**Implement JWT Authentication in ASP.NET Core Web API**

**CODE:**

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

    options.SwaggerDoc("v1", new OpenApiInfo { Title = "JwtAuthAPI", Version = "v1" });

    options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

    {

        Name = "Authorization",

        Type = SecuritySchemeType.ApiKey,

        Scheme = "Bearer",

        BearerFormat = "JWT",

        In = ParameterLocation.Header,

        Description = "Enter 'Bearer' followed by space and your token.\nExample: Bearer abc123xyz"

    });

    options.AddSecurityRequirement(new OpenApiSecurityRequirement

    {

        {

            new OpenApiSecurityScheme

            {

                Reference = new OpenApiReference

                {

                    Type=ReferenceType.SecurityScheme,

                    Id="Bearer"

                }

            },

            Array.Empty<string>()

        }

    });

});

var jwtSettings = builder.Configuration.GetSection("Jwt");

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

    .AddJwtBearer(options =>

    {

        options.TokenValidationParameters = new TokenValidationParameters

        {

            ValidateIssuer = true,

            ValidateAudience = true,

            ValidateLifetime = true,

            ValidateIssuerSigningKey = true,

            ValidIssuer = jwtSettings["Issuer"],

            ValidAudience = jwtSettings["Audience"],

            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings["Key"]))

        };

    });

builder.Services.AddAuthorization();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Controller.cs**

**-AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using JwtAuthAPI.Models;

[ApiController]

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

public class AuthController : ControllerBase

{

    private readonly IConfiguration \_config;

    public AuthController(IConfiguration config)

    {

        \_config = config;

    }

    [HttpPost("login")]

    public IActionResult Login([FromBody] LoginModel model)

    {

        if (model.Username == "admin" && model.Password == "password")

        {

            var token = GenerateToken(model.Username);

            return Ok(new { Token = token });

        }

        return Unauthorized();

    }

    private string GenerateToken(string username)

    {

        var claims = new[]

        {

            new Claim(ClaimTypes.Name, username)

        };

        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config["Jwt:Key"]));

        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

        var token = new JwtSecurityToken(

            issuer: \_config["Jwt:Issuer"],

            audience: \_config["Jwt:Audience"],

            claims: claims,

            expires: DateTime.Now.AddMinutes(60),

            signingCredentials: creds);

        return new JwtSecurityTokenHandler().WriteToken(token);

    }

}

**-SecureController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

[ApiController]

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

public class SecureController : ControllerBase

{

    [HttpGet]

    [Authorize]

    public IActionResult GetSecret()

    {

        return Ok("This is a protected API endpoint.");

    }

}

**LoginModel.cs**

namespace JwtAuthAPI.Models

{

    public class LoginModel

    {

        public string Username { get; set; }

        public string Password { get; set; }

    }

}

**OUTPUT:**





