**1.JsonWebToken**

AuthController.cs :

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace EmpDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [AllowAnonymous]

*public* class AuthController : ControllerBase

    {

        [HttpGet]

*public* IActionResult GetToken()

        {

            try

            {

                var token = GenerateJSONWebToken(1, "Admin");

                return Ok(token);

            }

            catch (Exception ex)

            {

                return StatusCode(500, $"Token generation failed: {ex.Message}");

            }

        }

*private* string GenerateJSONWebToken(int userId, string userRole)

        {

            var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkey1234567890!!"));

            var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

            var claims = new List<Claim>

            {

                new Claim(ClaimTypes.Role, userRole),

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

            };

            var token = new JwtSecurityToken(

                issuer: "mySystem",

                audience: "myUsers",

                claims: claims,

                expires: DateTime.Now.AddMinutes(10),

                signingCredentials: credentials

            );

            return new JwtSecurityTokenHandler().WriteToken(token);

        }

    }

}

Program.cs:

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

string securityKey = "mysuperdupersecretkey1234567890!!";

var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));

builder.Services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

builder.Services.AddControllers();

builder.Services.AddSwaggerGen(options =>

{

options.SwaggerDoc("v1", new OpenApiInfo

{

Title = "EmpDemo",

Version = "v1",

Description = "JWT Auth + Swagger Integration"

});

options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Description = "Enter 'Bearer' [space] and your valid token",

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

options.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseHttpsRedirection();

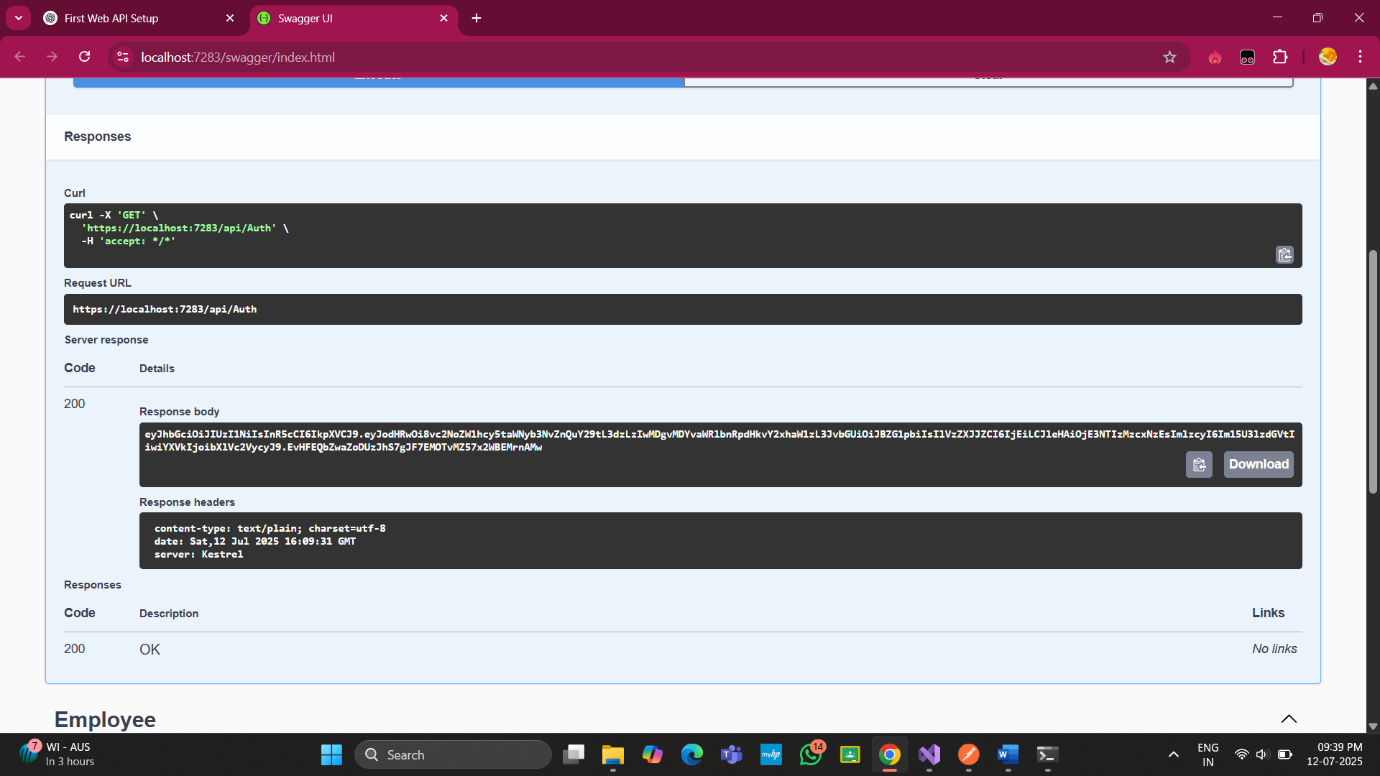
app.UseAuthentication();

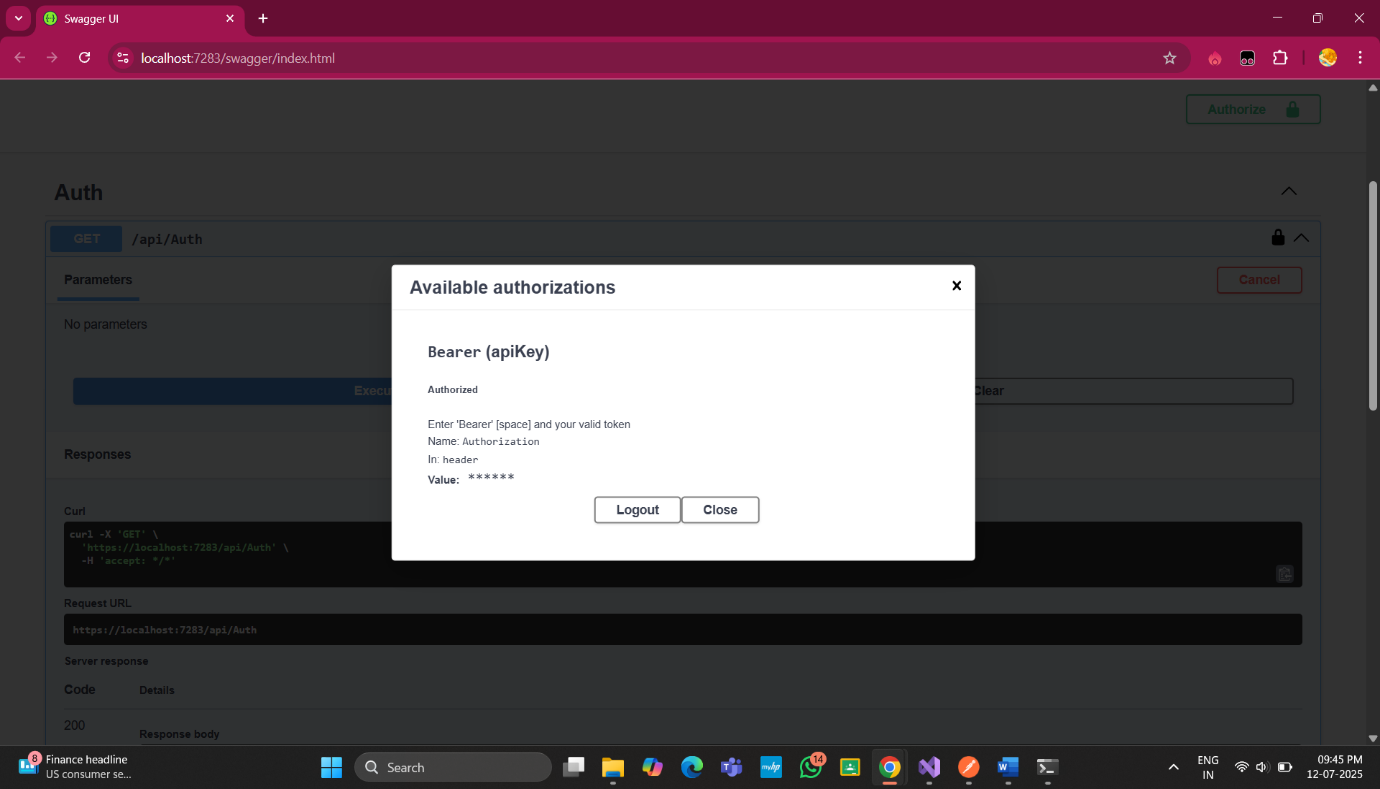
app.UseAuthorization();

app.MapControllers();

app.Run();

Output :





**Check for JWT expiration**

AuthController.cs :

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace EmpDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [AllowAnonymous]

*public* class AuthController : ControllerBase

    {

        [HttpGet]

*public* IActionResult GetToken()

        {

            try

            {

                var token = GenerateJSONWebToken(1, "Admin");

                return Ok(token);

            }

            catch (Exception ex)

            {

                return StatusCode(500, $"Token generation failed: {ex.Message}");

            }

        }

*private* string GenerateJSONWebToken(int userId, string userRole)

        {

            var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkey1234567890!!"));

            var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

            var claims = new List<Claim>

            {

                new Claim(ClaimTypes.Role, userRole),

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

            };

            var token = new JwtSecurityToken(

                issuer: "mySystem",

                audience: "myUsers",

                claims: claims,

                expires: DateTime.Now.AddMinutes(2), // 2 min

                signingCredentials: credentials

            );

            return new JwtSecurityTokenHandler().WriteToken(token);

        }

    }

}

**Add the roles to be authorized in the Authorize attribute**

**EmployeeController.cs**

**namespace EmpDemo.Controllers**

**{**

**[Authorize(Roles = "Admin,POC")]**

**[ApiController]**

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

**[CustomAuthFilter]**

***public* class EmployeeController : ControllerBase**

**{**

***private* *static* List<Employee> employees;**

***static* EmployeeController()**

**{**

**employees = GetStandardEmployeeList();**

**}**

**[HttpGet]**

**[ProducesResponseType(typeof(List<Employee>), 200)]**

**[ProducesResponseType(500)]**

***public* ActionResult<List<Employee>> Get()**

**{**

**return Ok(employees);**

**}**

**[HttpGet("standard")]**

**[ProducesResponseType(typeof(Employee), 200)]**

***public* ActionResult<Employee> GetStandard()**

**{**

**return Ok(employees.First());**

**}**

**[HttpPost]**

**[ProducesResponseType(typeof(Employee), 201)]**

***public* ActionResult<Employee> Post([FromBody] Employee emp)**

**{**

**emp.Id = employees.Max(e => e.Id) + 1;**

**employees.Add(emp);**

**return CreatedAtAction(nameof(Get), new { id = emp.Id }, emp);**

**}**

**[HttpPut("{id}")]**

**[ProducesResponseType(typeof(Employee), 200)]**

**[ProducesResponseType(400)]**

***public* ActionResult<Employee> Put(int id, [FromBody] Employee updatedEmp)**

**{**

**if (id <= 0)**

**{**

**return BadRequest("Invalid employee id");**

**}**

**var existingEmp = employees.FirstOrDefault(e => e.Id == id);**

**if (existingEmp == null)**

**{**

**return BadRequest("Invalid employee id");**

**}**

**existingEmp.Name = updatedEmp.Name;**

**existingEmp.Salary = updatedEmp.Salary;**

**existingEmp.Permanent = updatedEmp.Permanent;**

**existingEmp.DateOfBirth = updatedEmp.DateOfBirth;**

**existingEmp.Department = updatedEmp.Department;**

**existingEmp.Skills = updatedEmp.Skills;**

**return Ok(existingEmp);**

**}**

***private* *static* List<Employee> GetStandardEmployeeList()**

**{**

**return new List<Employee>**

**{**

**new Employee**

**{**

**Id = 1,**

**Name = "John Doe",**

**Salary = 60000,**

**Permanent = true,**

**DateOfBirth = new DateTime(1990, 1, 1),**

**Department = new Department { Id = 1, Name = "Finance" },**

**Skills = new List<Skill>**

**{**

**new Skill { Id = 1, Name = "C#" },**

**new Skill { Id = 2, Name = "SQL" }**

**}**

**},**

**new Employee**

**{**

**Id = 2,**

**Name = "Jane Smith",**

**Salary = 70000,**

**Permanent = false,**

**DateOfBirth = new DateTime(1992, 5, 15),**

**Department = new Department { Id = 2, Name = "HR" },**

**Skills = new List<Skill>**

**{**

**new Skill { Id = 3, Name = "Excel" },**

**new Skill { Id = 4, Name = "Communication" }**

**}**

**}**

**};**

**}**

**}**

**}**