ASP.NET Core Web API JWT Authentication - Code Only

# Program.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;  
using Microsoft.IdentityModel.Tokens;  
using System.Text;  
using Microsoft.AspNetCore.Authorization;  
  
var builder = WebApplication.CreateBuilder(args);  
  
builder.Services.AddControllers();  
builder.Services.AddEndpointsApiExplorer();  
builder.Services.AddSwaggerGen();  
  
// JWT Authentication configuration  
builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)  
 .AddJwtBearer(options =>  
 {  
 var jwtSettings = builder.Configuration.GetSection("Jwt");  
 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"] ?? throw new InvalidOperationException("JWT Key is missing.")))  
 };  
  
 options.Events = new JwtBearerEvents  
 {  
 OnAuthenticationFailed = context =>  
 {  
 if (context.Exception.GetType() == typeof(SecurityTokenExpiredException))  
 {  
 context.Response.Headers.Append("Token-Expired", "true");  
 }  
 return Task.CompletedTask;  
 }  
 };  
 });  
  
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();

# LoginModel.cs

namespace JwtAuthDemo.Models  
{  
 public class LoginModel  
 {  
 public string Username { get; set; }  
 public string Password { get; set; }  
 }  
}

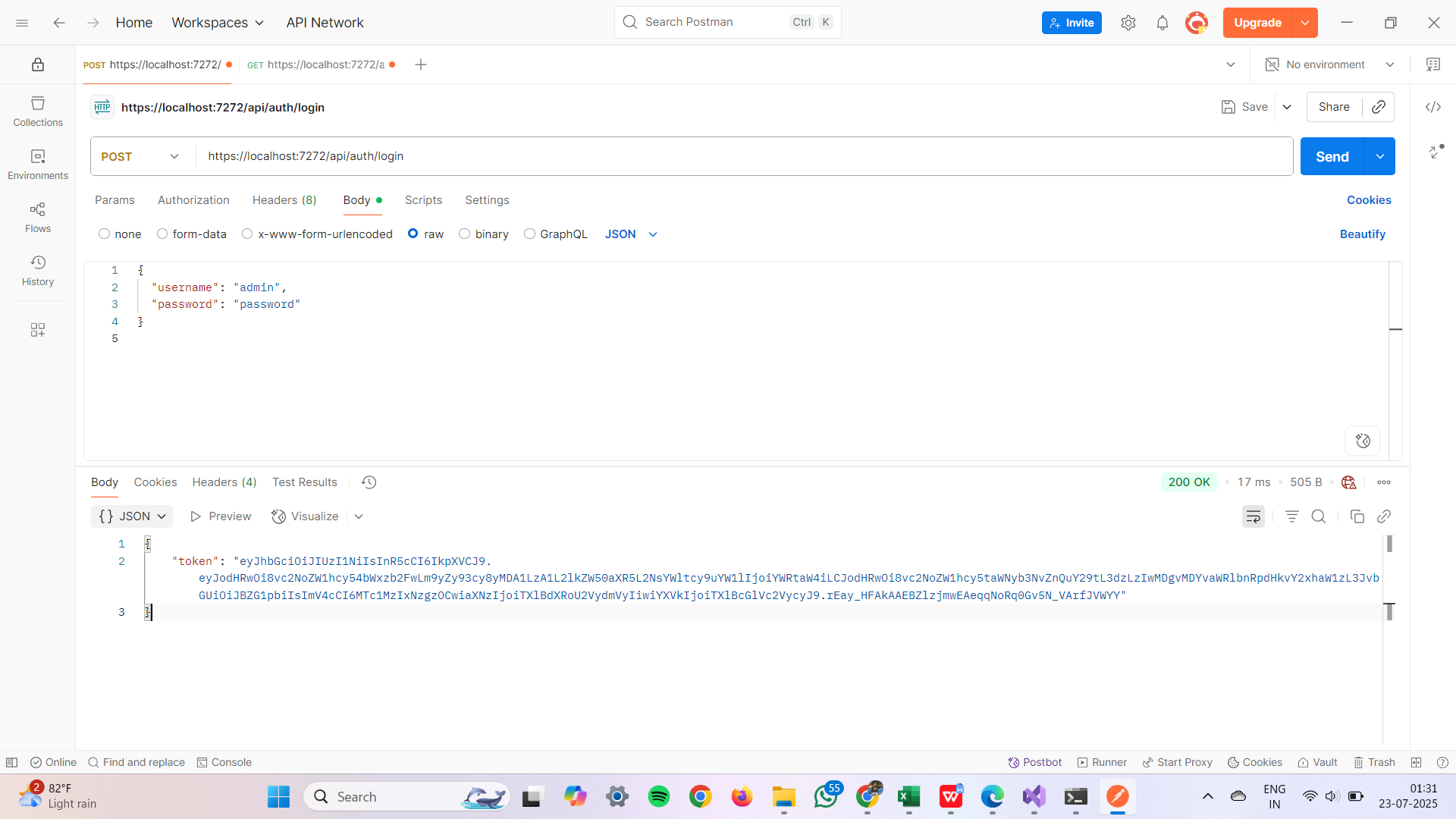
# AuthController.cs

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  
 {  
 [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),  
 new Claim(ClaimTypes.Role, "Admin")  
 };  
  
 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);  
 }  
 }  
}

# SecureController.cs

using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
  
namespace JwtAuthDemo.Controllers  
{  
 [ApiController]  
 [Route("api/[controller]")]  
 public class SecureController : ControllerBase  
 {  
 [HttpGet("data")]  
 [Authorize]  
 public IActionResult GetSecureData()  
 {  
 return Ok("This is protected data.");  
 }  
 }  
}

# AdminController.cs

using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
  
namespace JwtAuthDemo.Controllers  
{  
 [ApiController]  
 [Route("api/[controller]")]  
 public class AdminController : ControllerBase  
 {  
 [HttpGet("dashboard")]  
 [Authorize(Roles = "Admin")]  
 public IActionResult GetAdminDashboard()  
 {  
 return Ok("Welcome to the admin dashboard.");  
 }  
 }  
}  
  
  
  
  
**OUTPUT  
  
POST METHOD :**  
  
 **GET METHOD :**