**Week 5 ASP.NET Core 8.0 Web API**

**Source code**

**Task1 : Authentication and Authorization in ASP.NET Core**

**Web API Microservices**

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Configure JWT Authentication

var key = Encoding.ASCII.GetBytes("ThisIsASecretKeyForJwtTokenGenerationDontShare");

builder.Services.AddAuthentication(options =>

{

    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

    options.RequireHttpsMetadata = false;

    options.SaveToken = true;

    options.TokenValidationParameters = new TokenValidationParameters

    {

        ValidateIssuerSigningKey = true,

        IssuerSigningKey = new SymmetricSecurityKey(key),

        ValidateIssuer = false,

        ValidateAudience = false,

        ClockSkew = TimeSpan.Zero

    };

});

builder.Services.AddAuthorization();

var app = builder.Build();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

// Models

public record UserLogin(string Username, string Password);

// Controllers

[ApiController]

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

public class AuthController : ControllerBase

{

    private readonly byte[] \_key;

    public AuthController(IConfiguration config)

    {

        \_key = Encoding.ASCII.GetBytes("ThisIsASecretKeyForJwtTokenGenerationDontShare");

    }

    [HttpPost("login")]

    public IActionResult Login([FromBody] UserLogin user)

    {

        // For demo, we hardcode a single user

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

        {

            var tokenHandler = new System.IdentityModel.Tokens.Jwt.JwtSecurityTokenHandler();

            var tokenDescriptor = new SecurityTokenDescriptor

            {

                Subject = new System.Security.Claims.ClaimsIdentity(new[]

                {

                    new System.Security.Claims.Claim(System.Security.Claims.ClaimTypes.Name, user.Username)

                }),

                Expires = DateTime.UtcNow.AddHours(1),

                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(\_key), SecurityAlgorithms.HmacSha256Signature)

            };

            var token = tokenHandler.CreateToken(tokenDescriptor);

            var tokenString = tokenHandler.WriteToken(token);

            return Ok(new { Token = tokenString });

        }

        return Unauthorized();

    }

}

[ApiController]

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

public class SampleController : ControllerBase

{

    [HttpGet("public")]

    public IActionResult Public() => Ok("This is a public endpoint.");

    [HttpGet("protected")]

    [Authorize]

    public IActionResult Protected() => Ok($"This is a protected endpoint. Hello {User.Identity.Name}!");

}

**Exercise1.csproj**

<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>

    <TargetFramework>net9.0</TargetFramework>

    <Nullable>enable</Nullable>

    <ImplicitUsings>enable</ImplicitUsings>

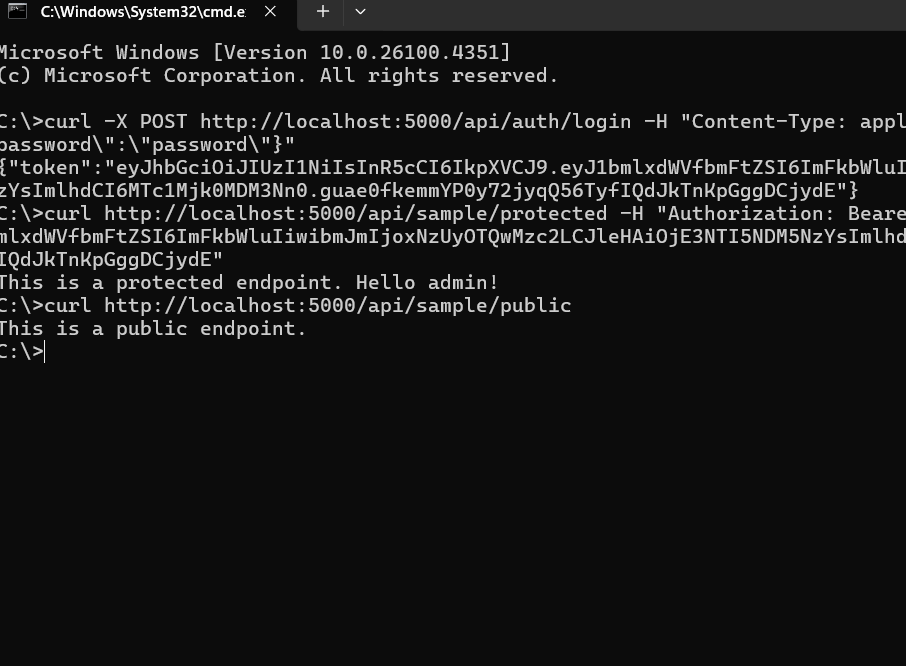
  </PropertyGroup>

  <ItemGroup>

    <PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="7.0.0" />

  </ItemGroup>

</Project>



**Task 2: Implement JWT Authentication in ASP.NET Core Web API**

**Source code**

**API Gateway**

**Program.cs**

using Ocelot.DependencyInjection;

using Ocelot.Middleware;

var builder = WebApplication.CreateBuilder(args);

builder.Configuration.AddJsonFile("ocelot.json", optional: false, reloadOnChange: true);

builder.Services.AddOcelot(builder.Configuration);

var app = builder.Build();

await app.UseOcelot();

app.Run();

**Authservice**

**Authcontroller.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

[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 user)

    {

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

        {

            var tokenString = GenerateJWTToken(user.Username);

            return Ok(new { Token = tokenString });

        }

        return Unauthorized();

    }

    private string GenerateJWTToken(string username)

    {

        var claims = new[]

        {

            new Claim(JwtRegisteredClaimNames.Sub, username),

            new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())

        };

        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);

    }

}

public class LoginModel

{

    public string Username { get; set; }

    public string Password { get; set; }

}

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

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 = builder.Configuration["Jwt:Issuer"],

        ValidAudience = builder.Configuration["Jwt:Audience"],

        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))

    };

});

builder.Services.AddAuthorization();

var app = builder.Build();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**ProductService**

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

    .AddJwtBearer(options =>

    {

        options.TokenValidationParameters = new TokenValidationParameters

        {

            ValidateIssuer = true,

            ValidateAudience = true,

            ValidateLifetime = true,

            ValidateIssuerSigningKey = true,

            ValidIssuer = builder.Configuration["Jwt:Issuer"],

            ValidAudience = builder.Configuration["Jwt:Audience"],

            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))

        };

    });

builder.Services.AddAuthorization();

var app = builder.Build();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

