1) Startup.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using Microsoft.IdentityModel.Tokens;

using NetCoreWebApi.Service;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace NetCoreWebApi

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

var securityKey = Encoding.ASCII.GetBytes("This is my sample key");

services.AddControllers();

services.AddSwaggerGen();

services.AddCors(option => option.AddPolicy("OIPAPolicy",

builder =>

{

builder.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod();

})

);

//Jwt Authentication

//services.AddAuthentication(au =>

//{

// au.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

// au.DefaultSignInScheme = JwtBearerDefaults.AuthenticationScheme;

//}).AddJwtBearer(jwt =>

//{

// jwt.RequireHttpsMetadata = false;

// jwt.SaveToken = true;

// jwt.TokenValidationParameters = new TokenValidationParameters

// {

// ValidateIssuerSigningKey = true,

// IssuerSigningKey = new SymmetricSecurityKey(securityKey),

// ValidateIssuer = false,

// ValidateAudience = false,

// };

//});

services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(x => {

x.RequireHttpsMetadata = false;

x.SaveToken = false;

x.TokenValidationParameters = new Microsoft.IdentityModel.Tokens.TokenValidationParameters

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(securityKey),

ValidateIssuer = false,

ValidateAudience = false,

};

});

services.AddTransient<UserService, UserService>();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "My Test1 Api v1");

});

app.UseAuthentication();

app.UseRouting();

app.UseCors("OIPAPolicy");

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

===============================================================================

2) AuthorizeController.ce

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using NetCoreWebApi.Models;

using NetCoreWebApi.Service;

using System;

using System.Collections.Generic;

using System.IdentityModel.Tokens.Jwt;

using System.Linq;

using System.Security.Claims;

using System.Text;

using System.Threading.Tasks;

namespace NetCoreWebApi.Controllers

{

[EnableCors("OIPAPolicy")]

[Route("api/[controller]")]

[ApiController]

public class AuthorizeController : ControllerBase

{

private readonly UserService \_userService;

public AuthorizeController(UserService userService)

{

this.\_userService = userService;

}

[HttpPost("Login")]

public IActionResult Login(LoginModel loginModel)

{

var user = \_userService.GetUserList().Where(x => x.UserName == loginModel.UserName && x.Password == loginModel.Password).FirstOrDefault();

var token = \_userService.GenerateJwtToken(user.UserId, user.UserName);

UserModel userModel = new UserModel();

userModel.UserId = user.UserId;

userModel.UserName = user.UserName;

userModel.Token = token;

return Ok(userModel);

}

}

}

=====================================================================

3) EmployeeController.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using NetCoreWebApi.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace NetCoreWebApi.Controllers

{

[Authorize]

[EnableCors("OIPAPolicy")]

[Route("api/[controller]")]

[ApiController]

public class EmployeeController : ControllerBase

{

[HttpGet("GetEmployeeList")]

public IActionResult GetEmployeeList()

{

List<EmployeeModel> empList = new List<EmployeeModel>()

{

new EmployeeModel{EmployeeId=1, EmployeeName="Employee1",EmployeeEmail="test@gmail.com"}

};

return Ok(empList);

}

}

}

==================================================================

4) UserService.cs

using Microsoft.IdentityModel.Tokens;

using NetCoreWebApi.Models;

using System;

using System.Collections.Generic;

using System.IdentityModel.Tokens.Jwt;

using System.Linq;

using System.Security.Claims;

using System.Text;

using System.Threading.Tasks;

namespace NetCoreWebApi.Service

{

public class UserService

{

public List<LoginModel> GetUserList()

{

List<LoginModel> lst = new List<LoginModel>()

{

new LoginModel{ UserId=1,UserName="test", Password="abc"},

new LoginModel{ UserId=2,UserName="test2", Password="abc"},

};

return lst;

}

public string GenerateJwtToken(int userId, string userName)

{

var tokenHandler = new JwtSecurityTokenHandler();

var securityKey = Encoding.ASCII.GetBytes("This is my sample key");

var tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(new Claim[]

{

new Claim("id", userId.ToString()),

new Claim("userName", userName)

}),

Expires = DateTime.UtcNow.AddHours(1),

SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(securityKey), SecurityAlgorithms.HmacSha256Signature)

};

var token = tokenHandler.CreateToken(tokenDescriptor);

var finaltoken = tokenHandler.WriteToken(token);

return finaltoken;

}

}

}