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

**#TestController.cs:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthMicroservice.Controllers

{

[ApiController]

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

public class TestController : ControllerBase

{

[HttpGet("public")]

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

[Authorize]

[HttpGet("private")]

public IActionResult Private() => Ok("This is a protected endpoint. You are authenticated.");

}

}

**#AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using JwtAuthMicroservice.Models;

namespace JwtAuthMicroservice.Controllers

{

[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 (IsValidUser(model))

{

var token = GenerateJwtToken(model.Username);

return Ok(new { Token = token });

}

return Unauthorized();

}

private bool IsValidUser(LoginModel model)

{

// Static mock user check

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(\_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(Convert.ToDouble(\_config["Jwt:DurationInMinutes"])),

signingCredentials: creds

);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

**#LoginModel.cs**

namespace JwtAuthMicroservice.Models

{

public class LoginModel

{

public string Username { get; set; }

public string Password { get; set; }

}

}

**#appsettings.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"Jwt": {

"Key": "ThisIsASecretKeyForJwtToken1234567890",

"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 to the container

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

// ? Configure Swagger with JWT support

builder.Services.AddSwaggerGen(c =>

{

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

// ?? Add Bearer token support in Swagger UI

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer' followed by your JWT token.\r\n\r\nExample: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

// ? JWT Authentication Configuration

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

var config = builder.Configuration;

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = config["Jwt:Issuer"],

ValidAudience = config["Jwt:Audience"],

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

};

});

builder.Services.AddAuthorization();

var app = builder.Build();

// ? Swagger should be enabled always (not just in development)

app.UseSwagger();

app.UseSwaggerUI();

app.UseHttpsRedirection();

// ?? Authentication must come before Authorization

app.UseAuthentication();

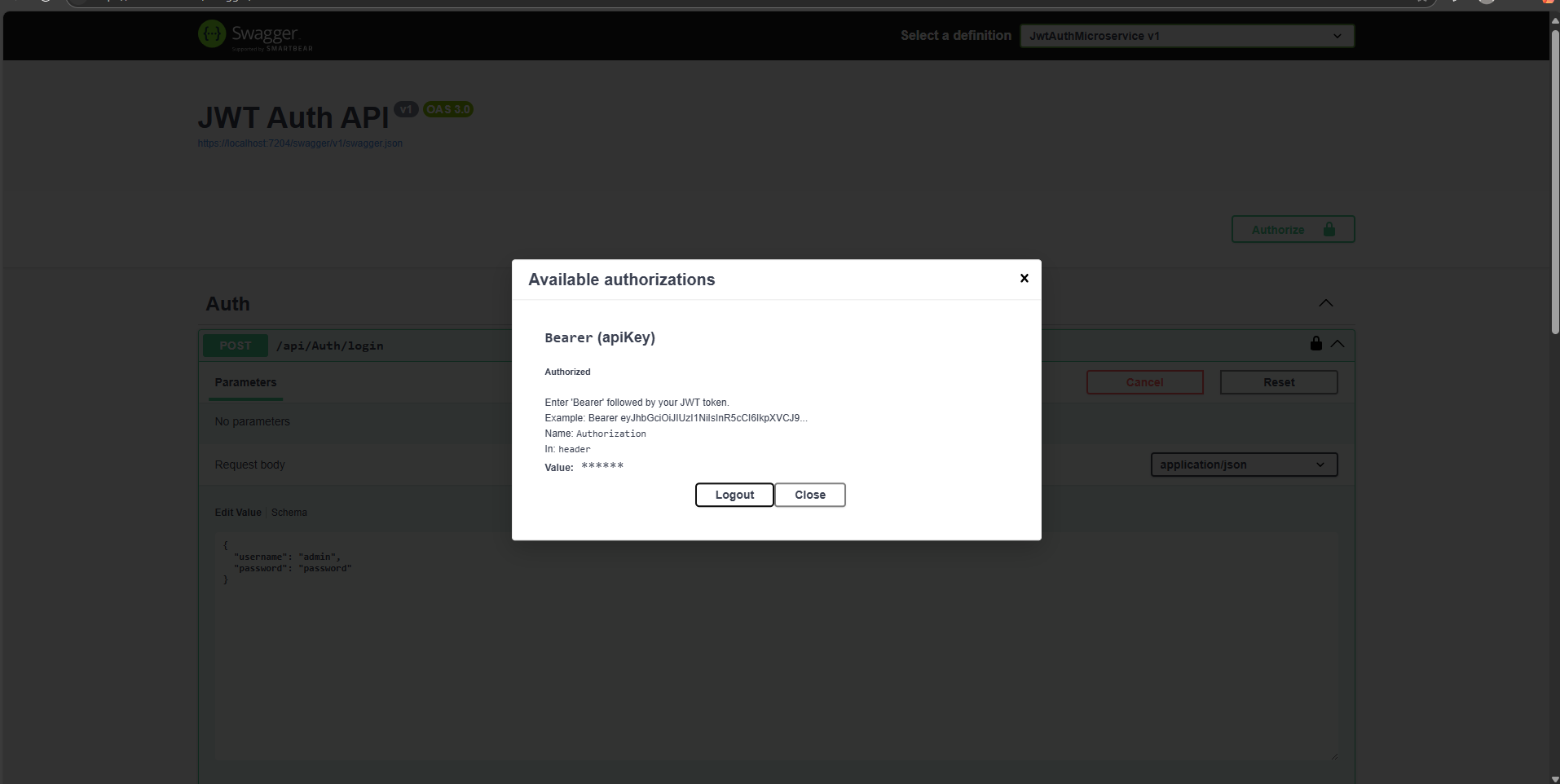
app.UseAuthorization();

app.MapControllers();

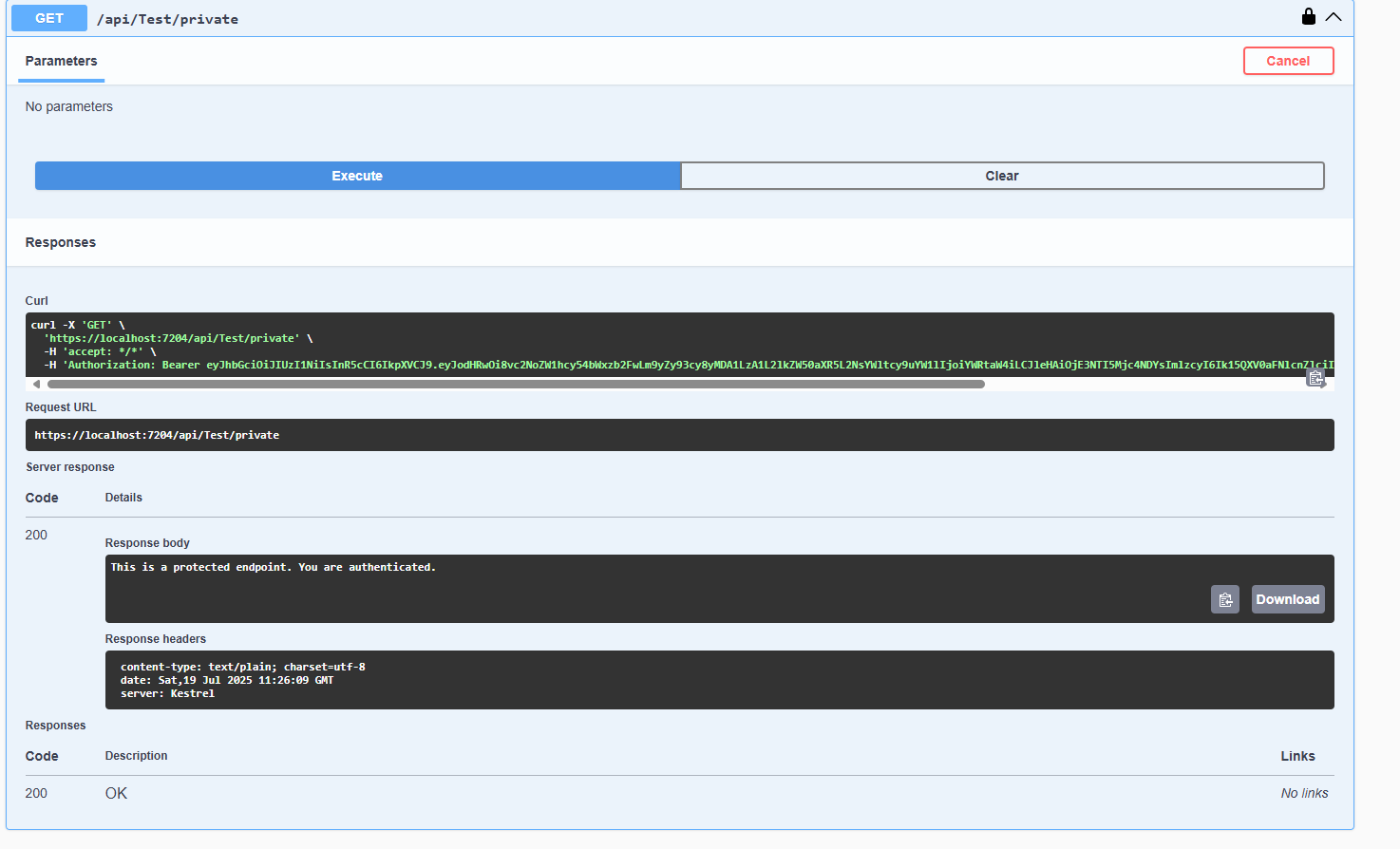
app.Run();

**#OUTPUT:**

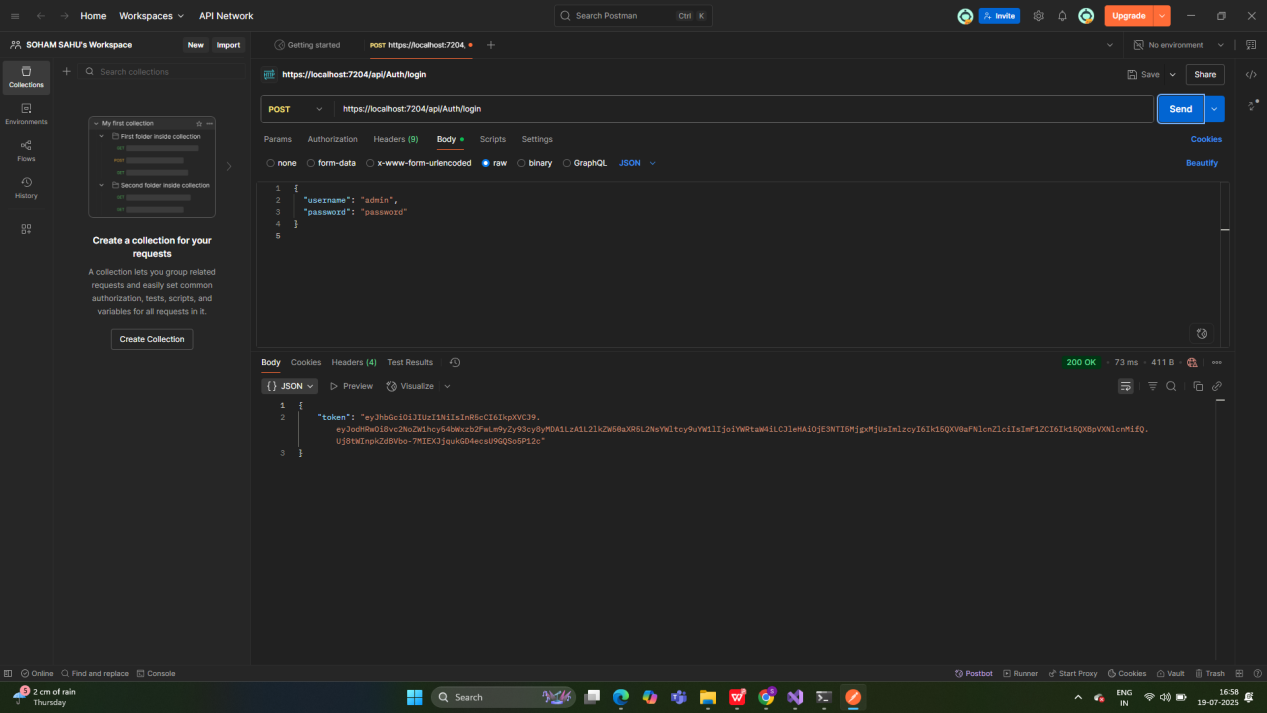
**Authorization:**

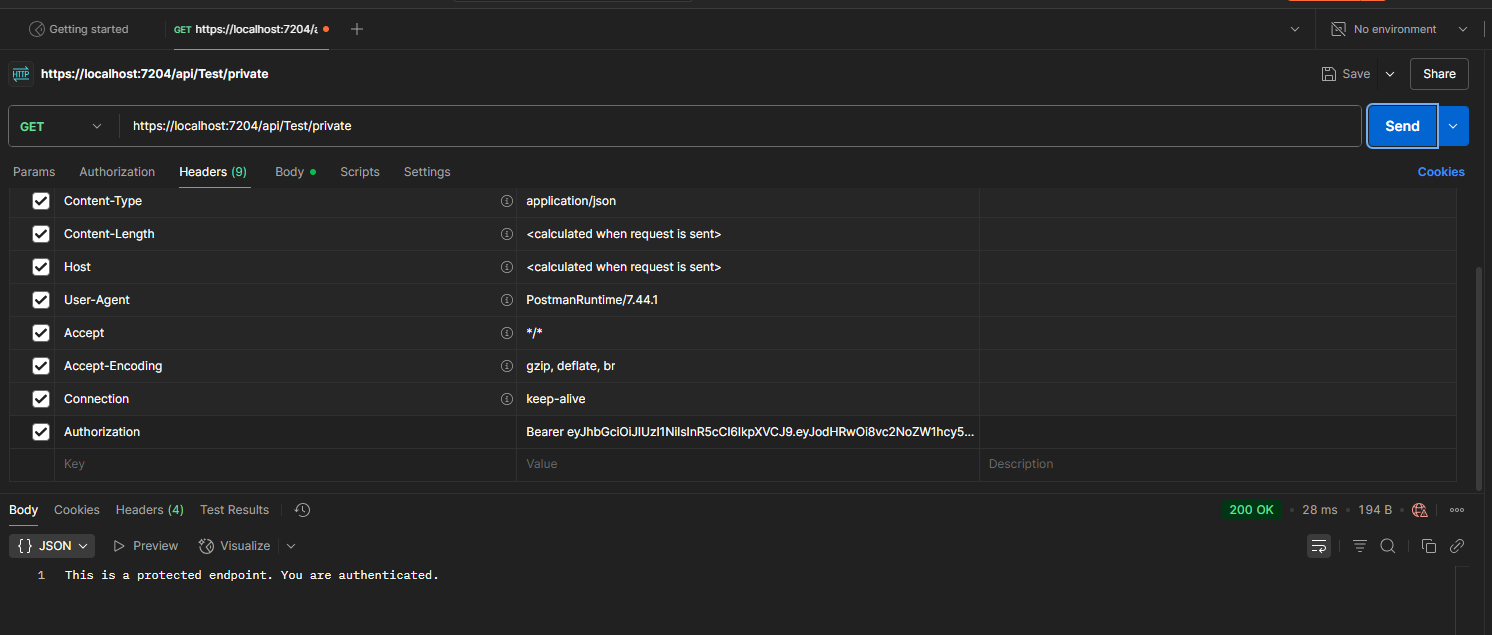
****

**#Access to private message:**

****

**#Using PostMan:**

****

****