**Sai Kumaravelu A.T.(916531)**

**Authentication—JWT**

**Startup.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

namespace authentication

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

var key = "This is my first Test Key";

services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(x =>

{

x.RequireHttpsMetadata = false;

x.SaveToken = true;

x.TokenValidationParameters = new Microsoft.IdentityModel.Tokens.TokenValidationParameters

{

ValidateIssuerSigningKey = true,

ValidateIssuer = false,

ValidateAudience = false,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(key))

};

});

services.AddSingleton<auth>(new authh(key));

services.AddSwaggerGen(c =>

{

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

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseSwagger();

app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "authentication v1"));

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

**MembersController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace authentication.Controllers

{

[Authorize]

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

[ApiController]

public class MemberController : ControllerBase

{

private readonly auth jwtAuth;

private readonly List<Member> lstMember = new List<Member>()

{

new Member{Id=1, Name="Kirtesh" },

new Member {Id=2, Name="Nitya" },

new Member{Id=3, Name="pankaj"}

};

public MemberController(auth jwtAuth)

{

this.jwtAuth = jwtAuth;

}

// GET: api/<MemberController>

[HttpGet]

public IEnumerable<Member> AllMembers()

{

return lstMember;

}

// GET api/<MemberController>/5

[HttpGet("{id}")]

public Member MemberByid(int id)

{

return lstMember.Find(x => x.Id == id);

}

[AllowAnonymous]

// POST api/<MemberController>

[HttpPost("authentication")]

public IActionResult Authentication([FromBody] UserCredential userCredential)

{

var token = jwtAuth.Authentication(userCredential.UserName, userCredential.Password);

if (token == null)

return Unauthorized();

return Ok(token);

}

}

}

**Authentication.interface**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace authentication

{

public interface auth

{

string Authentication(string username, string password);

}

}

**Authentication.cs**

using Microsoft.IdentityModel.Tokens;

using System;

using System.Collections.Generic;

using System.IdentityModel.Tokens.Jwt;

using System.Linq;

using System.Security.Claims;

using System.Text;

using System.Threading.Tasks;

namespace authentication

{

public class authh : auth

{

private readonly string username = "kirtesh";

private readonly string password = "Demo1";

private readonly string key;

public authh(string key)

{

this.key = key;

}

public string Authentication(string username, string password)

{

if (!(username.Equals(username) || password.Equals(password)))

{

return null;

}

// 1. Create Security Token Handler

var tokenHandler = new JwtSecurityTokenHandler();

// 2. Create Private Key to Encrypted

var tokenKey = Encoding.ASCII.GetBytes(key);

//3. Create JETdescriptor

var tokenDescriptor = new SecurityTokenDescriptor()

{

Subject = new ClaimsIdentity(

new Claim[]

{

new Claim(ClaimTypes.Name, username)

}),

Expires = DateTime.UtcNow.AddHours(1),

SigningCredentials = new SigningCredentials(

new SymmetricSecurityKey(tokenKey), SecurityAlgorithms.HmacSha256Signature)

};

//4. Create Token

var token = tokenHandler.CreateToken(tokenDescriptor);

// 5. Return Token from method

return tokenHandler.WriteToken(token);

}

}

}

**Member.cs**

namespace authentication.Controllers

{

public class Member

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**UserCredential.cs**

namespace authentication.Controllers

{

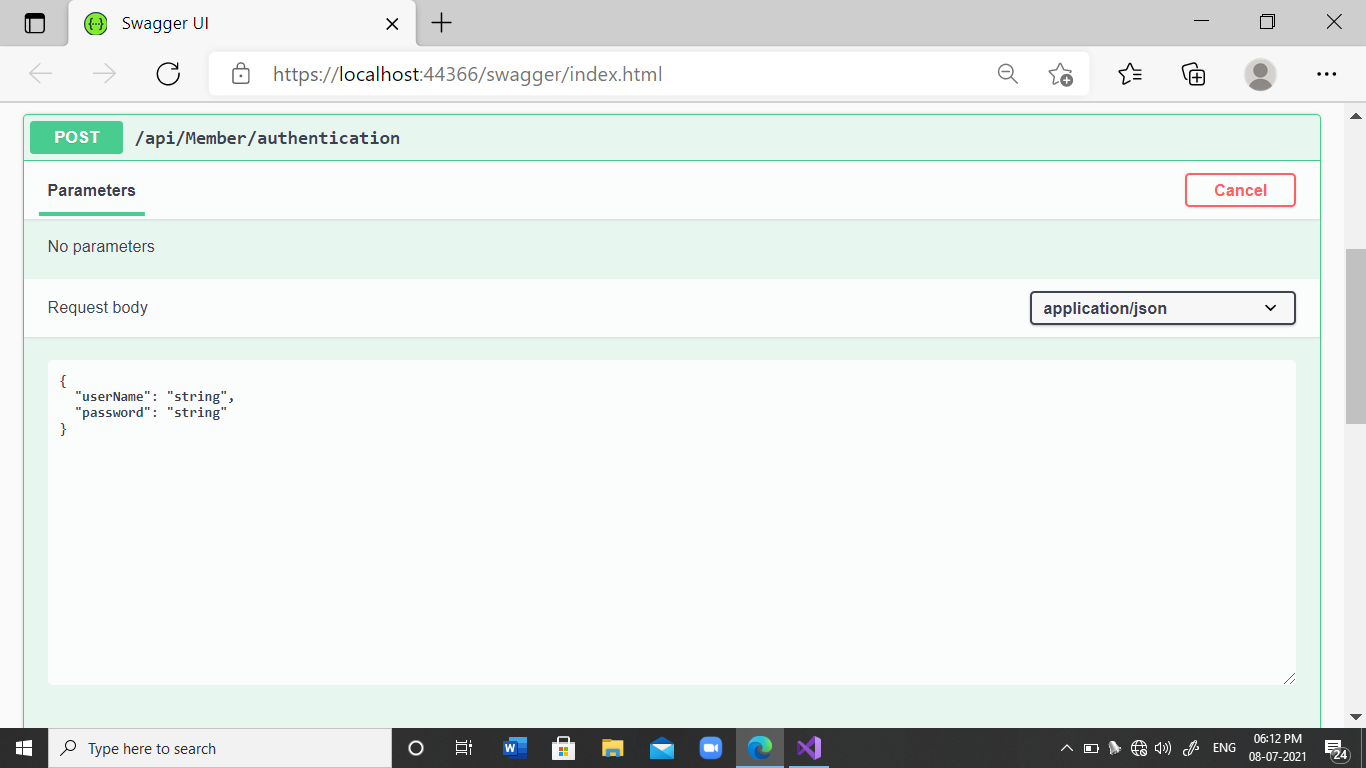
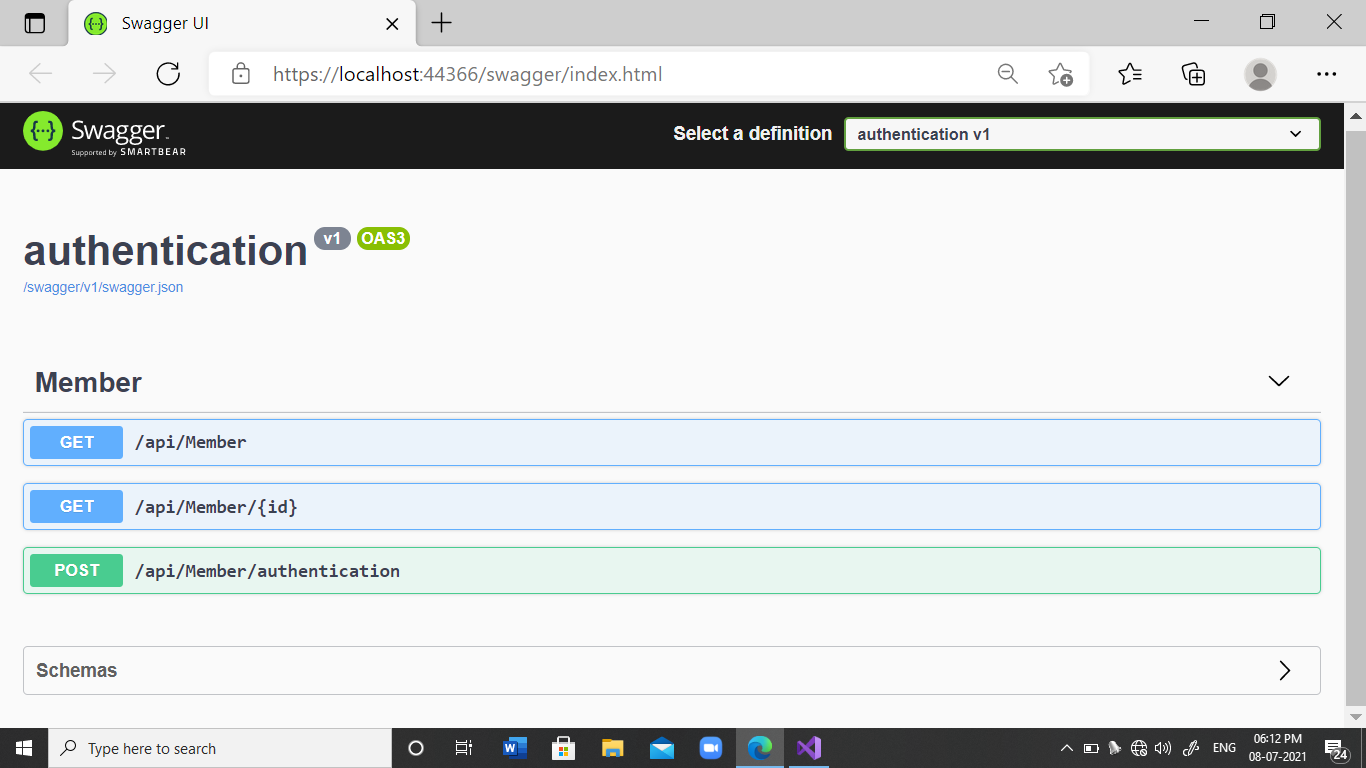
public class UserCredential

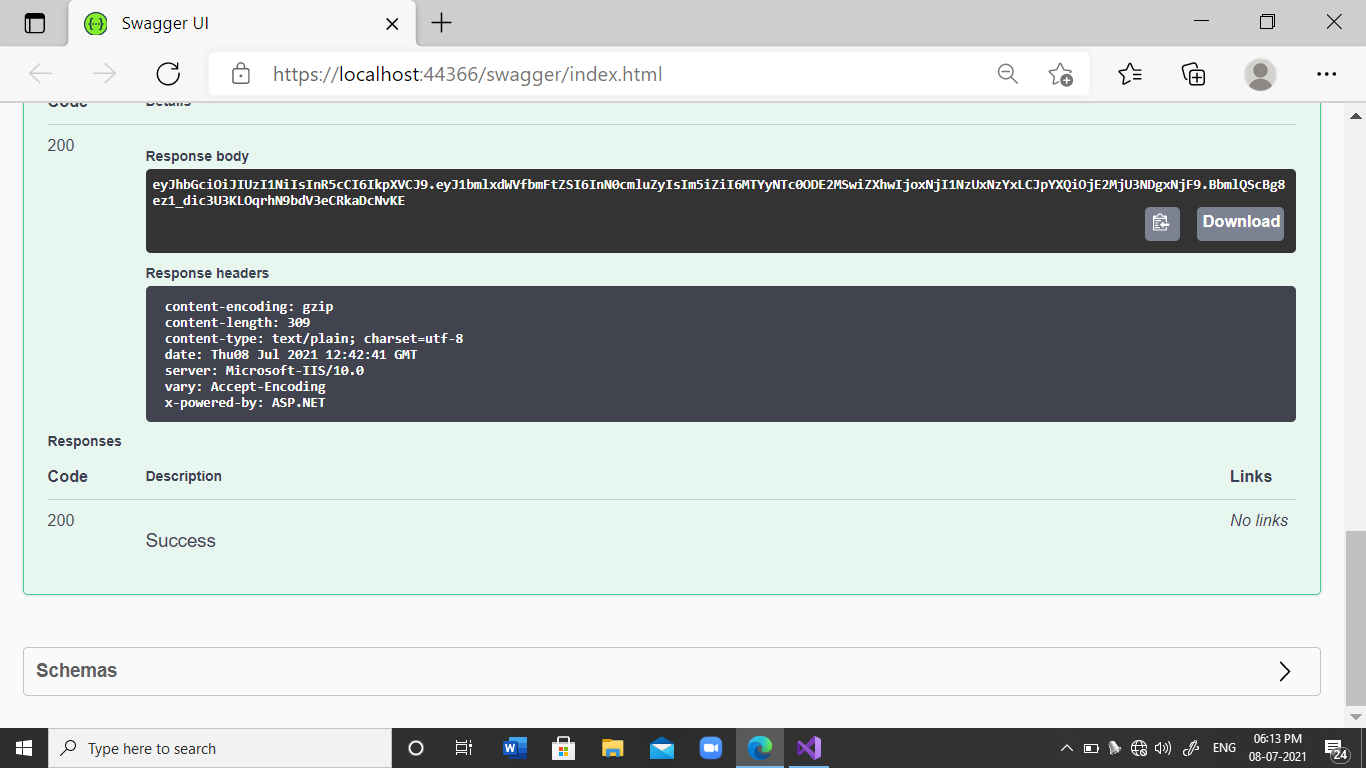
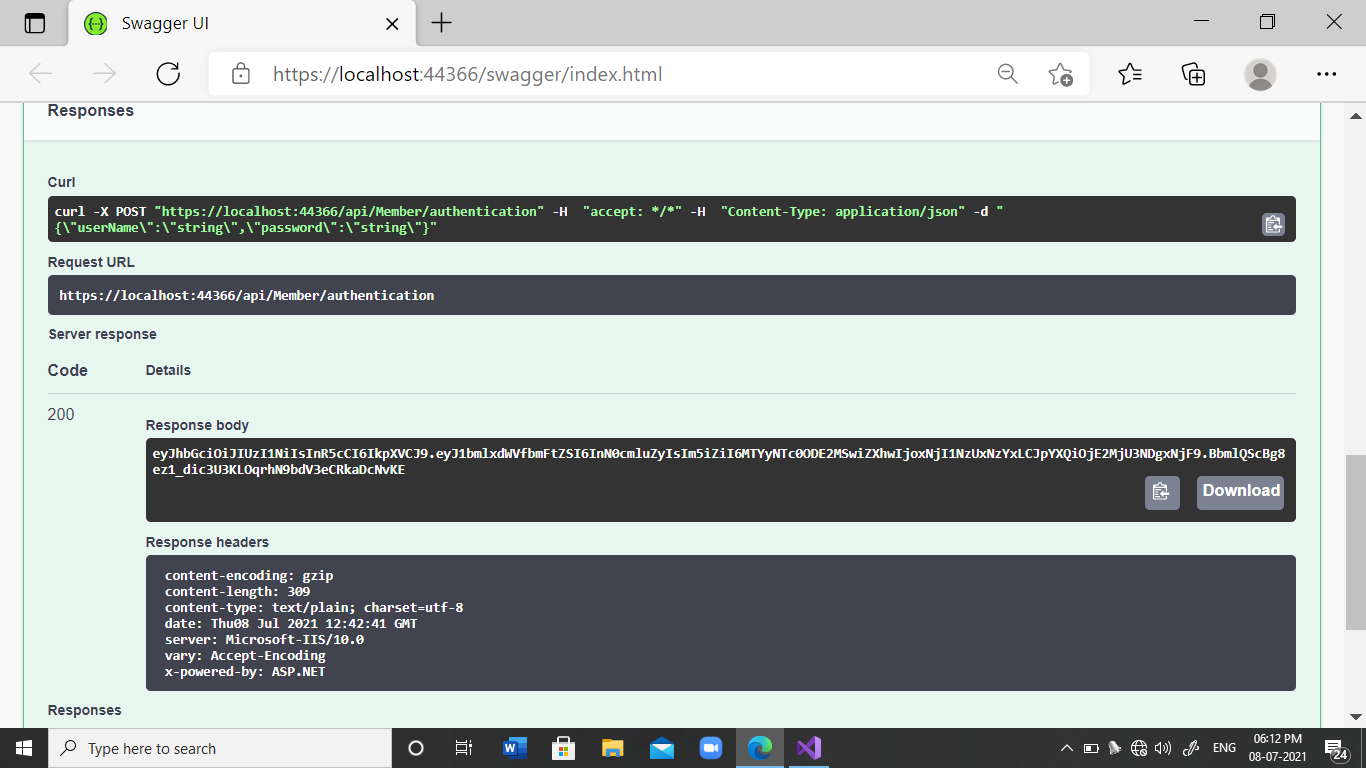
{

public string UserName { get; set; }

public string Password { get; set; }

}

}**OUTPUT**

****