**Simple Register and Login Using .net6 web Api, JWT, SQLite, Hashing and Salting**

**Setting up:**

* Install latest .net sdk ([.NET SDKs downloads for Visual Studio (microsoft.com)](https://dotnet.microsoft.com/en-us/download/visual-studio-sdks))
* Use Visual Studio Code([Visual Studio Code - Code Editing. Redefined](https://code.visualstudio.com/)) or Visual Studio – here I’m using Visual Studio Code
* For this project, I'm using Command Prompt.

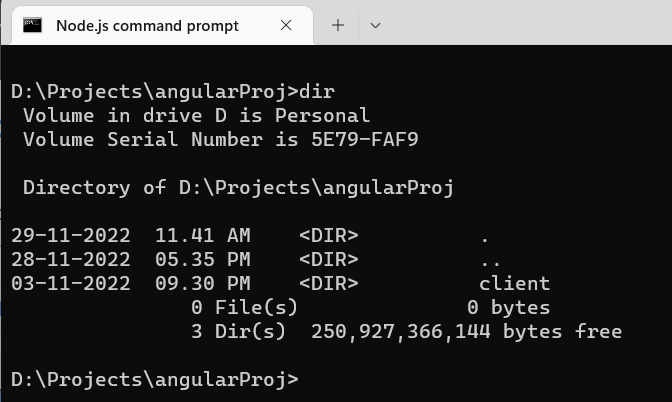
**Introduction**

Please refer to this document for instructions on how to create your own login method other than the inbuilt login function (I am not using the inbuilt identity component).

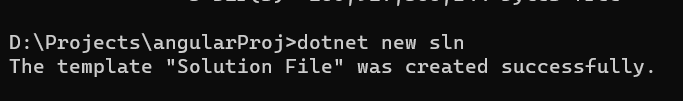
**Project Creation:**

Here I have created a folder called "angularPro," and I have added an anular project to that. We'll go over the.net API components rather than the Angular components; I'll add a new document about configuring the API service in an Angular project soon.

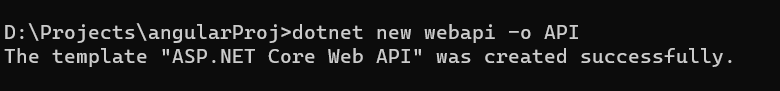
My existing folder structure below



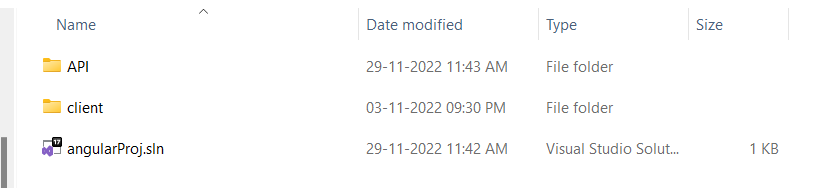
First, we need to create a solution file, and within that, we have to add the API project. Check out the screencaps below.



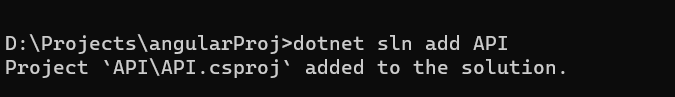
I am adding WEB API project



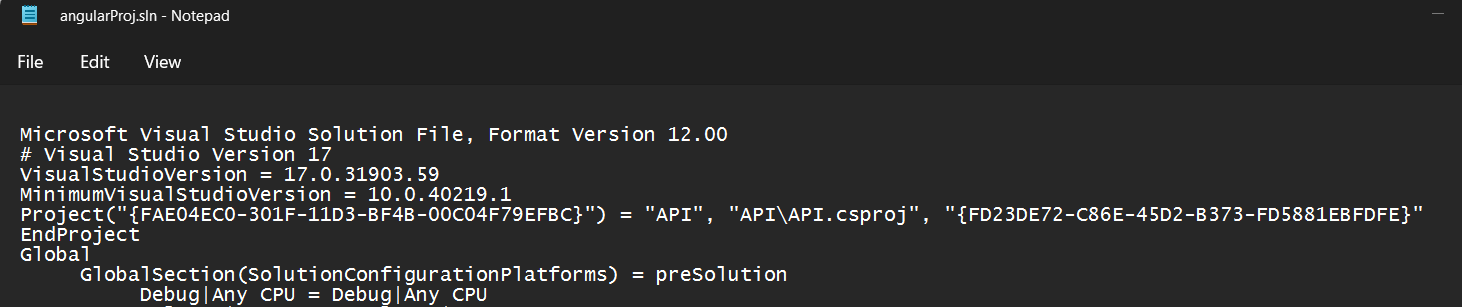
I have added the API project, Check out the screencaps below.



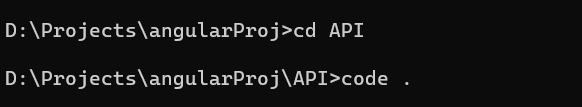
Here I am adding the API project to the solution.



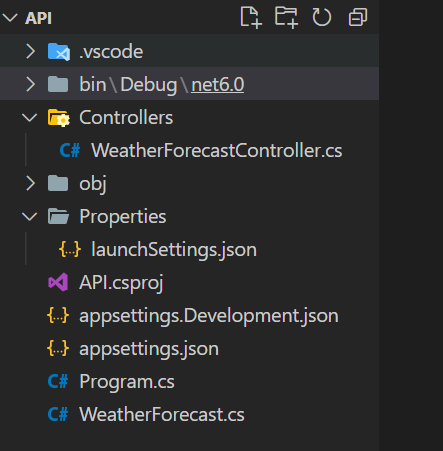
After adding, my .Sln file look like



Open the API project by using below command, I am opening the project in Visual Studio code

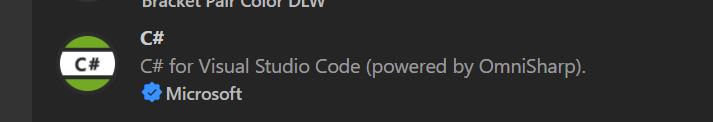


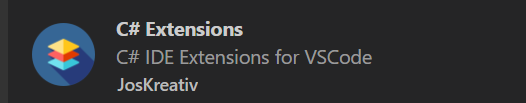
The folder structure in Visual Studio code is shown below.

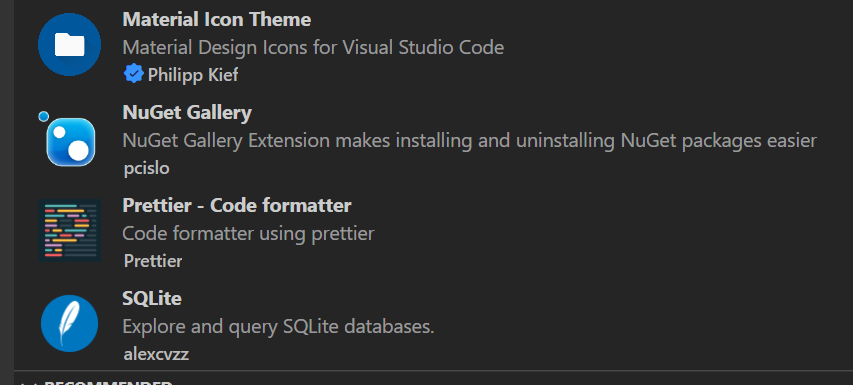


**Extensions**

Add the following: Visual Studio extension, please add the following extension.

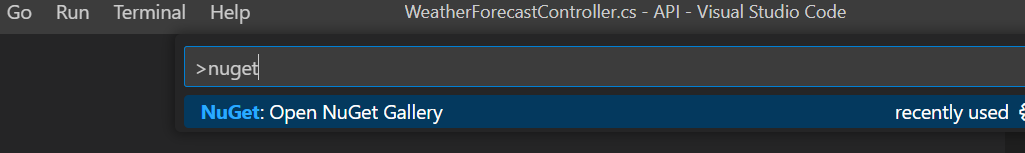




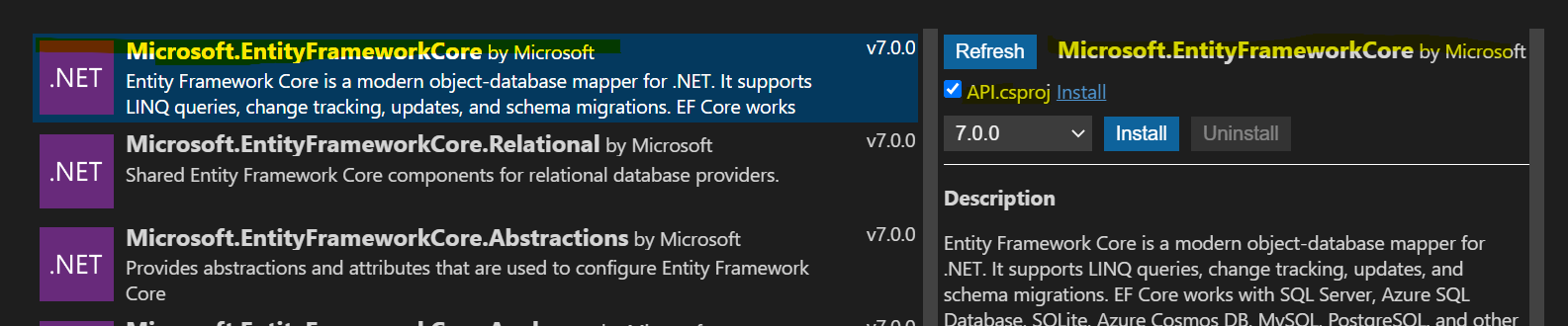


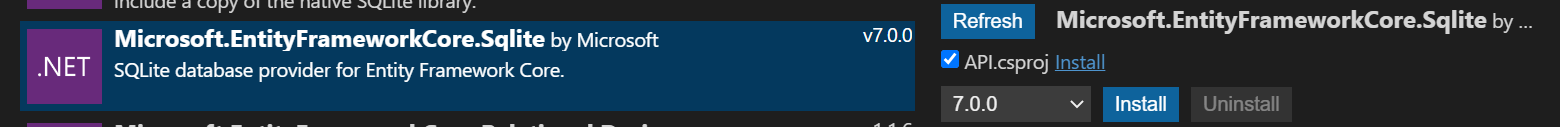
**Install Packages**

* Press Ctrl+Shift+P
* Search for Nuget



* Select and Press Enter
* Look for and install the following packages.







**Application Side:**

1. Create folder, and named it as “Models”, where we can store all user business logic
2. Create class and named it as “User”, the class below

namespace API.Models

{

    public class User

    {

        public int Id { get; set; }

        public string Username { get; set; }

        public byte[] PasswordHash { get; set; }

        public byte[] PasswordSalt { get; set; }

    }

}

If you are facing error on NULLABLE issue, Open the API.csproj and remove or comment the <Nullable> section

<PropertyGroup>

    <TargetFramework>net6.0</TargetFramework>

    <!-- <Nullable>enable</Nullable> -->

    <ImplicitUsings>enable</ImplicitUsings>

  </PropertyGroup>

1. Create “Model” in root level and add a new class and named as DataContext

using API.Models;

using Microsoft.EntityFrameworkCore;

namespace API.Data

{

    public class DataContext : DbContext

    {

        public DataContext(DbContextOptions<DataContext> options) : base(options){}

        public DbSet<User> Users { get; set; }

    }

}

**Appsettings.json**

{

  "ConnectionStrings": {

    "DefaultConnection": "Data Source=sampleapp.db"

  },

  "TokenKey": "secret key should be here"

  ,

  "Logging": {

    "LogLevel": {

      "Default": "Information",

      "Microsoft.AspNetCore": "Warning"

    }

  },

  "AllowedHosts": "\*"

}

**DTOs**

1. LoginDto
2. RegisterDto
3. UserDto
4. Login DTO:

namespace API.Dtos

{

    public class LoginDto

    {

        public string Username { get; set; }

        public string Password { get; set; }

    }

}

1. Register DTO

namespace API.Dtos

{

    public class RegisterDto

    {

        [Required]

        public string Username { get; set; }

        [Required]

        [StringLength(12, MinimumLength = 6, ErrorMessage = "You must specify password between 6 and 12 characters")]

        public string Password { get; set; }

    }

}

1. User DTO

namespace API.Dtos

{

    public class UserDto

    {

        public string Username { get; set; }

        public string Token { get; set; }

    }

}

**Services:**

1. **ITokenService**
2. **TokenService**
3. **ITokenService**

namespace API.Services

{

    public interface ITokenService

    {

         string CreateToken(User user);

    }

}

1. **TokenService**

namespace API.Services

{

    public class TokenService : ITokenService

    {

        private readonly SymmetricSecurityKey \_key;

        public TokenService(IConfiguration config)

        {

            \_key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(config["TokenKey"]));

        }

        public string CreateToken(User user)

        {

            var claims = new List<Claim>

            {

                new Claim(JwtRegisteredClaimNames.NameId, user.Username)

            };

            var creds = new SigningCredentials(\_key, SecurityAlgorithms.HmacSha512Signature);

            var tokenDescriptor = new SecurityTokenDescriptor

            {

                Subject = new ClaimsIdentity(claims),

                Expires = DateTime.Now.AddDays(7),

                SigningCredentials = creds

            };

            var tokenHandler = new JwtSecurityTokenHandler();

            var token = tokenHandler.CreateToken(tokenDescriptor);

            return tokenHandler.WriteToken(token);

        }

    }

}

**API Controller**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using API.Dtos;

using API.Models;

using Microsoft.AspNetCore.Mvc;

using System.IdentityModel.Tokens.Jwt;

using Microsoft.Extensions.Configuration;

using System.Security.Claims;

using API.Services;

using API.Data;

using System.Security.Cryptography;

using System.Text;

using Microsoft.EntityFrameworkCore;

namespace API.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class AuthController : ControllerBase

    {

        private readonly DataContext \_context;

        private readonly ITokenService \_tokenService;

        public AuthController(DataContext context, ITokenService tokenService)

        {

            \_tokenService = tokenService;

            \_context = context;

        }

        [HttpPost("register")] // POST: api/auth

        public async Task<ActionResult<UserDto>> Register(RegisterDto registerDto)

        {

            if (await UserExists(registerDto.Username)) return BadRequest("Username is taken");

            using var hmac = new HMACSHA512();

            var user = new User

            {

                Username = registerDto.Username.ToLower(),

                PasswordHash = hmac.ComputeHash(Encoding.UTF8.GetBytes(registerDto.Password)),

                PasswordSalt = hmac.Key

            };

            \_context.Users.Add(user);

            await \_context.SaveChangesAsync();

            return new UserDto

            {

                Username = user.Username,

                Token = \_tokenService.CreateToken(user)

            };

        }

        [HttpPost("login")]

        public async Task<ActionResult<UserDto>> Login(LoginDto loginDto)

        {

            var user = await \_context.Users.SingleOrDefaultAsync(x =>

                x.Username == loginDto.Username);

            if (user == null) return Unauthorized("invalid username");

            using var hmac = new HMACSHA512(user.PasswordSalt);

            var computedHash = hmac.ComputeHash(Encoding.UTF8.GetBytes(loginDto.Password));

            for (int i = 0; i < computedHash.Length; i++)

            {

                if (computedHash[i] != user.PasswordHash[i]) return Unauthorized("invalid password");

            }

            return new UserDto

            {

                Username = user.Username,

                Token = \_tokenService.CreateToken(user)

            };

        }

        private async Task<bool> UserExists(string username)

        {

            return await \_context.Users.AnyAsync(x => x.Username == username.ToLower());

        }

    }

}

**Program.cs**

using API.Data;

using API.Services;

using Microsoft.EntityFrameworkCore;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<DataContext> (option=> {

    option.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"));

});

builder.Services.AddCors();

builder.Services.AddScoped<ITokenService, TokenService>();

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

                .AddJwtBearer(options =>

                {

                    options.TokenValidationParameters = new TokenValidationParameters

                    {

                        ValidateIssuerSigningKey = true,

                        IssuerSigningKey = new SymmetricSecurityKey(Encoding

                            .UTF8.GetBytes(builder.Configuration["TokenKey"])),

                        ValidateIssuer = false,

                        ValidateAudience = false

                    };

                });

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Migarion:**

The migrations feature in EF Core **provides a way to incrementally update the database schema to keep it in sync with the application's data model while preserving existing data in the database.**

**Ref.** [Migrations Overview - EF Core | Microsoft Learn](https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli)

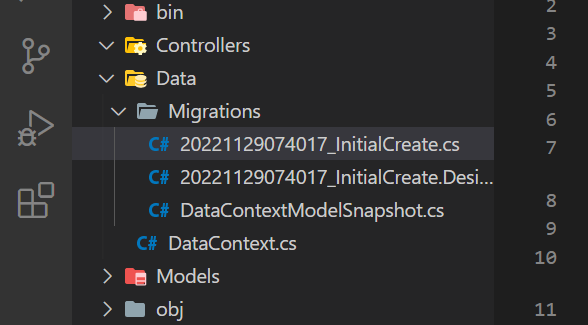
Here, we need togenerates code for a DbContext and entity types for a database. We must first install **dotnet ef** Open the terminal (Ctrl+') and enter the following code.

API> **dotnet tool install --global dotnet-ef**

Once installed, we are ready to migrate our entities. enter the following command in terminal

**API> dotnet ef migrations add InitialCreate -o Data/Migrations**

Then this will create a migration script like below.

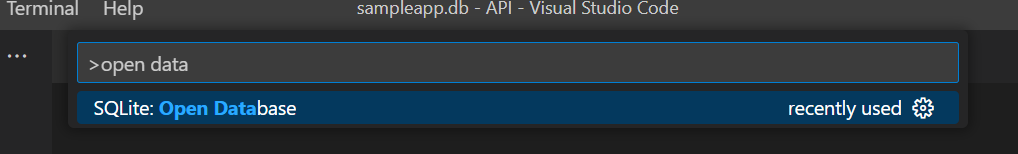


We generated our migration class files, and we need to add the migration snapshots into the database. enter the following command:

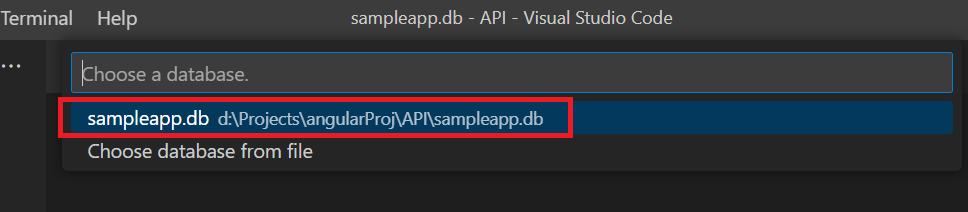
API> **dotnet ef database update**

**How can I check if a database was created or not?**

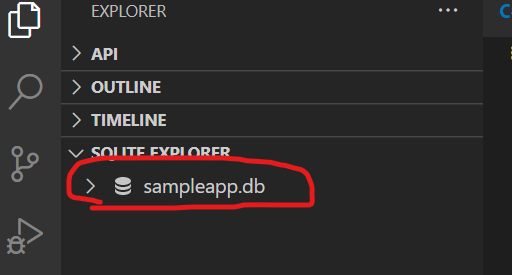
Press "Show all commands" (Ctrl+Shift+P), type "Open Database,"  you will get the screen below, and press enter.

****

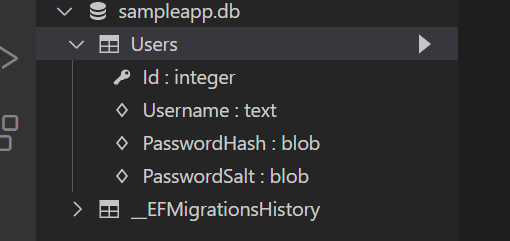
Select your Database, like below



Select your Database from SQLITE EXPLORER



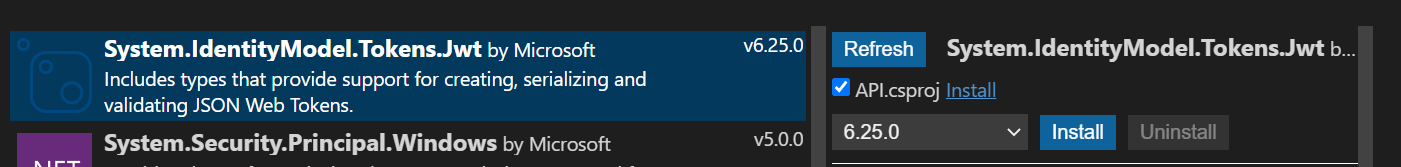
The table has been created successfully. please verify



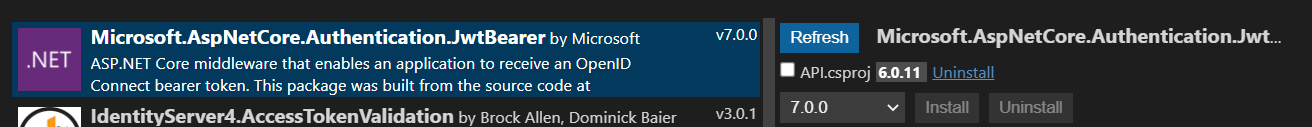
Install JWT

**Implementing JWT Authentication**

Open Nuget and install System.IdentityModel.Token.Jwt

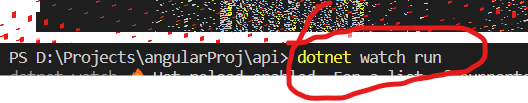


And install Microsoft.AspNetCore.Authentication.JwtBearer



**Application running**:

To run the application, use the following code



**Output**:

