**SuperSet Id: 6365365**

1. WebApi\_Handson

#### 1. ****Explain the concept of RESTful web service, Web API & Microservice****

**Includes:**

Features of REST architecture

Difference between WebService & WebAPI

Not restricted to XML

#### 2. ****Explain what is HttpRequest & HttpResponse****

can optionally use them in ValuesController like

var method = Request.Method;

Response.Headers.Add("X-Test", "From API");

#### 3. ****List the types of Action Verbs****

GET, POST, PUT, DELETE

#### 4. ****List the types of HttpStatusCodes used in WebAPI****

Ok

BadRequest

Unauthorized

InternalServerError

#### 5. ****Demonstrate creation of a simple WebAPI - With Read, Write actions****

✔️ created FirstWebApi project

✔️ ValuesController includes read/write (GET, POST, PUT, DELETE)

#### 6. ****Explain the types of Configuration files of WebAPI****

Includes:

Startup.cs with dependency injection

appsettings.json, launchSettings.json

**ValuesController.cs**

using Microsoft.AspNetCore.Mvc;

namespace FirstWebApi.Controllers

{

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

[ApiController]

public class ValuesController : ControllerBase

{

// GET: api/<ValuesController>

[HttpGet]

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

// GET api/<ValuesController>/5

[HttpGet("{id}")]

public string Get(int id)

{

return "value";

}

// POST api/<ValuesController>

[HttpPost]

public void Post([FromBody] string value)

{

}

// PUT api/<ValuesController>/5

[HttpPut("{id}")]

public void Put(int id, [FromBody] string value)

{

}

// DELETE api/<ValuesController>/5

[HttpDelete("{id}")]

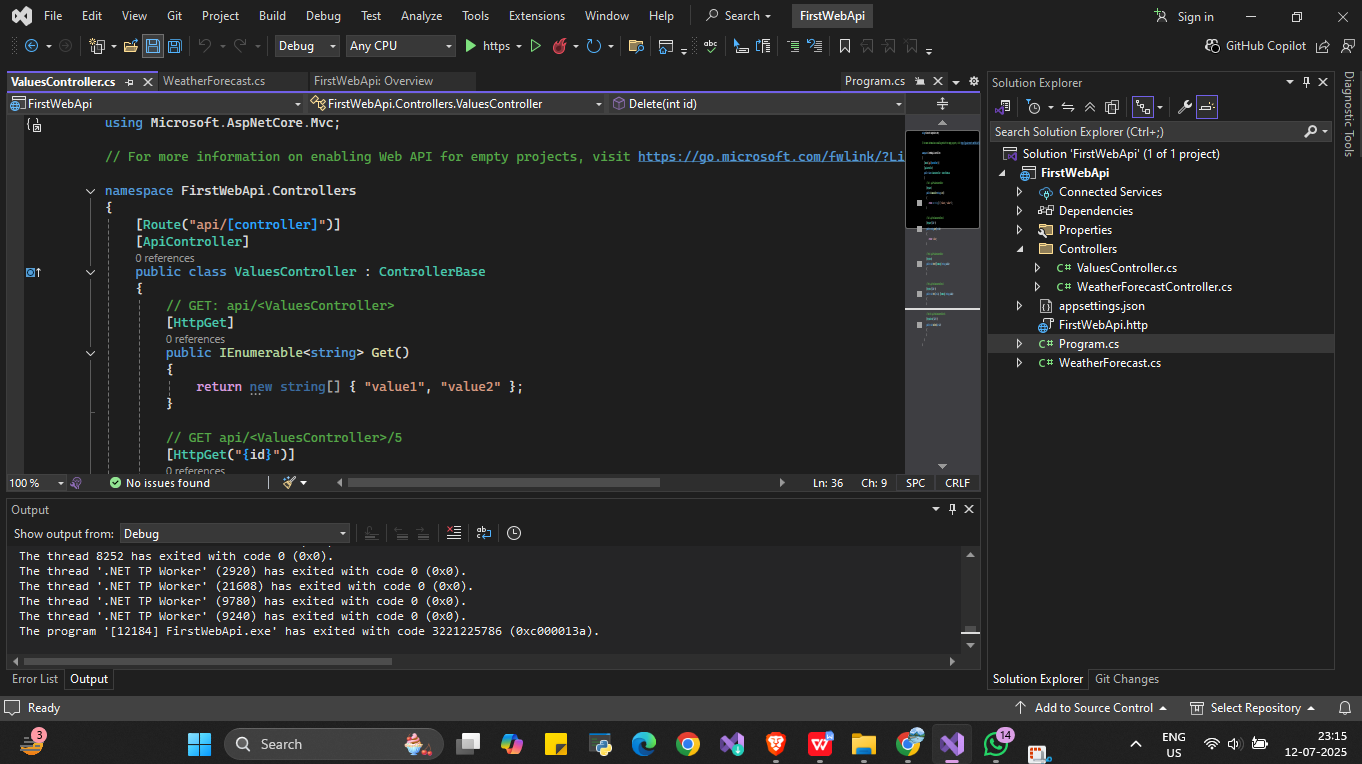
public void Delete(int id)

{

}

}

}



**Program.cs**

var builder = WebApplication.CreateBuilder(args);

// Add services to the container

builder.Services.AddControllers();

var app = builder.Build();

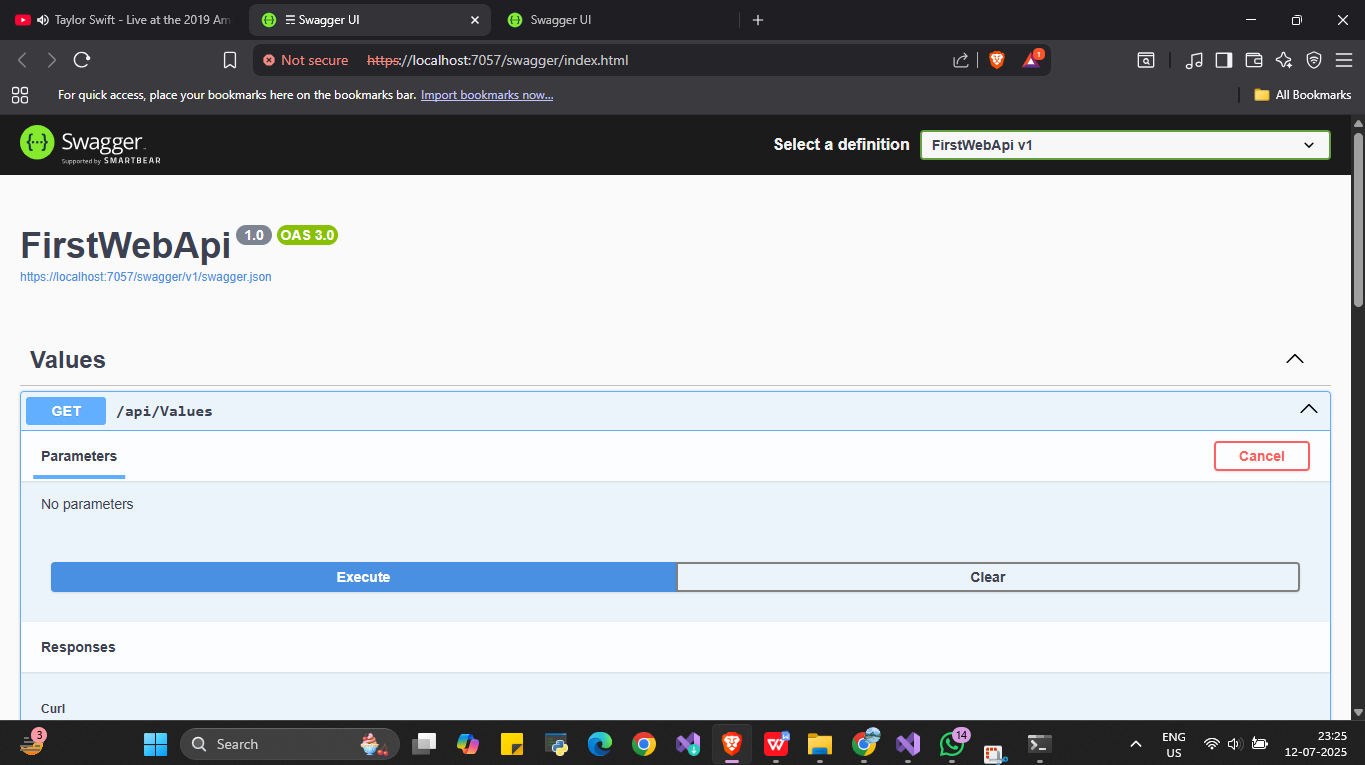
// Configure the HTTP request pipeline

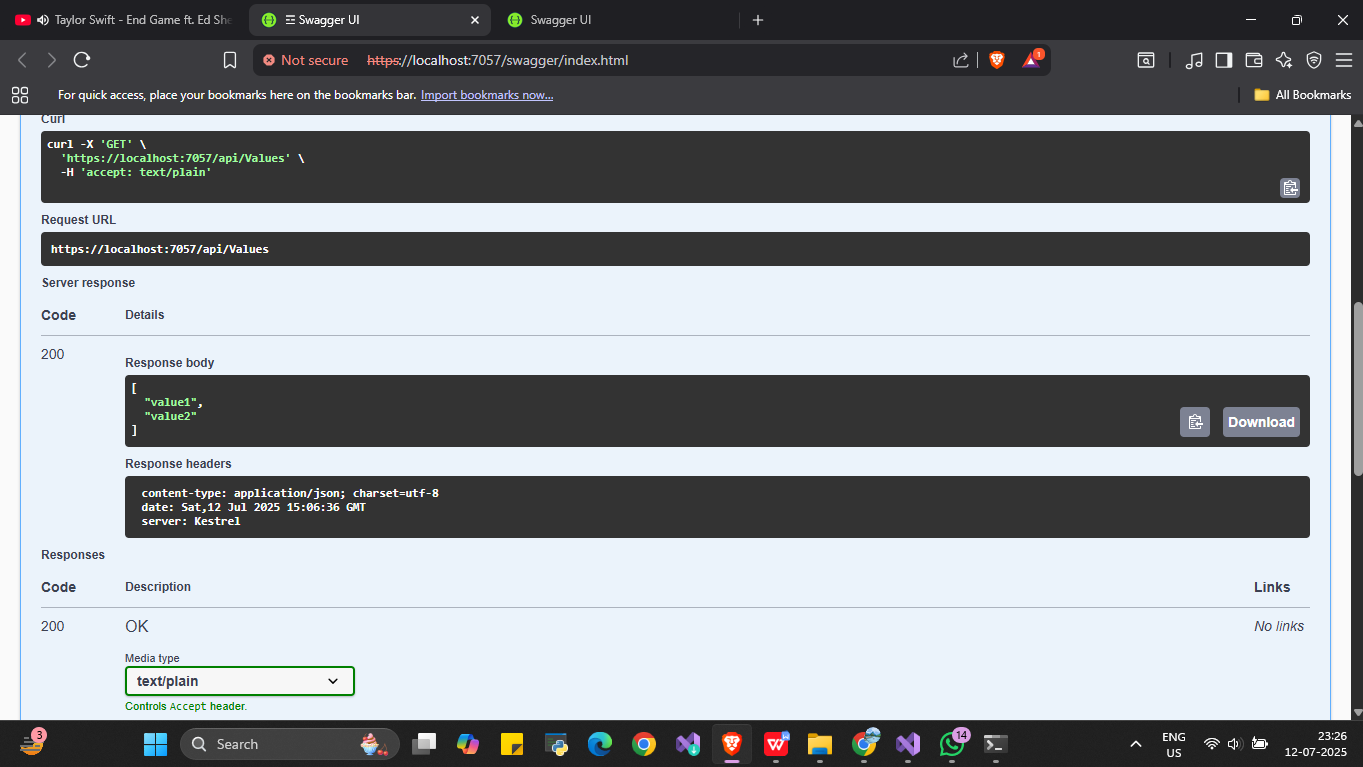
app.UseHttpsRedirection();

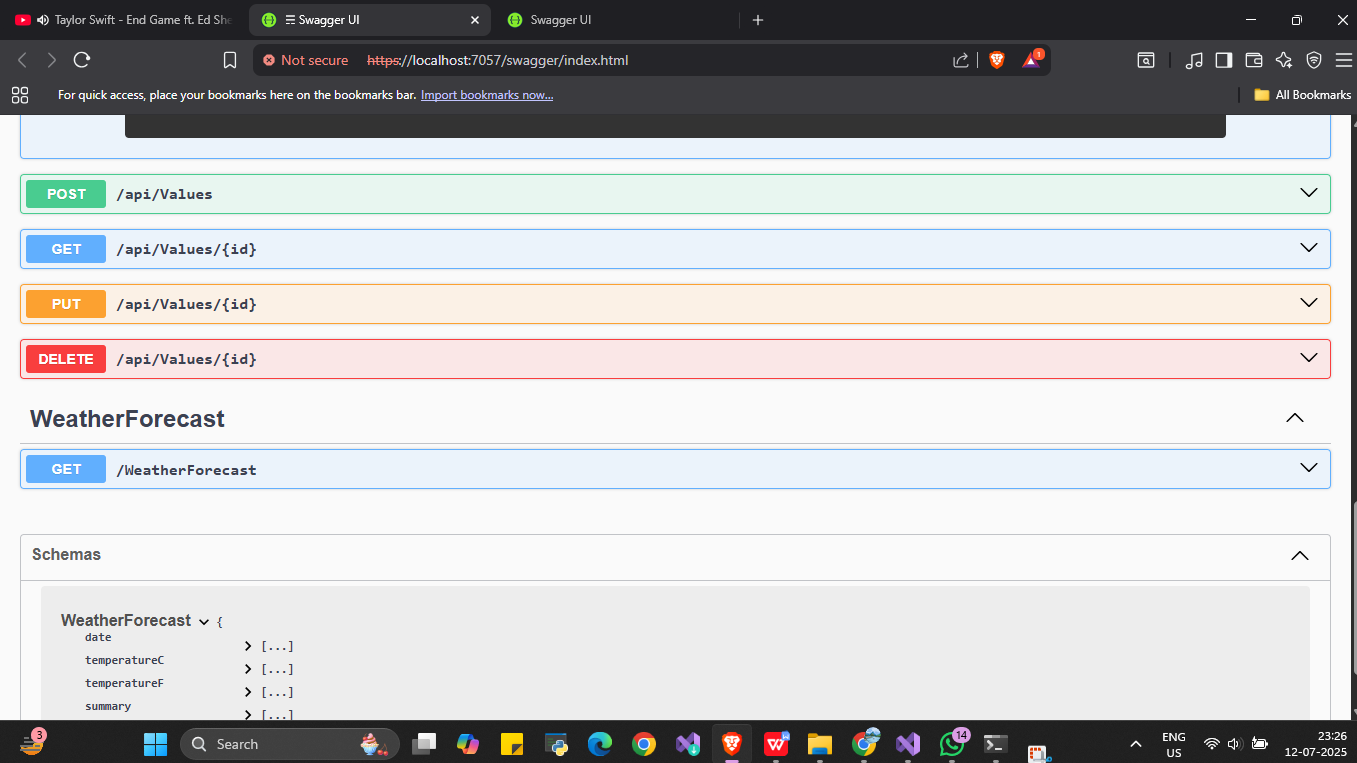
app.UseAuthorization();

app.MapControllers();

app.Run();







1. WebApi\_Handson

Program.cs

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Demo Swagger for FirstWebApi",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

Url = new Uri("https://www.example.com")

},

License = new OpenApiLicense

{

Name = "License Terms",

Url = new Uri("https://www.example.com")

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");

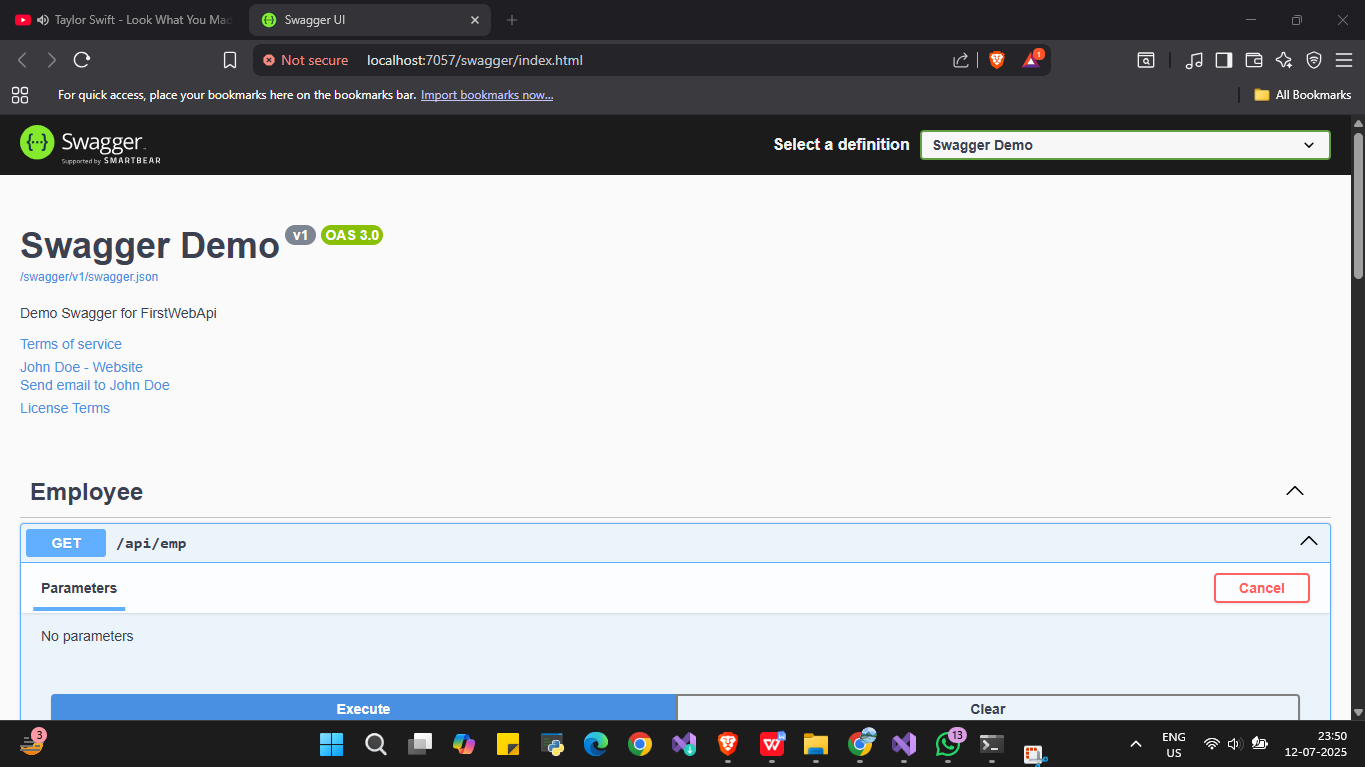
});

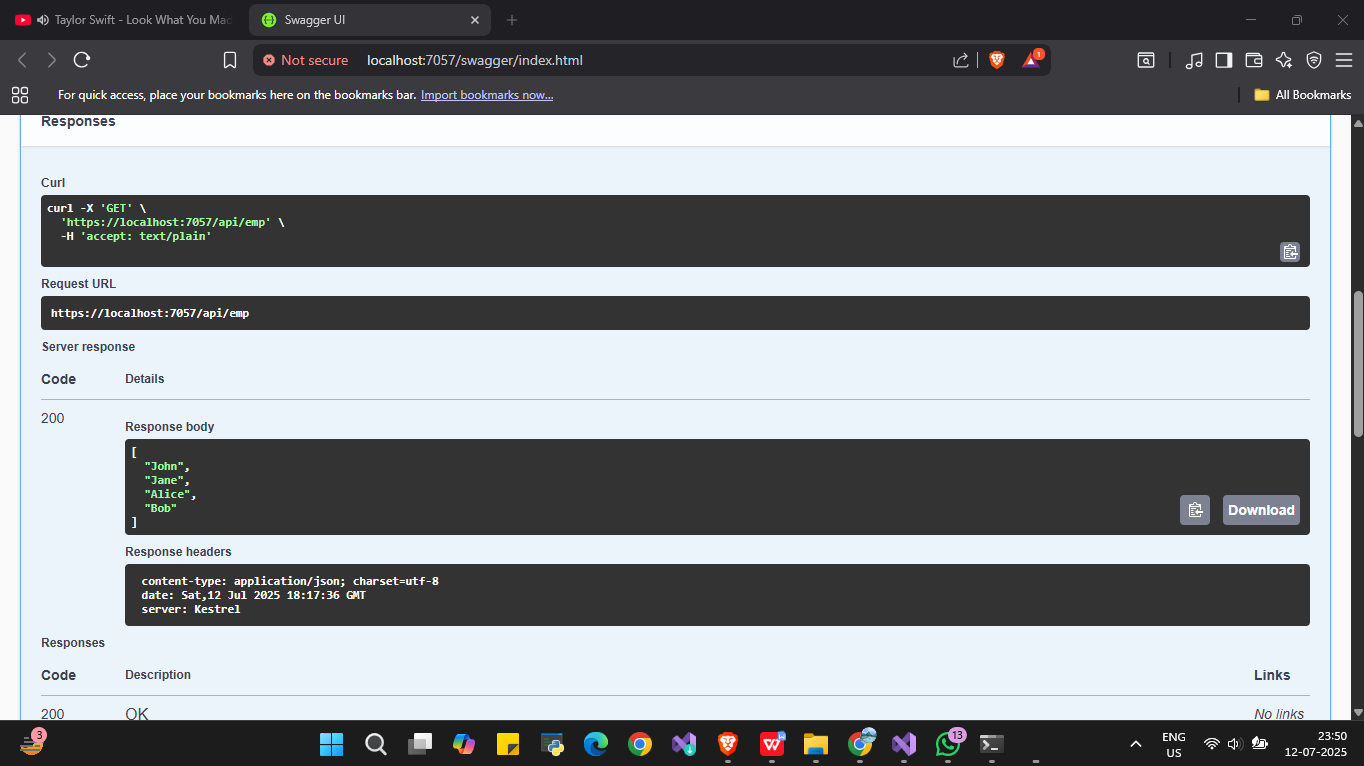
app.UseHttpsRedirection();

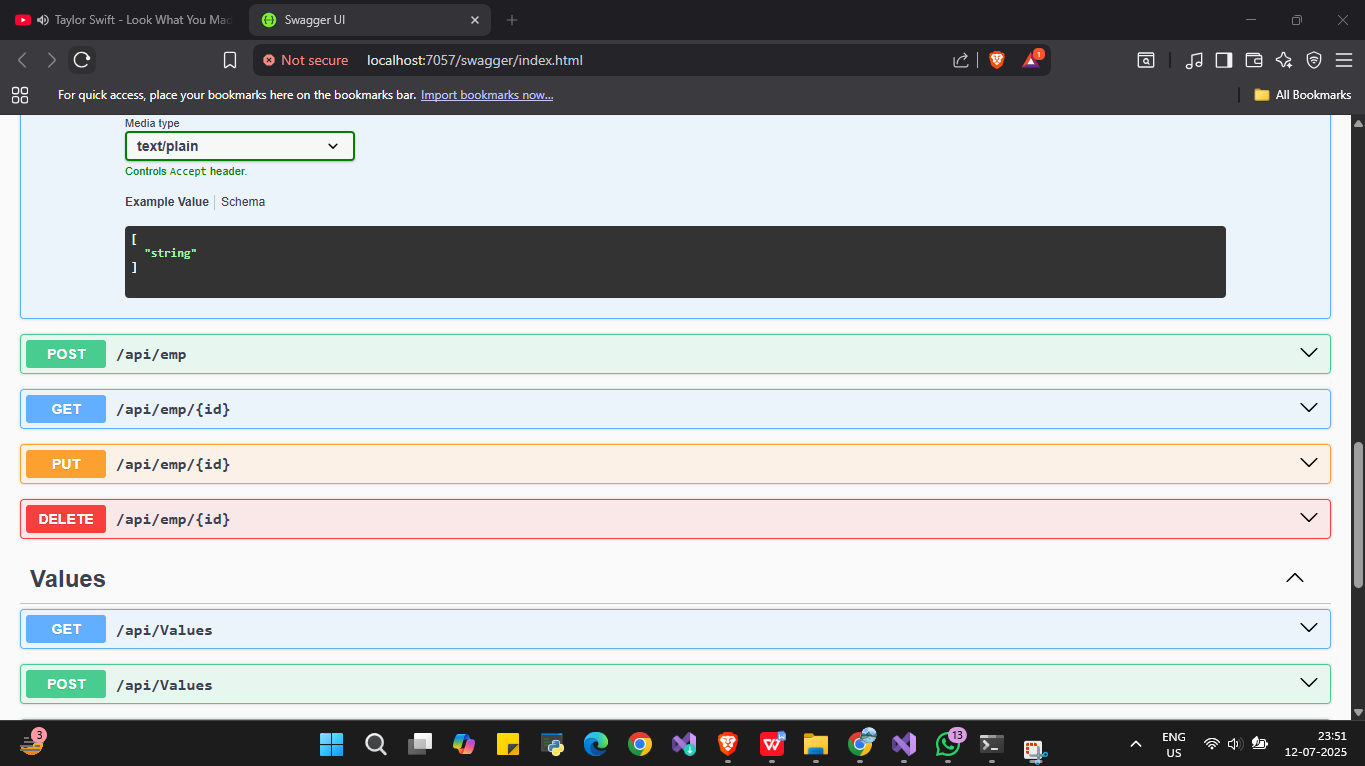
app.UseAuthorization();

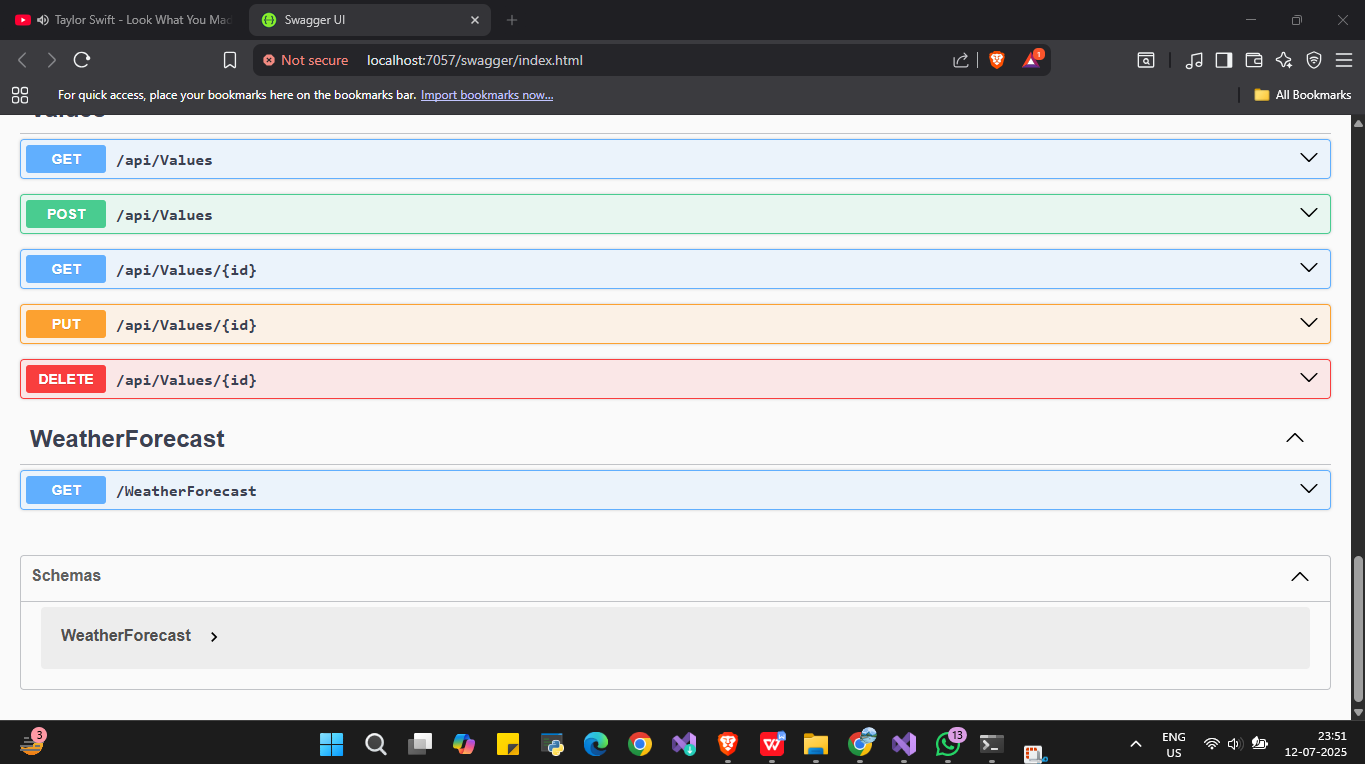
app.MapControllers();

app.Run();









**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace FirstWebApi.Controllers

{

[ApiController]

[Route("api/emp")]

public class EmployeeController : ControllerBase

{

private static List<string> employees = new List<string> { "John", "Jane", "Alice", "Bob" };

[HttpGet]

[ActionName("GetAll")]

public ActionResult<IEnumerable<string>> GetAllEmployees()

{

return Ok(employees);

}

[HttpGet("{id}")]

[ActionName("GetById")]

public ActionResult<string> GetEmployeeById(int id)

{

if (id < 0 || id >= employees.Count)

return NotFound("Employee not found");

return Ok(employees[id]);

}

[HttpPost]

public ActionResult AddEmployee([FromBody] string name)

{

employees.Add(name);

return Ok("Employee added");

}

[HttpPut("{id}")]

public ActionResult UpdateEmployee(int id, [FromBody] string name)

{

if (id < 0 || id >= employees.Count)

return BadRequest("Invalid employee ID");

employees[id] = name;

return Ok("Employee updated");

}

[HttpDelete("{id}")]

public ActionResult DeleteEmployee(int id)

{

if (id < 0 || id >= employees.Count)

return BadRequest("Invalid employee ID");

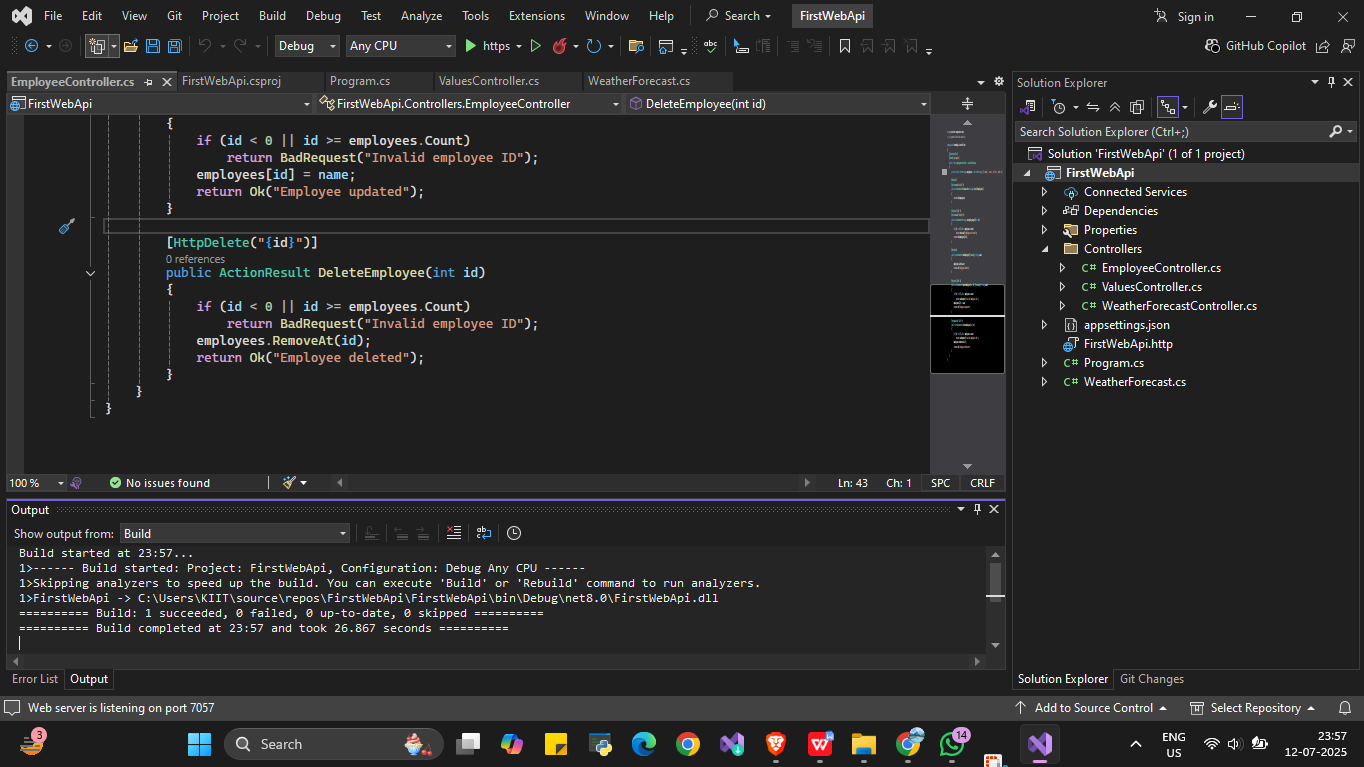
employees.RemoveAt(id);

return Ok("Employee deleted");

}

}

}



1. **WebApi\_Handson**

**Models**

**Department.cs**

namespace CustomApiDemo.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Employee.cs**

using System;

using System.Collections.Generic;

namespace CustomApiDemo.Models

{

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public bool Permanent { get; set; }

public Department Department { get; set; }

public List<Skill> Skills { get; set; }

public DateTime DateOfBirth { get; set; }

}

}

**Skill.cs**

namespace CustomApiDemo.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Filters**

**CustomAuthFilters.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace CustomApiDemo.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

if (!context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token))

{

context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

return;

}

if (!token.ToString().Contains("Bearer"))

{

context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

}

}

}

}

**CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

namespace CustomApiDemo.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var message = $"Exception occurred: {context.Exception.Message}";

File.AppendAllText("logs.txt", $"{DateTime.Now}: {message}\n");

context.Result = new ObjectResult("An unexpected error occurred")

{

StatusCode = 500

};

}

}

}

**Controllers**

**EmployeeController.cs**

using CustomApiDemo.Filters;

using CustomApiDemo.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace CustomApiDemo.Controllers

{

[ApiController]

[Route("[controller]")]

[AllowAnonymous] // Allow Swagger access without auth token

[CustomAuthFilter] // Apply custom authorization filter here

public class EmployeeController : ControllerBase

{

private List<Employee> \_employees;

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

// Private method to generate some employees

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 60000,

Permanent = true,

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "SQL" }

},

DateOfBirth = new DateTime(1990, 01, 01)

},

new Employee

{

Id = 2,

Name = "Jane",

Salary = 75000,

Permanent = false,

Department = new Department { Id = 2, Name = "Finance" },

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "Excel" },

new Skill { Id = 4, Name = "Power BI" }

},

DateOfBirth = new DateTime(1988, 05, 10)

}

};

}

// GET: /employee/getstandard

[HttpGet("GetStandard")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

// Uncomment below line to test exception filter

// throw new Exception("Simulated exception for testing");

return Ok(\_employees);

}

// POST: /employee

[HttpPost]

public ActionResult PostEmployee([FromBody] Employee emp)

{

\_employees.Add(emp);

return Ok(emp);

}

// PUT: /employee

[HttpPut]

public ActionResult PutEmployee([FromBody] Employee emp)

{

var existing = \_employees.FirstOrDefault(e => e.Id == emp.Id);

if (existing != null)

{

existing.Name = emp.Name;

existing.Salary = emp.Salary;

existing.Permanent = emp.Permanent;

existing.Department = emp.Department;

existing.Skills = emp.Skills;

existing.DateOfBirth = emp.DateOfBirth;

return Ok(existing);

}

return NotFound($"Employee with ID {emp.Id} not found");

}

}

}

**WeatherForecastController.cs**

using Microsoft.AspNetCore.Mvc;

namespace CustomApiDemo.Controllers

{

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

[HttpGet(Name = "GetWeatherForecast")]

public IEnumerable<WeatherForecast> Get()

{

return Enumerable.Range(1, 5).Select(index => new WeatherForecast

{

Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),

TemperatureC = Random.Shared.Next(-20, 55),

Summary = Summaries[Random.Shared.Next(Summaries.Length)]

})

.ToArray();

}

}

}

**Program.cs**

using CustomApiDemo.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers(options =>

{

// Register the custom exception filter globally

options.Filters.Add<CustomExceptionFilter>();

});

// Add Swagger (OpenAPI) support

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

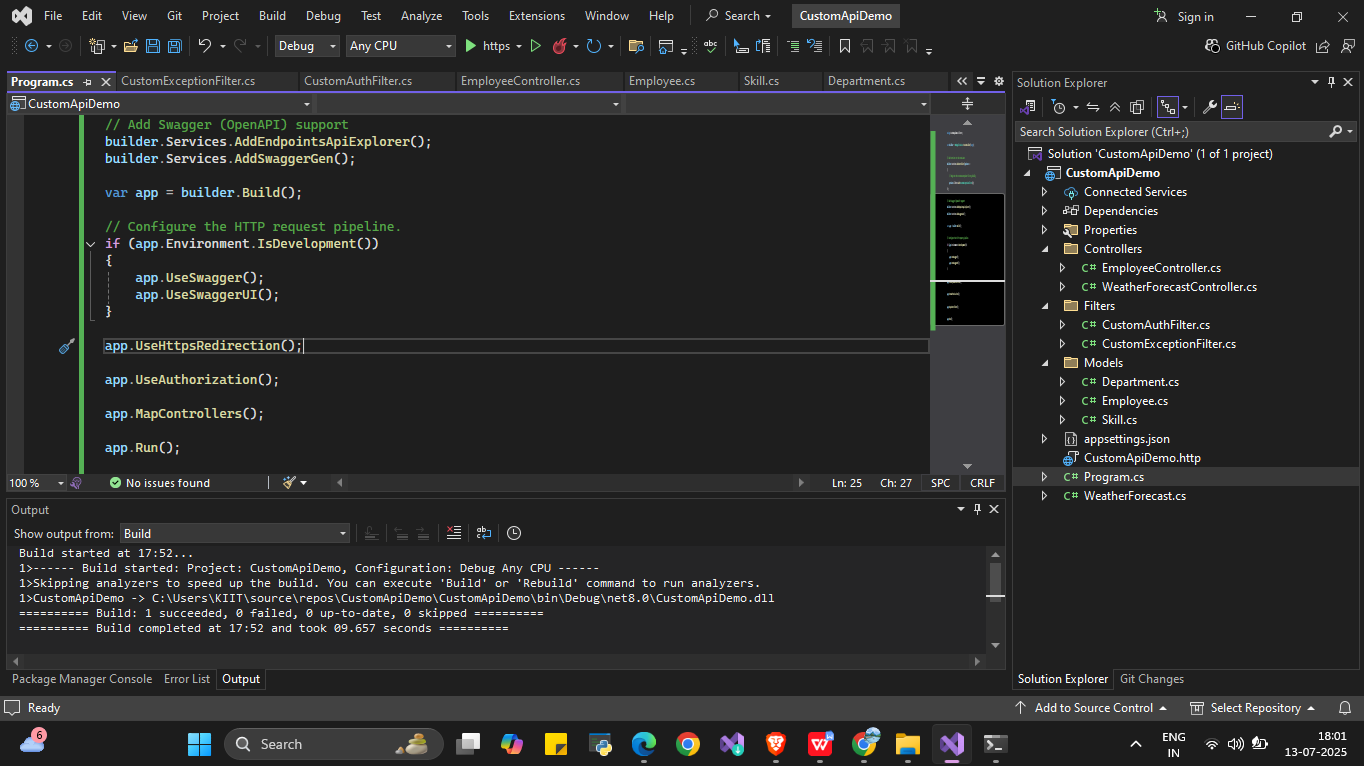
}

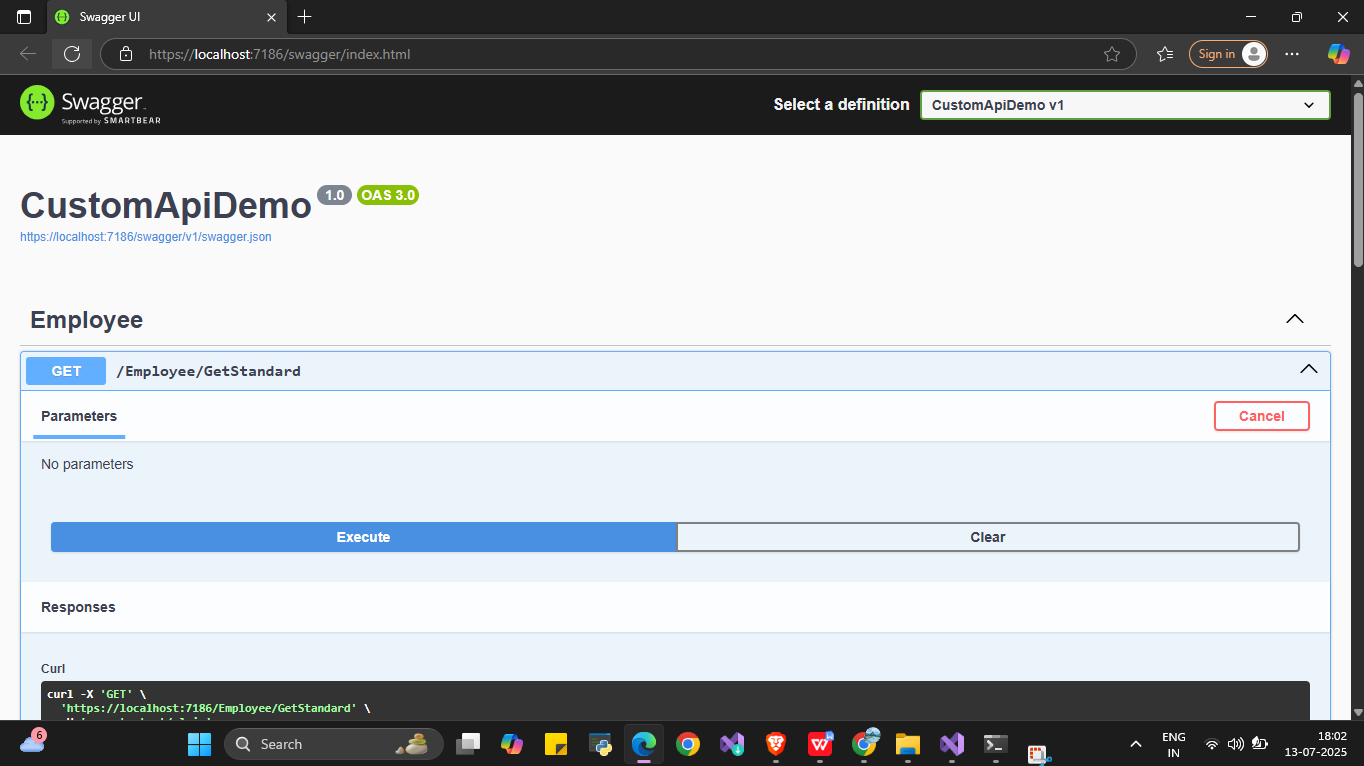
app.UseHttpsRedirection();

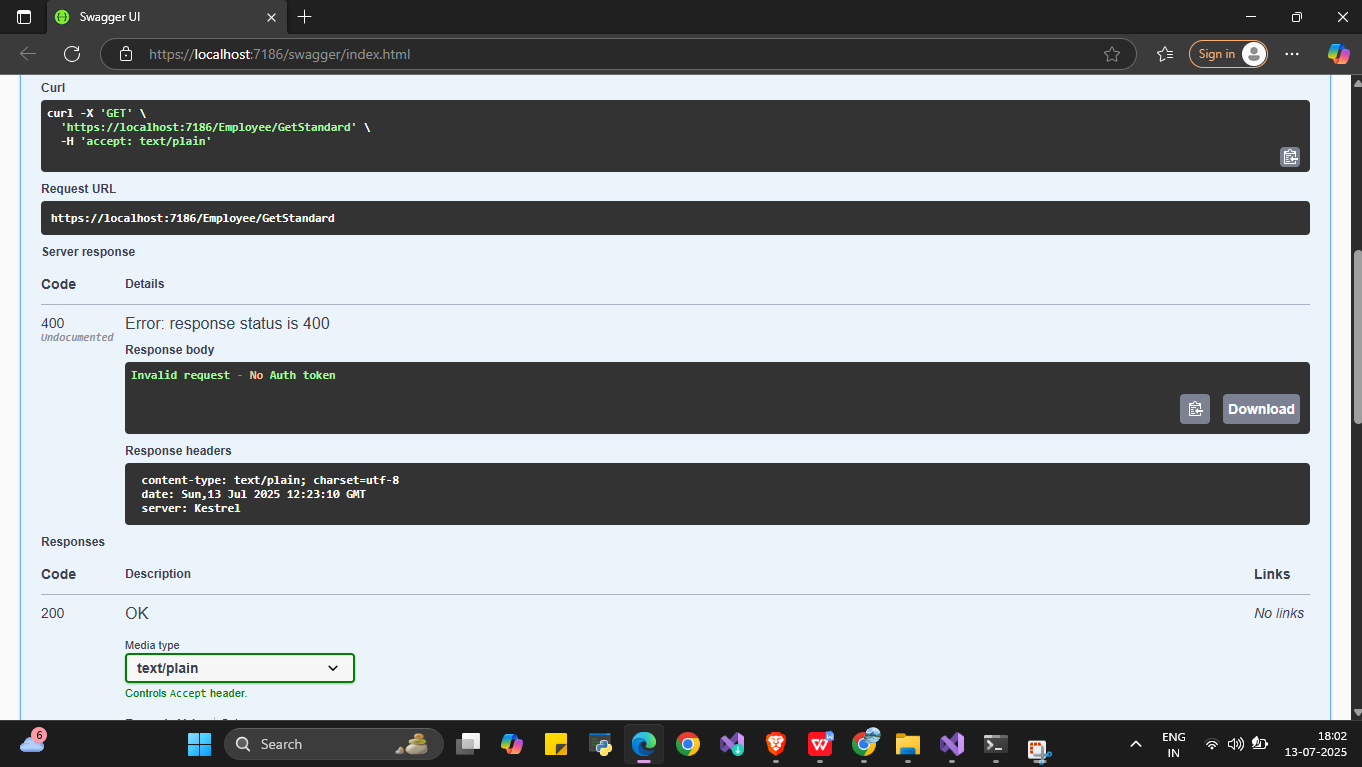
app.UseAuthorization();

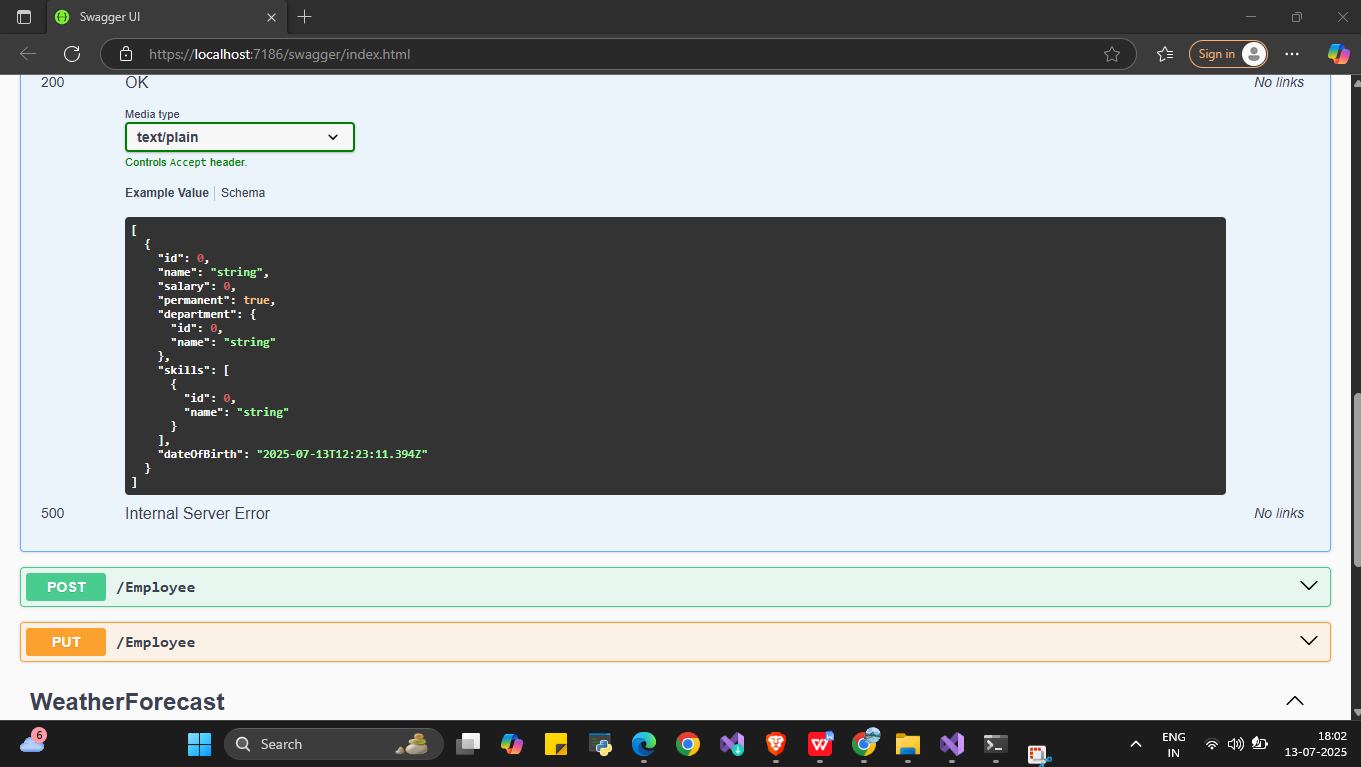
app.MapControllers();

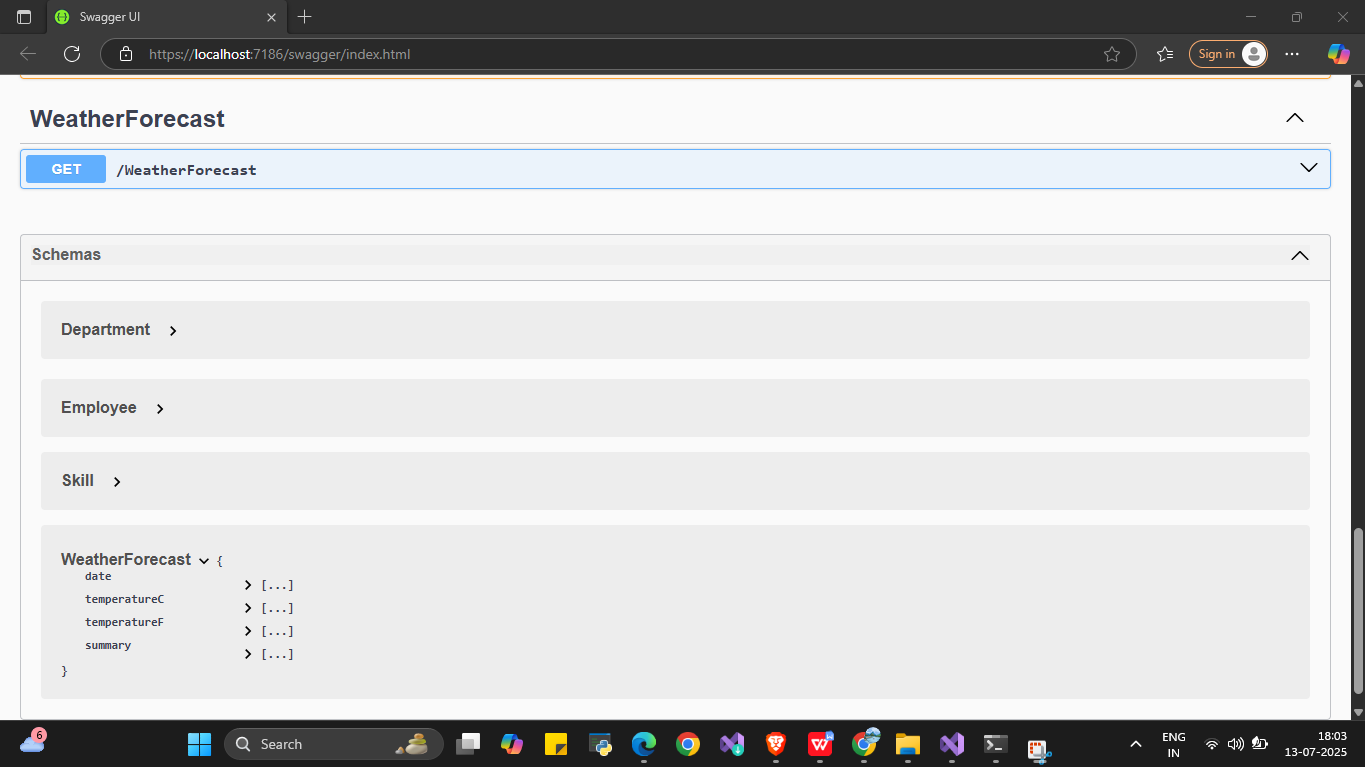
app.Run();











1. **WebApi\_ Handson**

**Models**

**Employee.cs**

namespace EmployeeWebApi.Models

{

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public string Department { get; set; }

}

}

**Controllers**

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using EmployeeWebApi.Models;

using System.Collections.Generic;

using System.Linq;

namespace EmployeeWebApi.Controllers

{

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

[ApiController]

public class EmployeeController : ControllerBase

{

// Hardcoded list of employees

private static List<Employee> employees = new List<Employee>

{

new Employee { Id = 1, Name = "John", Department = "IT" },

new Employee { Id = 2, Name = "Alice", Department = "HR" },

new Employee { Id = 3, Name = "Bob", Department = "Finance" }

};

// PUT: api/Employee/2

[HttpPut("{id}")]

public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updatedEmployee)

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

var existingEmployee = employees.FirstOrDefault(e => e.Id == id);

if (existingEmployee == null)

{

return BadRequest("Invalid employee id");

}

// Update employee details

existingEmployee.Name = updatedEmployee.Name;

existingEmployee.Department = updatedEmployee.Department;

return Ok(existingEmployee);

}

}

}

**Program.cs**

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

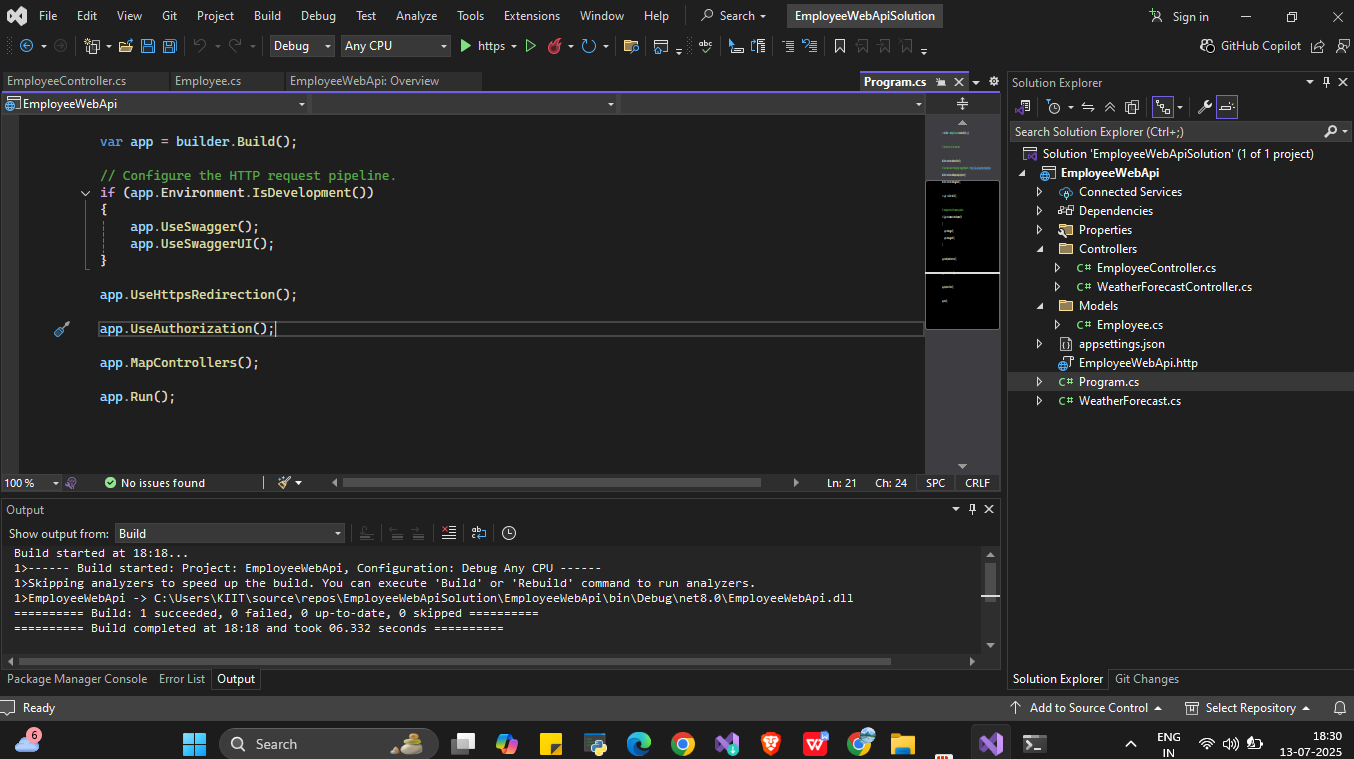
}

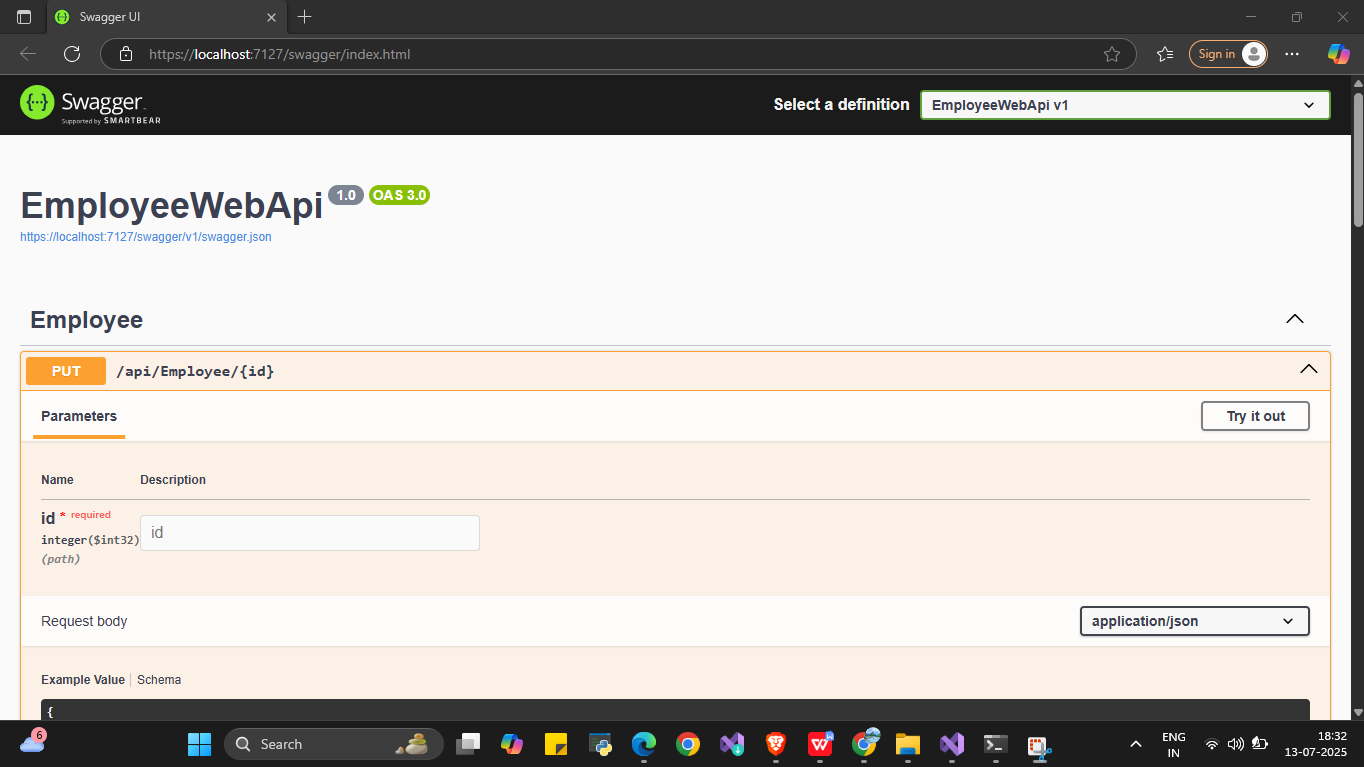
app.UseHttpsRedirection();

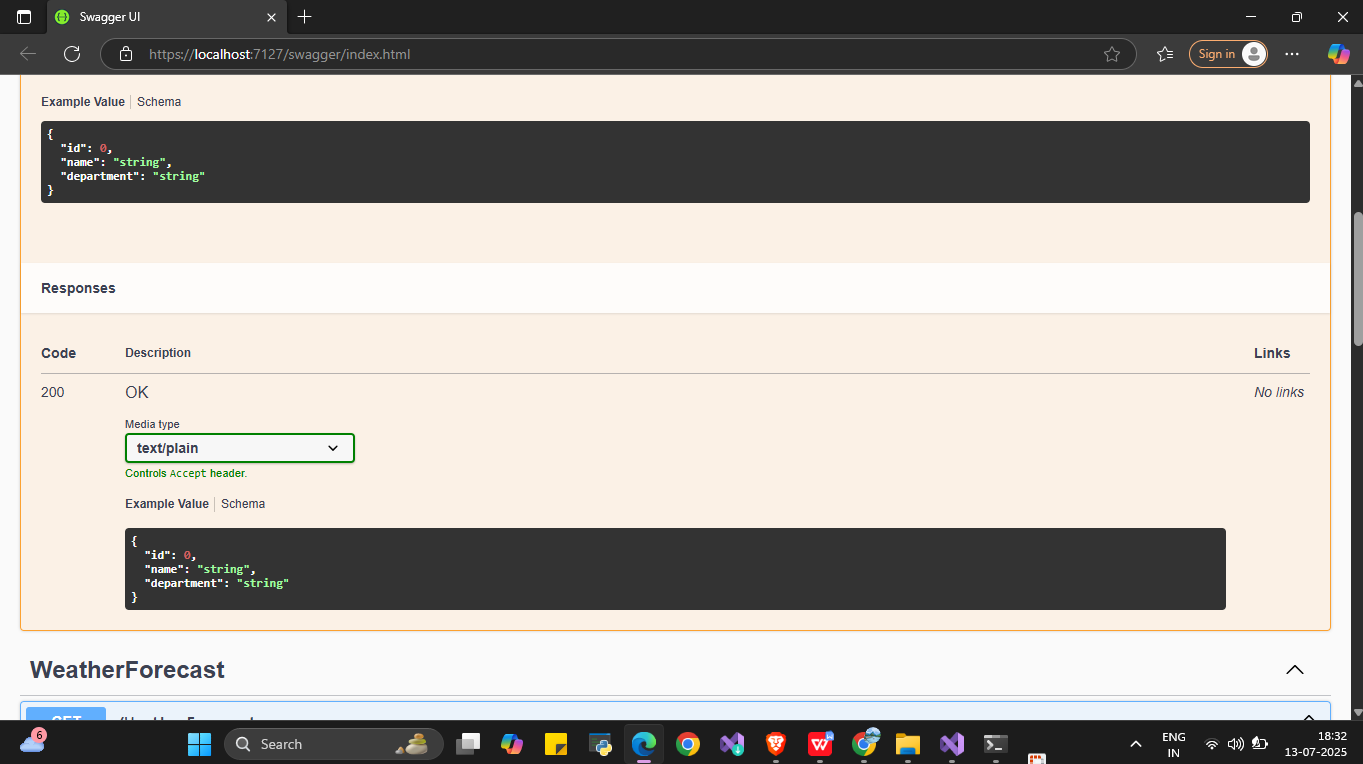
app.UseAuthorization();

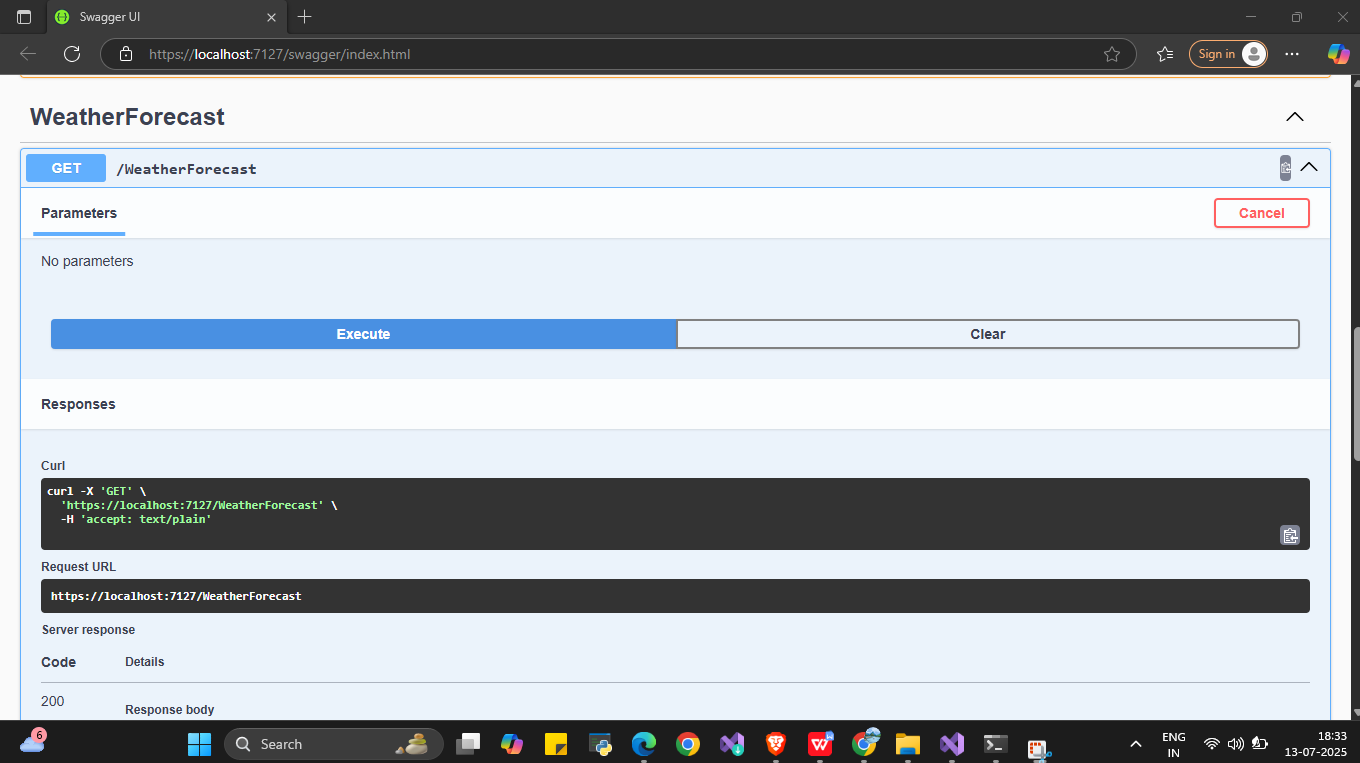
app.MapControllers();

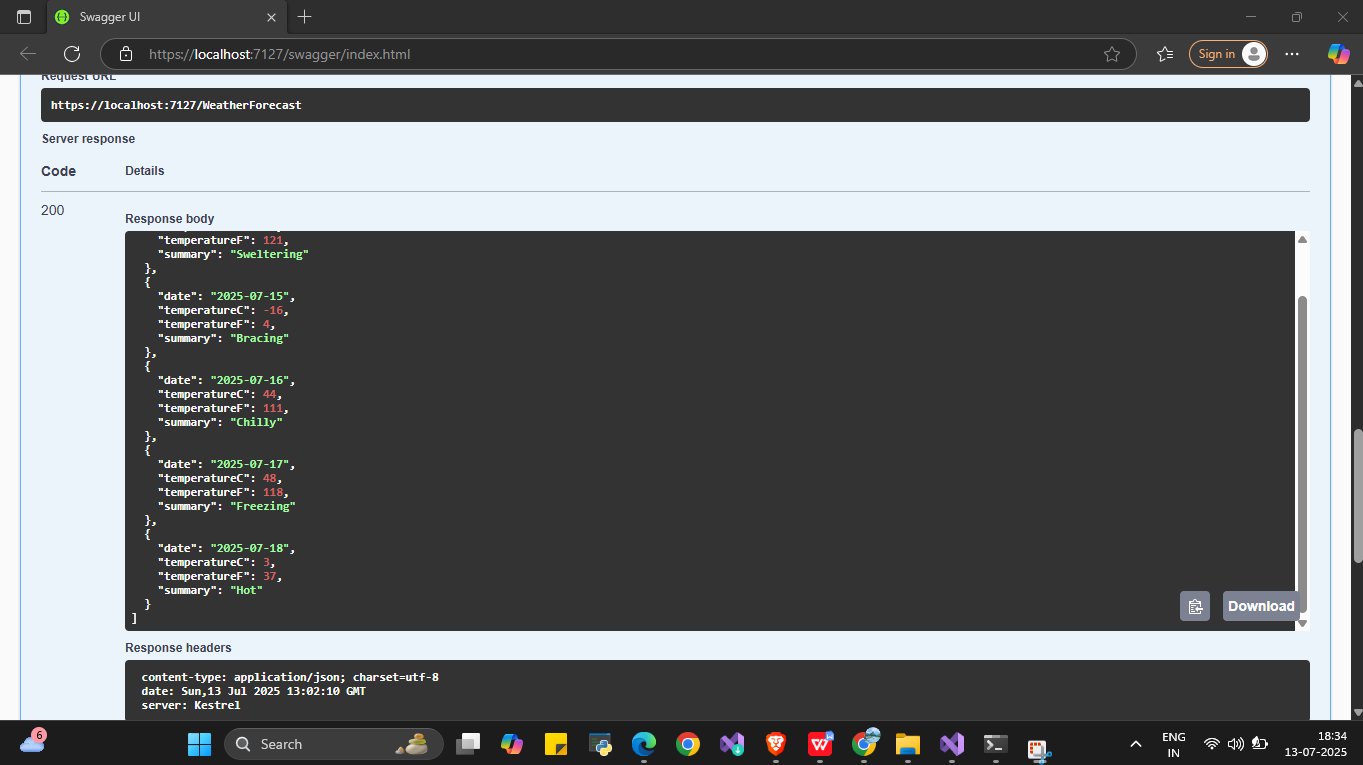
app.Run();











1. **WebApi\_HandsOn**

**CognizantApi**

**Controllers**

**AuthController.cs**

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

namespace CognizantApi.Controllers

{

[AllowAnonymous]

[ApiController]

[Route("[controller]")]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(101, "Admin"); // Example userId & role

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecret"));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, userRole),

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(2), // Short expiry for testing

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

string securityKey = "mysuperdupersecret";

var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowLocalhost",

builder =>

{

builder.WithOrigins("http://localhost:4200")

.AllowAnyHeader()

.AllowAnyMethod();

});

});

builder.Services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

builder.Services.AddControllers();

var app = builder.Build();

app.UseRouting();

app.UseCors("AllowLocalhost");

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

