

CMPS 3830 Spring 2026

Setup ASP.Net Identity Core

Requirements

- Configure your project to use asp.net core identity with cookie authentication
 - See:
<https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-10.0&tabs=visual-studio#configure-identity-services>
 - Within Program.cs:
 - **You need to call AddIdentity<User,Role> and AddEntityFrameworkStores**
 - Hint:
<https://www.nuget.org/packages/Microsoft.AspNetCore.Identity.EntityFrameworkCore/>
 - **You need to call UseAuthentication and UseAuthorization**
 - Reminder: the order of methods in Program.cs matter
 - The learn link above has extra things - it's recommended you only call the needed methods
 - Do not add scaffolding / razor pages / etc - we ONLY need asp.net core to be an API (You *can* use these as examples of how to do some things, just do so outside of your git repository)
- Add User, Role, and UserRole entities
 - User and Role have a many-to-many relationship
 - UserRole is the joining entity between User and Role
 - You must have navigation properties between them
 - see:
<https://docs.microsoft.com/en-us/ef/core/modeling/relationships?tabs=fluent-api%2Cfluent-api-simple-key%2Csimple-key#definition-of-terms>
 - Put these entities under Features in folders that follow the existing convention
 - An important gotcha when adding the navigation properties:
<https://learn.microsoft.com/en-us/aspnet/core/migration/1x-to-2x/identity-2x?view=aspnetcore-10.0#add-identityuser-poco-navigation-properties>

```

19 .....protected override void OnModelCreating(ModelBuilder builder)
20 .....{
21 .....    base.OnModelCreating(builder);
22 .....
23 .....    var userRoleBuilder = builder.Entity<UserRole>();
24 .....
25 .....    userRoleBuilder.HasKey(x => new { x.UserId, x.RoleId });
26 .....
27 .....    userRoleBuilder.HasOne( navigationExpression: x => x.Role)
28 .....        .WithMany( navigationExpression: x => x.Users)
29 .....        .HasForeignKey(x => x.RoleId);
30 .....
31 .....    userRoleBuilder.HasOne( navigationExpression: x => x.User)
32 .....        .WithMany( navigationExpression: x => x.Roles)
33 .....        .HasForeignKey(x => x.UserId);
34 .....}

```

- Have your DataContext inherit from:
 - IdentityDbContext<User, Role, int, IdentityUserClaim<int>, UserRole, IdentityUserLogin<int>, IdentityRoleClaim<int>, IdentityUserToken<int>>
 - Adjust the domain, modify seed functions, and add migrations as needed
 - See: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/customize-identity-model?view=aspnetcore-10.0#model-generic-types>
- Add one or more migrations as needed for all domain changes
 - **Do not delete the original migration**
- No user should share the same username
- Seed the following roles:
 - Admin
 - User
 - **Hint:** You'll need to call "CreateAsync" on the "RoleManager<Role>" to do this
- Seed the following users with the password "Password123!":
 - Username: galkadi - role: Admin
 - Username: bob - role: User
 - Username: sue - role: User
 - **Hint:** You'll need to call "CreateAsync" on the "UserManager<User>" to do this
- If a user is not logged in at all and attempts to access a resource that has some authorization tied to it then return an HTTP 401
- To get the HTTP 403 and 401 tests to pass you need to use **ConfigureApplicationCookie** in your Program.cs
 - **Hint:** options.Events.OnRedirectToLogin
- Note: if you have already setup the react web side then you will need to add authorization to the controllers like so:

