": "YOUR\_AUTH0\_DOMAIN", "Audience": "YOUR\_UNIQUE\_IDENTIFIER" }
5. In the sub-section titled "Integrate with Auth0" it says to replace the ConfgureServices() method.  Again, you have important code you added there previously.  Just add the auth0 related code as in their example.  
     
   services.AddAuthentication(options =>   
   {  
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;       options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme; }).AddJwtBearer(options =>  
   {  
    options.Authority = $"https://{Configuration["Auth0:Domain"]}/";  
    options.Audience = Configuration["Auth0:Audience"];  
   });
6. Don't replace the Confgure() method in Startup.cs.  Just add the following two lines of code as in their example:  
     
   app.UseAuthentication();  
   app.UseAuthorization();
7. Just secure the GET all students endpoint as described in the section titled "Securing the Endpoints".
8. Use Postman instead of curl to test the secured GET all students API endpoint.  You should get a HTTP status code of 401.  **Include a screenshot of Postman with the URL you used and the resulting status code returned.**
9. Once you get a JWT token from the auth0 dashboard test page, as described in the tutorial, you can test your API again with the token using Postman to make sure it authorizes correctly.  Select the "Authorization" tab under the method and URL entry.  Select "Bearer Token" under the "Type" dropdown menu.  Enter the token from the auth0 dashboard in the token field to the right of the type selection.  **Include a screenshot of Postman with the URL you used, the token entry and the resulting status code returned.**

**Exercise 3 - Give  your frontend webapp access to your secure backend webapi**

1. Follow the tutorial directions at: [**https://auth0.com/docs/quickstart/spa/vuejs/02-calling-an-api#create-an-api (Links to an external site.)**](https://auth0.com/docs/quickstart/spa/vuejs/02-calling-an-api#create-an-api)[**(Links to an external site.)**](https://auth0.com/blog/how-to-build-and-secure-web-apis-with-aspnet-core-3/)with the modifications listed in the following steps.
2. Skip down to the section titled "Set up a proxy to the backend API".  Don't restart the application as stated in this section.  Just add the setup of the proxy.
3. In the section titled "Call the API Using an Access Token" you just need to add the header token authentication to your existing GET all students REST call.  You can look at their example or see the example in my repository.
4. Start your webapp and navigate to the students list page.  It display the list of students as before with the data coming from your secure webapi.
5. **No** requirement this week to submit Docker container images.