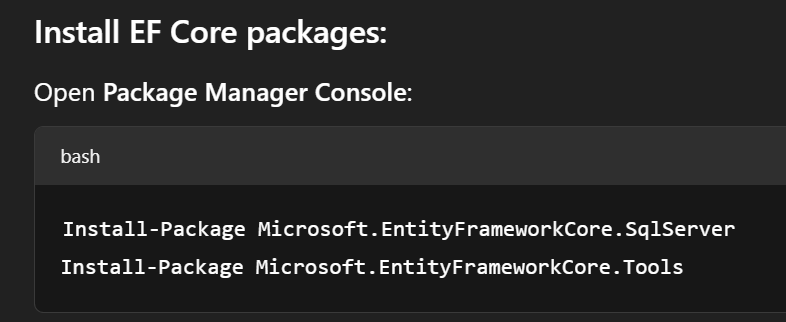
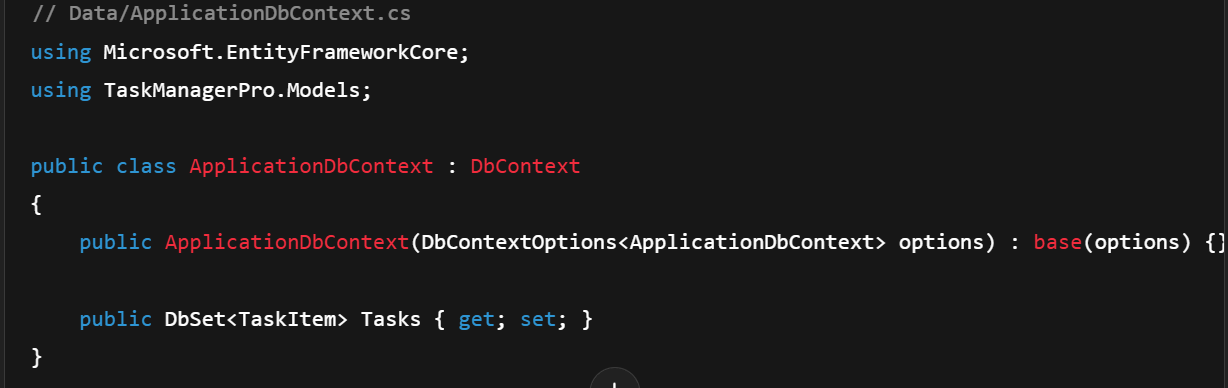
Phase 1: Initial Setup

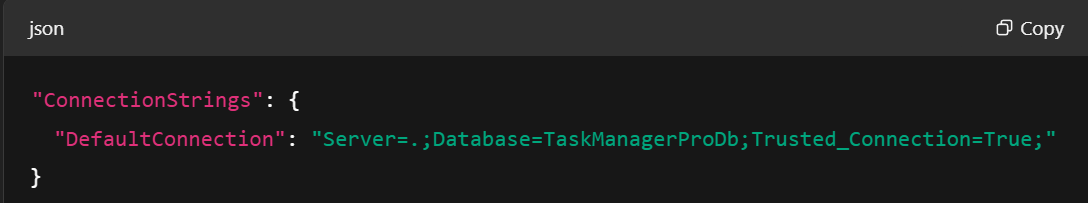
1. Install EFCore packages



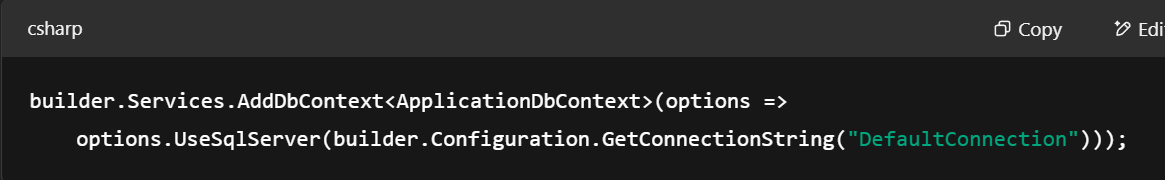
1. Create folder Data and add Database Context



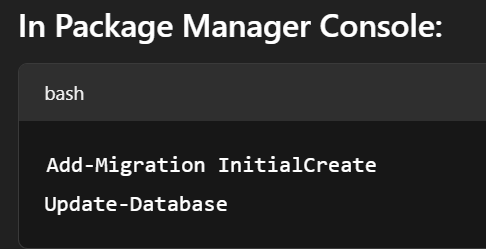
1. Add connection string in appsettings.json



Register your DbContext in Program.cs

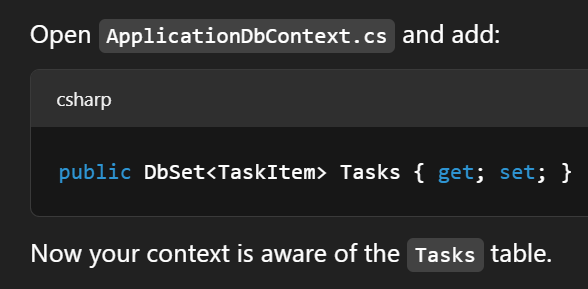


1. Add Migration and create Db



Phase 2: CRUD

1. Create TaskItem Model
2. Register TaskItem in ApplicationDbContext



1. Add Migration for TaskItem Model

1. Add Controller and Views through scaffolding



Right-click the Controllers folder → Add → New Scaffolded Item

Choose ‘MVC Controller with views, using Entity Framework’

Pick your TaskItem model and ApplicationDbContext

Controllers and Views will be created.

Now, the website will have the basic CRUD functionalities.

Now, when we run the app, if you want it to open TaskItems/Index by default, instead of you manually typing it, set default routing in program.cs. Change default controller to TaskItems from Home Controller.

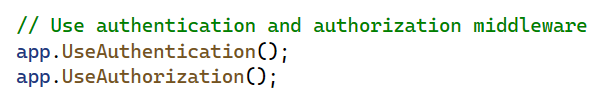
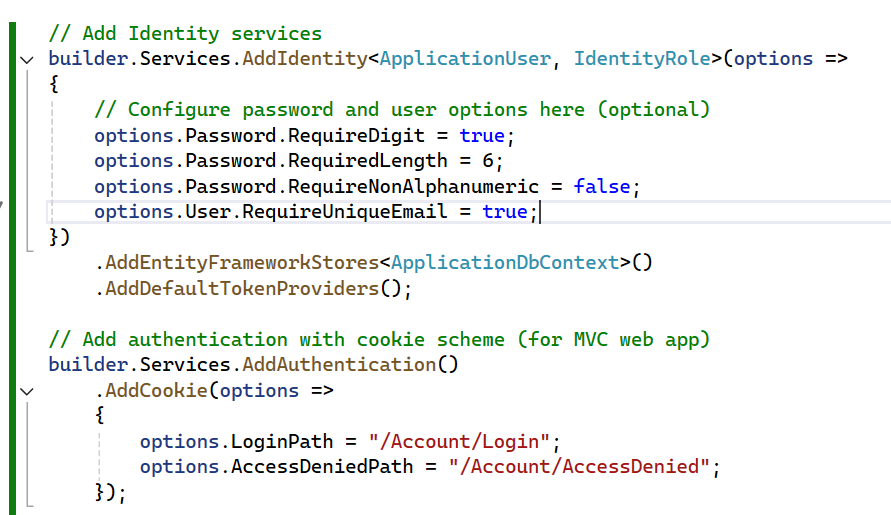
Phase 3: User Authentication and Role based Access Control

1. Install req Nuget Packages

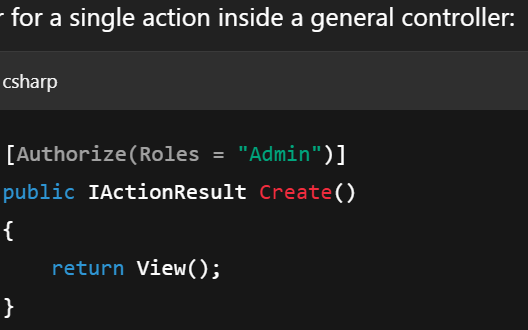
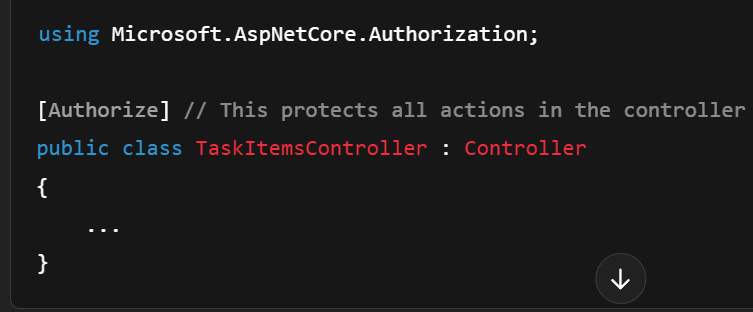
Install-Package Microsoft.AspNetCore.Identity.EntityFrameworkCore -Version 8.0.0

Install-Package Microsoft.AspNetCore.Authentication.JwtBearer -Version 8.0.0

1. Open your existing DbContext file (ApplicationDbContext.cs) and update it to inherit from IdentityDbContext. Also, create an ApplicationUser class.
2. Add Identity services and configure authentication in Program.cs.



1. Add a new controller AccountController.cs
2. Add a new folder ViewModels and create these: RegisterViewModel.cs, LoginViewModel.cs
3. Create Views for Register, Login, and AccessDenied inside new Folder Account: Register.cshtml, Login.cshtml, AccessDenied.cshtml
4. Seed Roles and Admin User – Create a static helper class to seed roles and an admin user on app startup. In Data folder create the AppDbInitializer.cs file. Call this in Program.cs right after building the app. Make sure Program.cs’s Main method is marked async.
5. Now we have to ensure – Only logged-in users can access certain pages (like task list), Only Admins can access Admin-only areas, Display role-based UI (e.g., show “Admin Panel” only to Admins)
6. Open TaskItemsController.cs, and add –



1. To render UI conditionally by role: Open Views/Shared/\_Layout.cshtml and replace the code in

<ul class="navbar-nav">