**WEEK 5**

**Microservices Architecture using ASP.NET Core Web API**

**- Adity Mansinka**

**Superset ID: 6362282**

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

**Scenario:**

**You are building a microservice that requires secure login. You need to implement JWT based authentication.**

**To achieve this:**

**1. Create a new ASP.NET Core Web API project.**

**2. Add a `User` model and a login endpoint.**

**3. Generate a JWT token upon successful login.**

**4. Secure an endpoint using `[Authorize]`.**

**Code –**

**Filename – Appsettings.json**

{

"Jwt": {

"Key": "ThisIsASecureKeyForJwtToken@2025##@@@",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**LoginModel.cs**

namespace JwtAuthDemo.Models

{

public class LoginModel

{

public required string Username { get; set; }

public required 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.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

builder.Services.AddAuthentication("Bearer")

.AddJwtBearer("Bearer", 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.UseHttpsRedirection();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**AuthController.cs**

using JwtAuthDemo.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

private readonly IConfiguration \_configuration;

public AuthController(IConfiguration configuration)

{

\_configuration = configuration;

}

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

if (IsValidUser(model))

{

var token = GenerateJwtToken(model.Username);

return Ok(new { Token = token });

}

return Unauthorized();

}

private bool IsValidUser(LoginModel model)

{

return model.Username == "admin" && model.Password == "password";

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

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

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

var token = new JwtSecurityToken(

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

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

claims: claims,

expires: DateTime.Now.AddMinutes(60),

signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

**SecretController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class SecretController : ControllerBase

{

[Authorize]

[HttpGet]

public IActionResult GetSecret()

{

return Ok("You have accessed a protected endpoint!");

}

}

}

**Output –**



