**Cognizant - DN 4.0 I Deep Skilling**

**WEEK-5**

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

**Hands-On Exercises: Authentication and Authorization in ASP.NET Core Web API Microservices**

This document contains 4 hands-on exercises focusing on Authentication and Authorization in ASP.NET Core Web API microservices, with an emphasis on implementing JWT (JSON Web Tokens) authentication. Each exercise includes a scenario, step-by-step instructions, and complete solution code.

**Question 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.

Steps:

1. Create a new ASP.NET Core Web API project.
2. 2. Add a `User` model and a login endpoint.
3. 3. Generate a JWT token upon successful login. 4. Secure an endpoint using `[Authorize]`.

new JwtSecurityTokenHandler().WriteToken(token); } }

**SOLUTION :**

**CODE -:**

**Appsettings.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"Jwt": {

"Key": "ThisIsASecretKeyForJwtToken",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60

}

}

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers(); // For API controllers

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

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

// Configure Swagger to use JWT Auth

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

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

Name = "Authorization",

In = ParameterLocation.Header,

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

// JWT Configuration

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();

// Middleware pipeline

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthentication(); // Important: add before UseAuthorization

app.UseAuthorization();

app.MapControllers(); // For API endpoints

app.Run();

**LoginModel.cs**

public class LoginModel

{

public string Username { get; set; }

public string Password { get; set; }

}

**Controllers/AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace KafkaChatApp.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

[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)

{

// Simplified validation — replace with DB logic

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

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("ThisIsASecretKeyForJwtToken"));

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

var token = new JwtSecurityToken(

issuer: "MyAuthServer",

audience: "MyApiUsers",

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; }

}

}

**Controllers/HelloController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

[ApiController]

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

public class HelloController : ControllerBase

{

[HttpGet("secure")]

[Authorize]

public IActionResult GetSecureMessage()

{

return Ok("This is a protected endpoint. You are authorized!");

}

}

**OUTPUT -:**











