JWT Authentication in ASP.NET Core Web API

Author: *Arya Vats*
SupersetID: 6358118

---

1. Project Overview

Scenario:
You're building a microservice that needs secure login using JSON Web Tokens (JWT). This API demonstrates:

- A hard-coded user login (`admin`/`1234`)
- Generation of a signed JWT
- A protected endpoint secured with `[Authorize]`
- Swagger integration with "Authorize" button

---

2. File Structure

pgsql
CopyEdit

```
JwtAuthDemo/
│
├── Controllers/
│   ├── AuthController.cs
│   └── WeatherController.cs
│
├── Properties/
│   └── launchSettings.json
│
├── appsettings.json
├── JwtAuthDemo.csproj
└── Program.cs
```

3. Configuration

3.1 appsettings.json

json
CopyEdit
```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "Jwt": {
    "Key": "ThisIsASuperSecureKeyWithExtraLength!123456",
    "Issuer": "JwtAuthDemoAPI",
    "Audience": "JwtAuthDemoAPI"
  },
  "AllowedHosts": "*"
}
```

---

4. Startup (`Program.cs`)

csharp
CopyEdit
```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Controllers
builder.Services.AddControllers();

// Swagger with JWT-Bearer support
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "JwtAuthDemo",
Version = "v1" });
    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
    {
        Name         = "Authorization",
        Type         = SecuritySchemeType.ApiKey,
        Scheme       = "Bearer",
        BearerFormat = "JWT",
        In           = ParameterLocation.Header,
        Description  = "Enter ‘Bearer’ [space] and your JWT
token.\n\nExample: \"Bearer eyJhbGciOiJI…\""
    });
    c.AddSecurityRequirement(new OpenApiSecurityRequirement {
        {
            new OpenApiSecurityScheme {
                Reference = new OpenApiReference {
                    Type = ReferenceType.SecurityScheme,
                    Id   = "Bearer"
                }
            },
            Array.Empty<string>()
        }
    });
});

// JWT Auth
var jwtKey      = builder.Configuration["Jwt:Key"]!;
var jwtIssuer   = builder.Configuration["Jwt:Issuer"]!;
var jwtAudience = builder.Configuration["Jwt:Audience"]!;
if (string.IsNullOrEmpty(jwtKey) ||
    string.IsNullOrEmpty(jwtIssuer) ||
    string.IsNullOrEmpty(jwtAudience))
    throw new InvalidOperationException("JWT config missing");

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              = jwtIssuer,
        ValidAudience            = jwtAudience,
        IssuerSigningKey         = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey))
    };
});

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllers();

app.Run();
```

---

5. Controllers

5.1 AuthController.cs

csharp
CopyEdit

```
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 _config;
        public AuthController(IConfiguration config) => _config =
config;

        [HttpPost("login")]
        public IActionResult Login([FromBody] UserLogin user)
        {
            if (user == null
             || string.IsNullOrWhiteSpace(user.Username)
             || string.IsNullOrWhiteSpace(user.Password))
                return BadRequest("Username and password required");

            if (user.Username == "admin" && user.Password == "1234")
            {
                return Ok(new { token =
GenerateJwtToken(user.Username) });
            }
            return Unauthorized("Invalid credentials");
        }

        private string GenerateJwtToken(string username)
        {
            var key       = _config["Jwt:Key"]!;
            var issuer    = _config["Jwt:Issuer"]!;
            var audience  = _config["Jwt:Audience"]!;
            var securityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(key));
            var creds       = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);
```

```
            var claims = new[]
            {
                new Claim(JwtRegisteredClaimNames.Sub, username),
                new Claim(JwtRegisteredClaimNames.Jti,
Guid.NewGuid().ToString())
            };

            var token = new JwtSecurityToken(
                issuer:             issuer,
                audience:           audience,
                claims:             claims,
                expires:            DateTime.UtcNow.AddMinutes(30),
                signingCredentials: creds
            );

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

    public class UserLogin
    {
        public string Username { get; set; } = "";
        public string Password { get; set; } = "";
    }
}
```

5.2 WeatherController.cs (Protected Endpoint)

csharp
CopyEdit

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class WeatherController : ControllerBase
    {
        [HttpGet("secure")]
        [Authorize]
        public IActionResult GetSecureWeather()
            => Ok(new { message = "This is a protected endpoint!"
});
    }
}
```

---

6. Testing in Swagger

1. **Run** the app (`dotnet run`) → open `http://localhost:5119/swagger`.
2. **POST** `/api/Auth/login`

Click **Try it out**, paste:

json
CopyEdit
```
{ "username":"admin", "password":"1234" }
```

   ○
   ○ Click **Execute** → copy the returned `token`.

Click **Authorize**, paste:

php-template
CopyEdit
```
Bearer <your-token>
```

3. → click **Authorize** → **Close**.

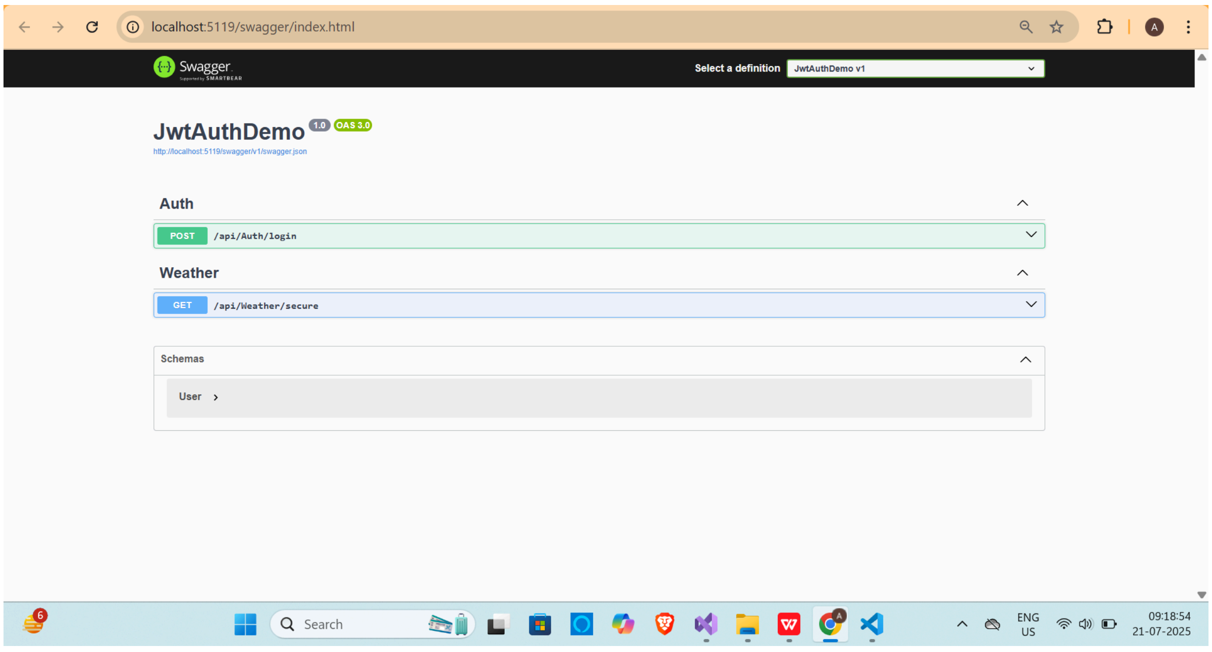
GET `/api/Weather/secure` → **Execute** → you should see:

json
CopyEdit
```
{ "message": "This is a protected endpoint!" }
```

4.

---

7. Screenshots



localhost:5119/swagger/index.html

Select a definition: JwtAuthDemo v1

JwtAuthDemo 1.0 OAS 3.0

http://localhost:5119/swagger/v1/swagger.json

Auth

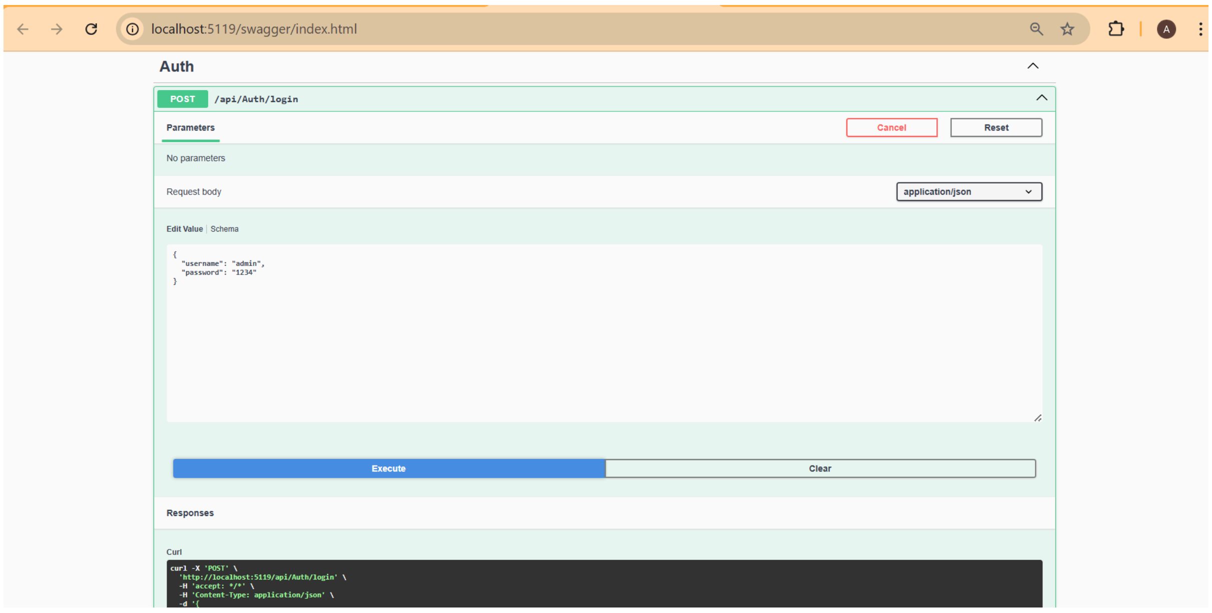
POST /api/Auth/login

Weather

GET /api/Weather/secure

Schemas

User



localhost:5119/swagger/index.html

Auth

POST /api/Auth/login

Parameters

No parameters

Request body: application/json

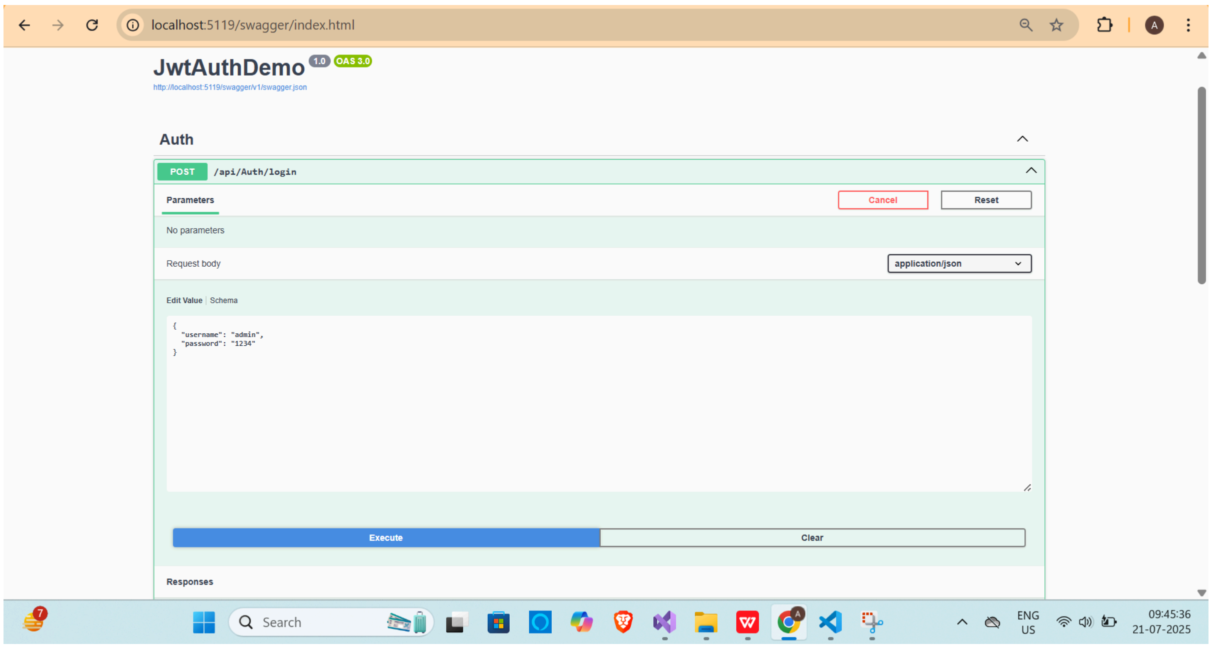
Edit Value | Schema

```
{
  "username": "admin",
  "password": "1234"
}
```

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5119/api/Auth/login' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
```



JwtAuthDemo

http://localhost:5119/swagger/v1/swagger.json

Auth

POST /api/Auth/login

Parameters

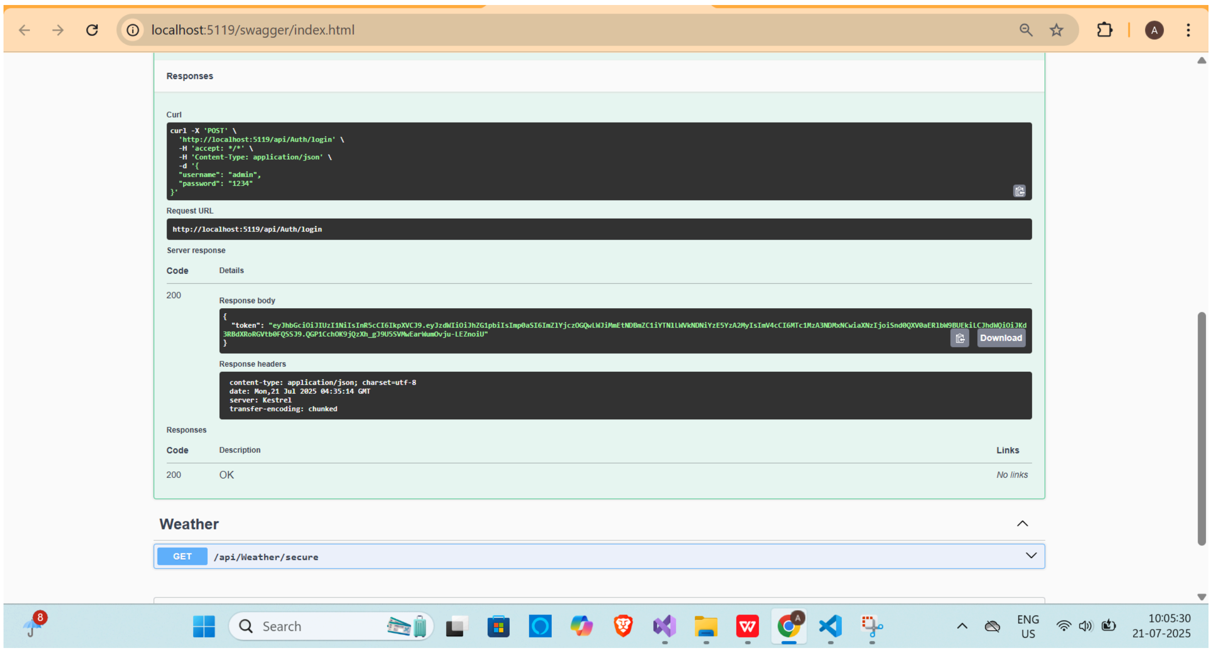
No parameters

Request body

Edit Value | Schema

```
{
  "username": "admin",
  "password": "1234"
}
```

Responses



Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5119/api/Auth/login' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "username": "admin",
  "password": "1234"
}'
```

Request URL

```
http://localhost:5119/api/Auth/login
```

Server response

Code | Details

200

Response body

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJhZG1pbiIsImp0aSI6ImZlYjczOGQwLWJiMmEtNDBmZC1iYTNlLWVkNDNiYzE5YzA2MyIsImV4cCI6MTc1MzA3NDMxNCwiaXNzIjoiSnd0QXV0aERlbW9BUEkiLCJhdWQiOiJKd
3RBdXRoRGVtb0FQSSJ9.QGP1CchOK9jQzXh_gJ9U5SVMwEarWumOvju-LEZnoiU"
}
```

Response headers

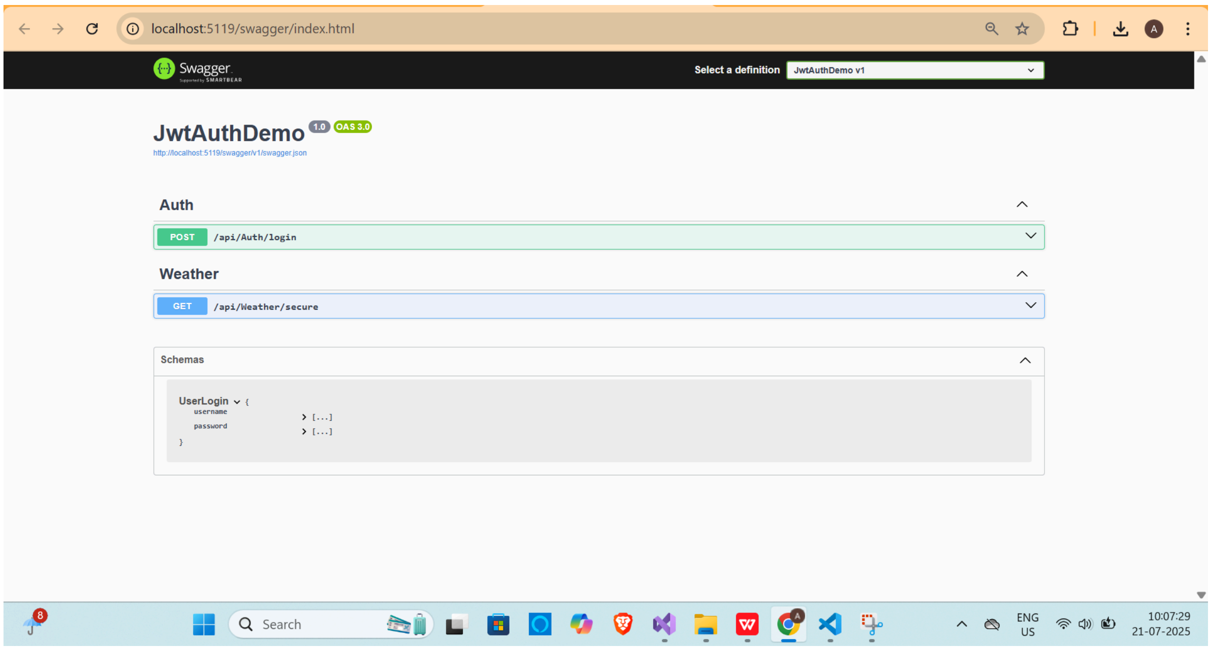
```
 content-type: application/json; charset=utf-8
 date: Mon,21 Jul 2025 04:35:14 GMT
 server: Kestrel
 transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Weather

GET /api/Weather/secure



JwtAuthDemo 1.0 OAS 3.0

http://localhost:5119/swagger/v1/swagger.json

Auth

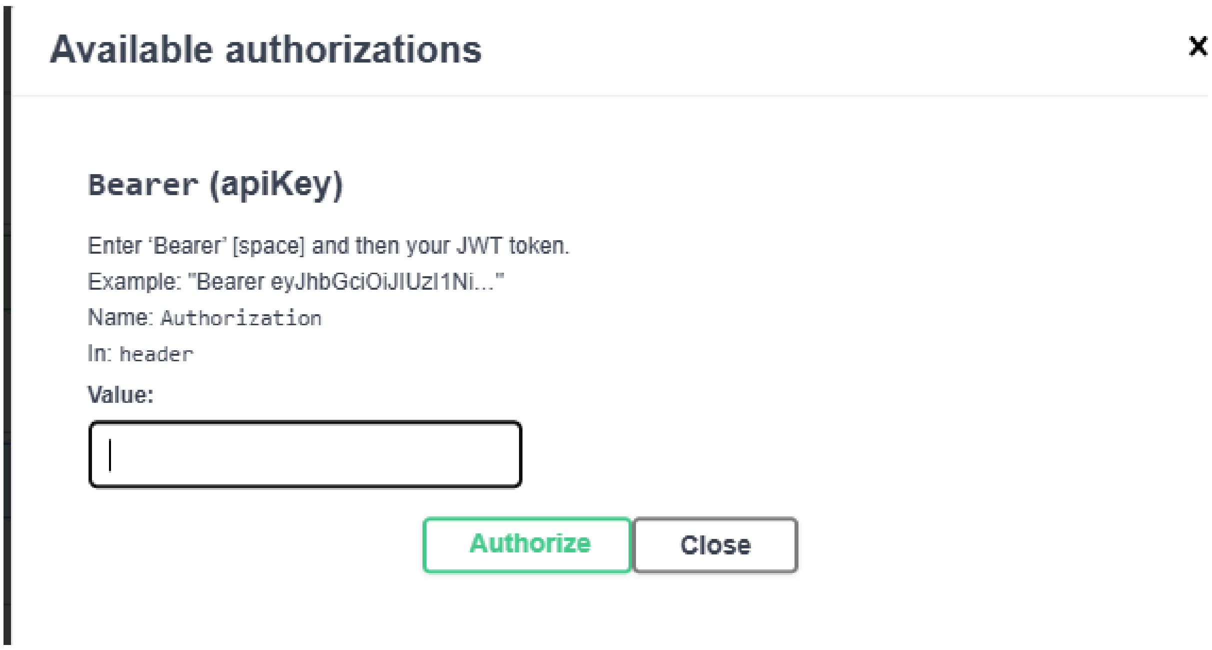
POST /api/Auth/login

Weather

GET /api/Weather/secure

Schemas

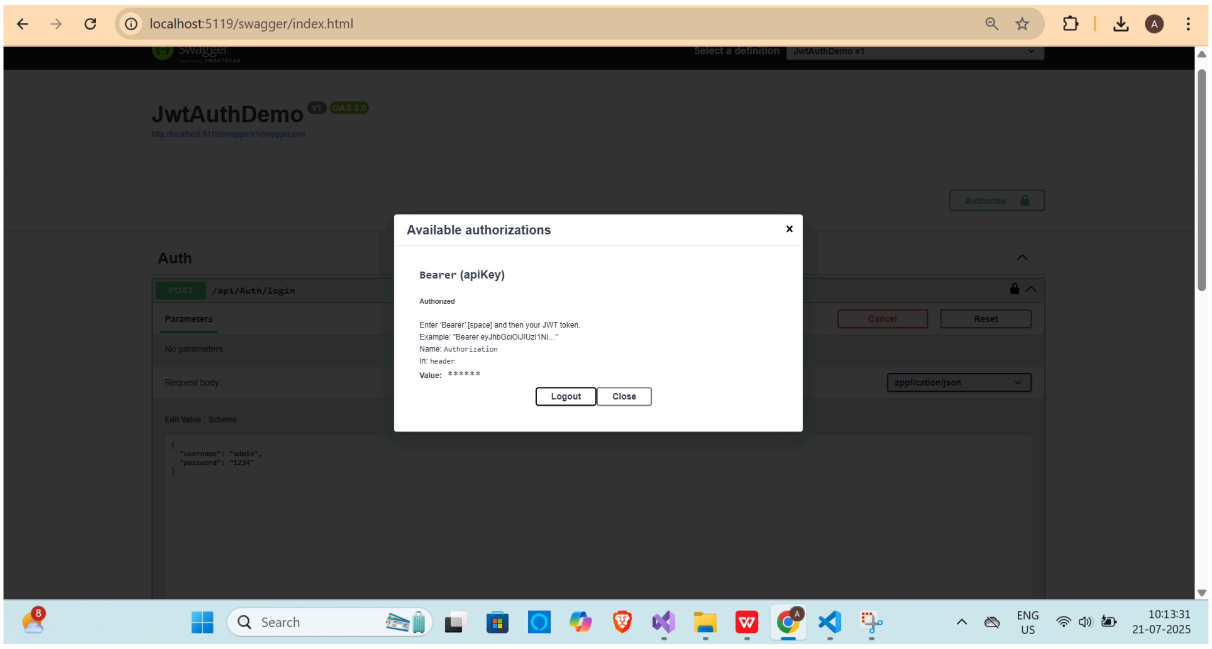
```
UserLogin {
    username        > [...]
    password        > [...]
}
```



Available authorizations

Bearer (apiKey)

Enter 'Bearer' [space] and then your JWT token.
Example: "Bearer eyJhbGciOiJIUzI1Ni..."
Name: `Authorization`
In: `header`
Value:



Available authorizations

Bearer (apiKey)

Authorized

Enter 'Bearer' [space] and then your JWT token.
Example: "Bearer eyJhbGciOiJIUzI1Ni..."
Name: Authorization
In: header
Value: ******

```
{
  "username": "admin",
  "password": "1234"
}
```

---

8. Conclusion

- Configured JWT in ASP.NET Core
- Generated and validated tokens
- Protected endpoints with `[Authorize]`
- Integrated Swagger to issue and test tokens