**CODE:**

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

var securityKey = "mysuperdupersecretkeythatismorethan32chars";

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

builder.Services.AddAuthentication(options =>

{

    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(options =>

{

    options.TokenValidationParameters = new TokenValidationParameters

    {

        ValidateIssuer = true,

        ValidateAudience = true,

        ValidateLifetime = true,

        ValidateIssuerSigningKey = true,

        ValidIssuer = "mySystem",

        ValidAudience = "myUsers",

        IssuerSigningKey = signingKey

    };

});

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

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

    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

    {

        Type = SecuritySchemeType.Http,

        Scheme = "bearer",

        BearerFormat = "JWT",

        In = ParameterLocation.Header,

        Description = "Enter 'Bearer {your JWT token}'"

    });

    c.AddSecurityRequirement(new OpenApiSecurityRequirement

    {

        {

            new OpenApiSecurityScheme

            {

                Reference = new OpenApiReference

                { Type = ReferenceType.SecurityScheme, Id = "Bearer" }

            },

            new string[] {}

        }

    });

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

    c.SwaggerEndpoint("/swagger/v1/swagger.json", "MyFirstWebAPI v1");

});

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Startup.cs**

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;

namespace MyFirstWebAPI

{

    public class Startup

    {

        public IConfiguration Configuration { get; }

        public Startup(IConfiguration configuration)

        {

            Configuration = configuration;

        }

        public void ConfigureServices(IServiceCollection services)

        {

            services.AddCors(options =>

            {

                options.AddDefaultPolicy(builder =>

                {

                    builder.AllowAnyOrigin()

                           .AllowAnyMethod()

                           .AllowAnyHeader();

                });

            });

            services.AddControllers();

            string securityKey = "mysuperdupersecretkeythatismorethan32chars";

            var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeythatismorethan32chars"));

            services.AddAuthentication(x =>

            {

                x.DefaultAuthenticateScheme = "JwtBearer";

                x.DefaultChallengeScheme = "JwtBearer";

            })

            .AddJwtBearer("JwtBearer", x =>

            {

                x.TokenValidationParameters = new TokenValidationParameters

                {

                    ValidateIssuer = true,

                    ValidateAudience = true,

                    ValidateLifetime = true,

                    ValidateIssuerSigningKey = true,

                    ValidIssuer = "mySystem",

                    ValidAudience = "myUsers",

                    IssuerSigningKey = symmetricSecurityKey

                };

            });

            services.AddSwaggerGen(c =>

            {

                c.SwaggerDoc("v1", new OpenApiInfo

                {

                    Title = "My API with JWT",

                    Version = "v1",

                    Description = "Demo Web API with JWT Auth",

                    Contact = new OpenApiContact

                    {

                        Name = "Sumithra",

                        Email = "sumithra@example.com",

                        Url = new System.Uri("https://example.com")

                    }

                });

                c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

                {

                    In = ParameterLocation.Header,

                    Description = "JWT Authorization header using Bearer scheme. Example: \"Bearer {token}\"",

                    Name = "Authorization",

                    Type = SecuritySchemeType.ApiKey,

                    Scheme = "Bearer"

                });

                c.AddSecurityRequirement(new OpenApiSecurityRequirement

                {

                    {

                        new OpenApiSecurityScheme {

                            Reference = new OpenApiReference {

                                Type = ReferenceType.SecurityScheme,

                                Id = "Bearer"

                            }

                        },

                        new string[] {}

                    }

                });

            });

        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

        {            if (env.IsDevelopment())

            {

                app.UseDeveloperExceptionPage();

            }

            app.UseHttpsRedirection();

            app.UseSwagger();

            app.UseSwaggerUI(c =>

            {

                c.SwaggerEndpoint("/swagger/v1/swagger.json", "JWT API v1");

            });

            app.UseRouting();

            app.UseCors();

            app.UseAuthentication();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>

            {

                endpoints.MapControllers();

            });

        }

    }

}

**CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Linq;

public class CustomAuthFilter : ActionFilterAttribute

{

    public override void OnActionExecuting(ActionExecutingContext context)

    {

        var hasHeader = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);

        if (!hasHeader)

        {

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

        }

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

        {

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

        }

    }

}

**CustomExecptionFilter.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

public class CustomExceptionFilter : IExceptionFilter

{

    public void OnException(ExceptionContext context)

    {

        var errorMessage = $"[{DateTime.Now}] Exception: {context.Exception.Message}\n";

        File.AppendAllText("log.txt", errorMessage);

        context.Result = new ObjectResult("Something went wrong!")

        {

            StatusCode = 500

        };

    }

}

**AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using Microsoft.AspNetCore.Authorization;

namespace MyFirstWebAPI.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [AllowAnonymous]

    public class AuthController : ControllerBase

    {

        [HttpGet("token")]

        public IActionResult GetToken()

        {

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

            return Ok(new { token });

        }

        private string GenerateJSONWebToken(int userId, string userRole)

        {

            var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeythatismorethan32chars"));

            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),

                signingCredentials: credentials

            );

            return new JwtSecurityTokenHandler().WriteToken(token);

        }

    }

}

**EmployeeController.cs: [1 and 2]**

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebAPI.Controllers

{

    [ApiController]

    [Route("Employee")]

    public class EmployeeController : ControllerBase

    {

        [HttpGet]

        public IActionResult GetEmployees()

        {

            var employees = new[]

            {

                new { Id = 1, Name = "Alice", Role = "Developer" },

                new { Id = 2, Name = "Bob", Role = "Tester" }

            };

            return Ok(employees);

        }

        [HttpPost]

        public IActionResult AddEmployee([FromBody] object emp)

        {

            return Ok(new { Message = "Employee added", Data = emp });

        }

    }

}

**EmployeeController.cs: [for 3,4,5]**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Authorization;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace MyFirstWebAPI.Controllers

{

    [ApiController]

    [Route("Emp")]

    [Authorize(Roles = "Admin")]

    [AllowAnonymous]

    public class EmployeeController : ControllerBase

    {

        private static List<Employee> employeeList = new List<Employee>

{

    new Employee

    {

        Id = 1,

        Name = "Alice",

        Salary = 50000,

        Permanent = true,

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

        Department = new Department { Id = 1, Name = "HR" },

        Skills = new List<Skill> {

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

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

        }

    },

    new Employee

    {

        Id = 2,

        Name = "Bob",

        Salary = 60000,

        Permanent = false,

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

        Department = new Department { Id = 2, Name = "IT" },

        Skills = new List<Skill> {

            new Skill { Id = 3, Name = "Java" },

            new Skill { Id = 4, Name = "Angular" }

        }

    }

};

        [HttpGet("GetStandard")]

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

        [ProducesResponseType(500)]

        public ActionResult<List<Employee>> GetStandard()

        {

           // throw new Exception("Testing exception filter!");  to test exception

            return employeeList;

        }

        [HttpPut]

[ProducesResponseType(typeof(Employee), 200)]

[ProducesResponseType(400)]

public ActionResult<Employee> UpdateEmployee([FromBody] Employee updatedEmp)

{

    if (updatedEmp.Id <= 0)

        return BadRequest("Invalid employee id");

    var existingEmp = employeeList.Find(e => e.Id == updatedEmp.Id);

    if (existingEmp == null)

        return BadRequest("Invalid employee id");

    existingEmp.Name = updatedEmp.Name;

    existingEmp.Salary = updatedEmp.Salary;

    existingEmp.Permanent = updatedEmp.Permanent;

    existingEmp.Department = updatedEmp.Department;

    existingEmp.Skills = updatedEmp.Skills;

    existingEmp.DateOfBirth = updatedEmp.DateOfBirth;

    return Ok(existingEmp);

}

    }

}

**OUTPUT:**





  






