**Creating a Role and Assigning a User to the Role (Admin, in this case)**

1. Create a new asp.net core web application as per normal, using .NET Core 2.0. (Note: in Asp.Net Core 2.1 or later, Identity should be scaffolded into the project. See link [Scaffold Identity in ASP.NET Core projects](https://docs.microsoft.com/en-us/aspnet/core/security/authentication/scaffold-identity?view=aspnetcore-2.2) ). Back to creating a new Core Project: Set ‘Authentication’ to ‘Individual user account’. Select Asp.Net MVC option as opposed to ‘Empty’ project, say. Click ok and let the scaffolding create the basic Models, Views and Controllers.
2. Shorten the database name in appsettings.json, as per the usual, by getting rid of additional string characters in the Db name beyond the project name (Append DB to project name, to improve self-descriptiveness of the database name). Save appsettings.json.
3. If you run the application you might get a message that a migration is pending. If so, apply migration, refresh the html page where you were asked to ‘Apply Migration’.
4. Using the dropdown menu (VS 2019) on the top right-hand corner of the home page register a user, A, say.
5. View ‘SQL Server Object Explorer’ in VS; refresh and find the relevant database, looking for database name in 2 above. ‘View Data’ on AspNetUsers table in this database, you will see the registered user here. Whereas ‘View Data’ on AspNetUsersRoles shows no user(s).
6. Create Role and Assign User for Role:
   1. Add method below to Startup.cs to create ‘Admin’ Role:

//Method to create User Role, added as per "Getting started with Asp.Net Core Identity and Role Management" notes, MSDN.

private async Task CreateUserRoles(IServiceProvider serviceProvider)

{

var RoleManager = serviceProvider.GetRequiredService<RoleManager<IdentityRole>>();

var UserManager = serviceProvider.GetRequiredService<UserManager<ApplicationUser>>();

IdentityResult roleResult;

var roleCheck = await RoleManager.RoleExistsAsync("Admin");

if (!roleCheck)

{

//create the roles and seed them to the database

roleResult = await RoleManager.CreateAsync(new IdentityRole("Admin"));

}

//Identify this new role by specific email below (and password)

//Now, create AdminController, and View; set Authorisation (see AdminController's Index())

ApplicationUser user = await UserManager.FindByEmailAsync("admin@salon.com"); //PWD: Admin\_123

await UserManager.AddToRoleAsync(user, "Admin");

}

1. Make a call to this method, CreateUserRoles(), in the Configure() function of Startup.cs. Remember to add ‘IServiceProvider services’ as the third input parameter to Configure() method, as seen below. (Note: if this exercise necessitate adding migration and updating database, delay the function call until after the migration and update).

...

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

template: "{controller=Home}/{action=Index}/{id?}");

});

**CreateUserRoles(services).Wait();**

}

1. Run the application and check that AspNetUserRoles table now contain this newly added user, with email address as shown in CreateUserRoles() method.
2. Now, create Admin Page and set authorisation:
   1. Create a Controller named AdminController, by right-hand clicking on Controller folder and selecting ‘MVC Controller – Empty’.
   2. Change the name defaultController in the popup box to AdminController.
   3. Select Index() method name inside the controller and right-hand click to ‘Add View’. Click ‘Add’ to create the Empty View. Check in Solution Explorer for the View.
   4. The View file opens; change page name from “Index” to “Admin Page”: ViewData["Title"] = "Admin Page";
   5. In the Shared layout, \_Layout.cshtml, add the list item for Admin page as follows:

<li><**a** **asp-area**="" **asp-controller**="Home" **asp-action**="Contact">Contact</**a**></li

<li><**a** **asp-area**="" **asp-controller**="Admin" **asp-action**="Index">Admin Page</**a**></li>

f. To allow only Admin to have access to this page set the Authorisation as follows in the AdminController:

[Authorize(Roles = "Admin")]

public IActionResult Index()

1. Run the application and Register an admin user with email [admin@salon.com](mailto:admin@salon.com) and password, say, Admin\_123. Note that only a user with this email and associated password can see the ‘Admin Page’. Pages like this can be used to add staff, change prices, etc. Note 2: if this user login is spoiled by, say, forgotten password, you have to:
   1. Go to ‘View’ -> ‘SQL Server Object Explorer’ and open inspect relevant database, delete this user from AspNetUserRoles, then from AspNetUsers
   2. In Startup.cs method CreateUserRole() change email address for Admin to one you that you prefer for the Admin user.
   3. Register a user with this email; note down password. You have a new Admin use.

End.