AuthController.cs

using System;

using System.Collections.Generic;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

namespace WebApiDemo.Controllers

{

    [Route("api/[controller]")]

    [ApiController]

    [AllowAnonymous]

    public class AuthController : ControllerBase

    {

        [HttpGet("token")]

        public IActionResult GetToken()

        {

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

            return Ok(token);

        }

        private string GenerateJSONWebToken(int userId, string userRole)

        {

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

            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);

        }

    }

}

EmployeeContoller.cs

using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WebApiDemo.Filters;

using WebApiDemo.Models;

namespace WebApiDemo.Controllers

{

    [Route("api/[controller]")]

    [ApiController]

    [Authorize(Roles = "Admin,POC")]

    [ServiceFilter(typeof(CustomAuthFilter))]

    [ServiceFilter(typeof(CustomExceptionFilter))]

    public class EmployeeController : ControllerBase

    {

        private static List<Employee> \_employees = GetStandardEmployeeList();

        private static List<Employee> GetStandardEmployeeList()

        {

            return new List<Employee>

            {

                new Employee { Id = 1, Name = "John", Salary = 50000, Permanent = true, DateOfBirth = new DateTime(1990, 1, 1),

                    Department = new Department { DeptId = 1, DeptName = "HR" }, Skills = new List<Skill> { new Skill { Id = 1, SkillName = "C#" } } }

            };

        }

        [HttpGet]

        [ProducesResponseType(200)]

        public ActionResult<List<Employee>> GetStandard()

        {

            //throw new Exception("Test exception");

            return Ok(\_employees);

        }

        [HttpPut("{id}")]

        public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updatedEmployee)

        {

            if (id <= 0)

                return BadRequest("Invalid employee id");

            var emp = \_employees.FirstOrDefault(e => e.Id == id);

            if (emp == null)

                return BadRequest("Invalid employee id");

            emp.Name = updatedEmployee.Name;

            emp.Salary = updatedEmployee.Salary;

            emp.Permanent = updatedEmployee.Permanent;

            emp.Department = updatedEmployee.Department;

            emp.Skills = updatedEmployee.Skills;

            emp.DateOfBirth = updatedEmployee.DateOfBirth;

            return Ok(emp);

        }

    }

}

CustomAuthFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebApiDemo.Filters

{

    public class CustomAuthFilter : ActionFilterAttribute

    {

        public override void OnActionExecuting(ActionExecutingContext context)

        {

            if (!context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token))

            {

                context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

                return;

            }

            if (!token.ToString().Contains("Bearer"))

            {

                context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

            }

        }

    }

}

CustomExceptionFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

namespace WebApiDemo.Filters

{

    public class CustomExceptionFilter : IExceptionFilter

    {

        public void OnException(ExceptionContext context)

        {

            File.WriteAllText("logs.txt", context.Exception.ToString());

            context.Result = new ObjectResult("An error occurred") { StatusCode = 500 };

        }

    }

}

Employee.cs

using System;

using System.Collections.Generic;

namespace WebApiDemo.Models

{

    public class Employee

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public int Salary { get; set; }

        public bool Permanent { get; set; }

        public Department Department { get; set; }

        public List<Skill> Skills { get; set; }

        public DateTime DateOfBirth { get; set; }

    }

    public class Department

    {

        public int DeptId { get; set; }

        public string DeptName { get; set; }

    }

    public class Skill

    {

        public int Id { get; set; }

        public string SkillName { get; set; }

    }

}

Program.cs

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

namespace WebApiDemo

{

    public class Program

    {

        public static void Main(string[] args)

        {

            CreateHostBuilder(args).Build().Run();

        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>

            Host.CreateDefaultBuilder(args)

                .ConfigureWebHostDefaults(webBuilder =>

                {

                    webBuilder.UseStartup<Startup>();

                });

    }

}

Startup.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

using WebApiDemo.Filters;

public class Startup

{

    public void ConfigureServices(IServiceCollection services)

    {

        services.AddControllers();

        services.AddScoped<CustomAuthFilter>();

        services.AddScoped<CustomExceptionFilter>();

        services.AddCors(options =>

        {

            options.AddPolicy("AllowAll", builder =>

                builder.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod());

        });

        services.AddAuthentication(x =>

        {

            x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

            x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

        })

        .AddJwtBearer(x =>

        {

            x.TokenValidationParameters = new TokenValidationParameters

            {

                ValidateIssuer = true,

                ValidateAudience = true,

                ValidateLifetime = true,

                ValidateIssuerSigningKey = true,

                ValidIssuer = "mySystem",

                ValidAudience = "myUsers",

                IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecret"))

            };

        });

        services.AddSwaggerGen(c =>

        {

            c.SwaggerDoc("v1", new OpenApiInfo { Title = "Swagger Demo", Version = "v1" });

        });

    }

    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

    {

        app.UseCors("AllowAll");

        app.UseSwagger();

        app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo"));

        app.UseRouting();

        app.UseAuthentication();

        app.UseAuthorization();

        app.UseEndpoints(endpoints => { endpoints.MapControllers(); });

    }

}

Output:



