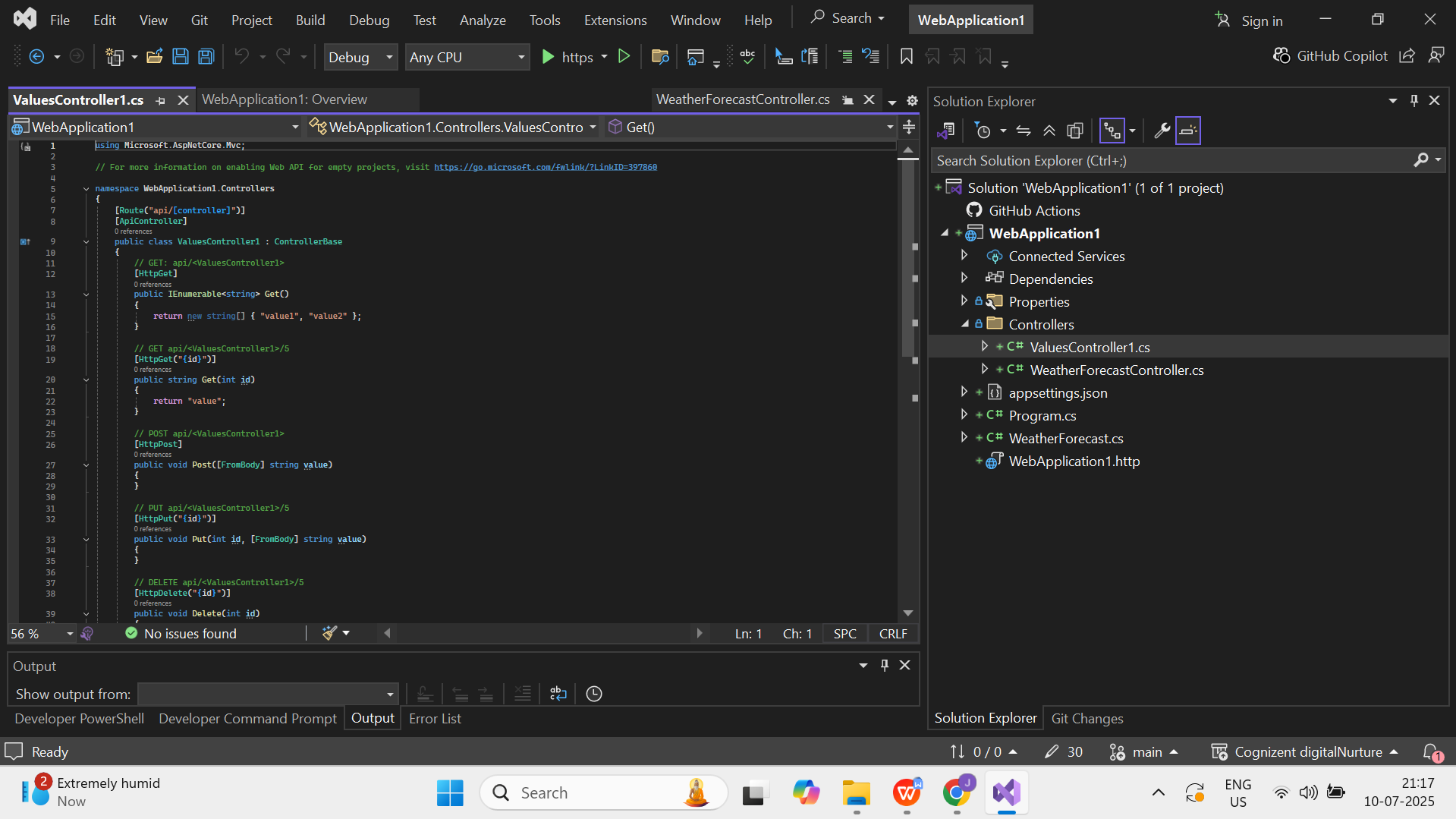
**Digital Nurture 4.0 .NET FSE (Week-4 Report)**

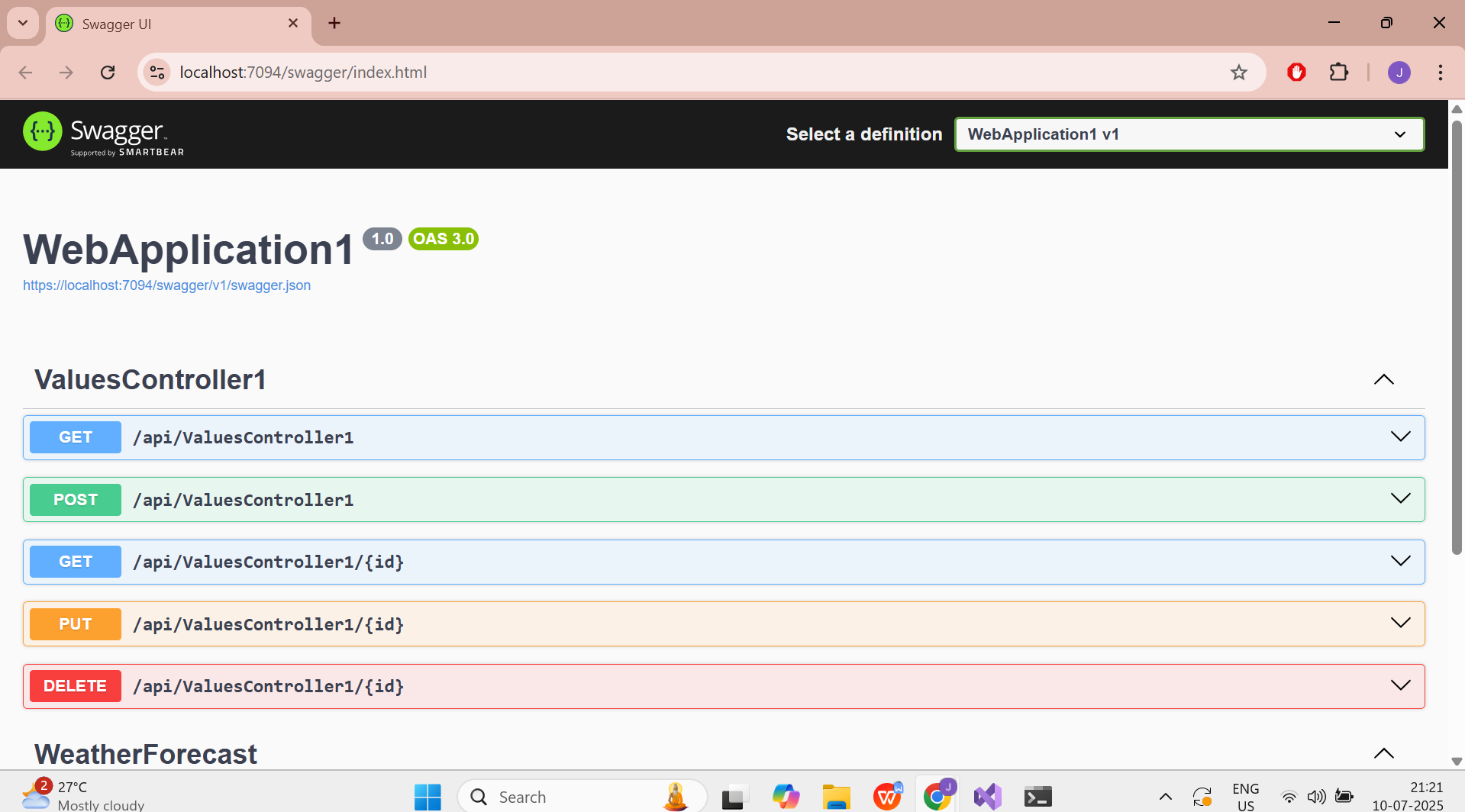
By-Jayati Pal

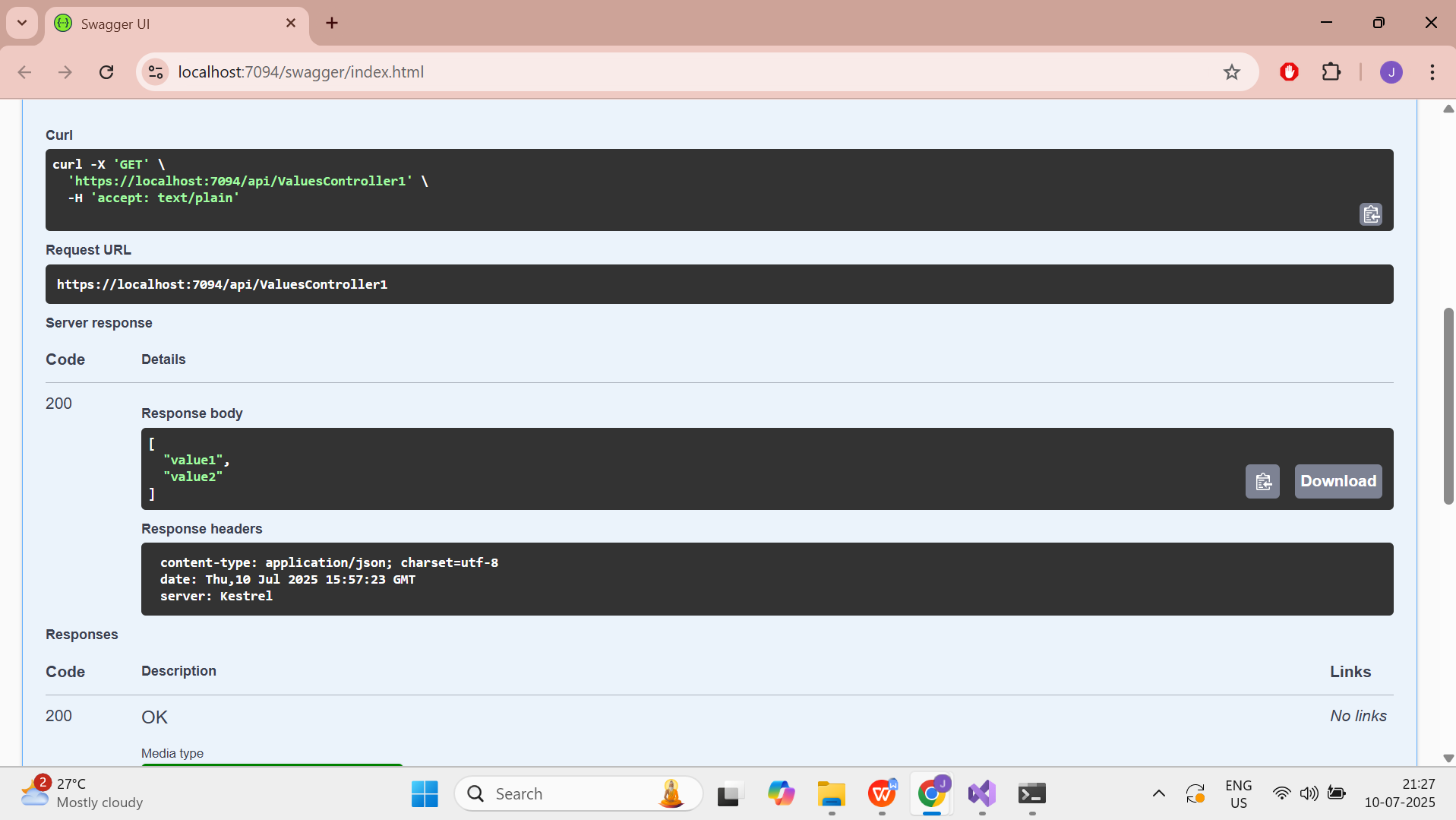
**Module 6 - ASP.NET Core 8.0 Web API**

Lab 1:First Web Api using .Net core

Code and Output:







Lab 2:Web API using .Net core with Swagger

Code:

using Microsoft.OpenApi.Models;

namespace WebApplication2

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

TermsOfService = new Uri("https://www.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();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

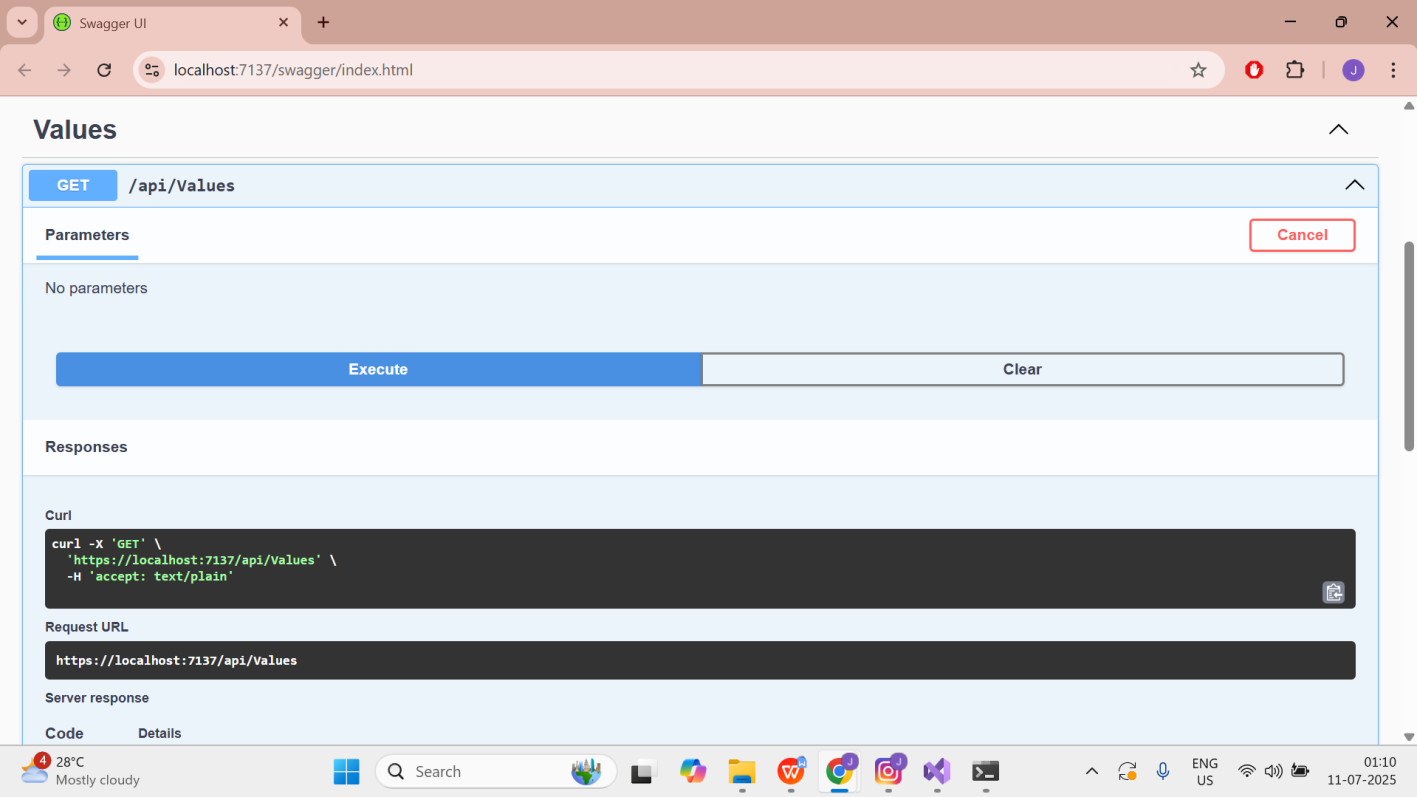
app.Run();

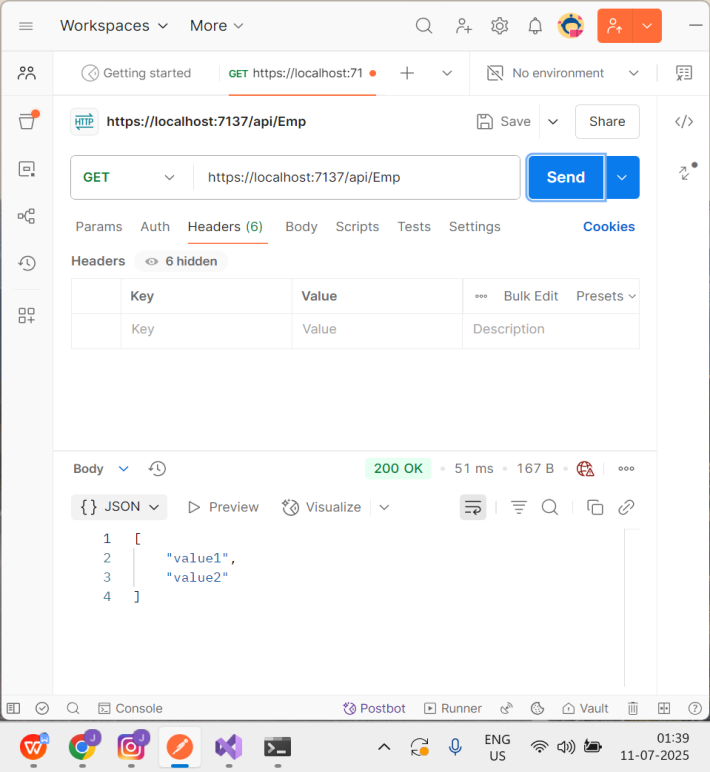
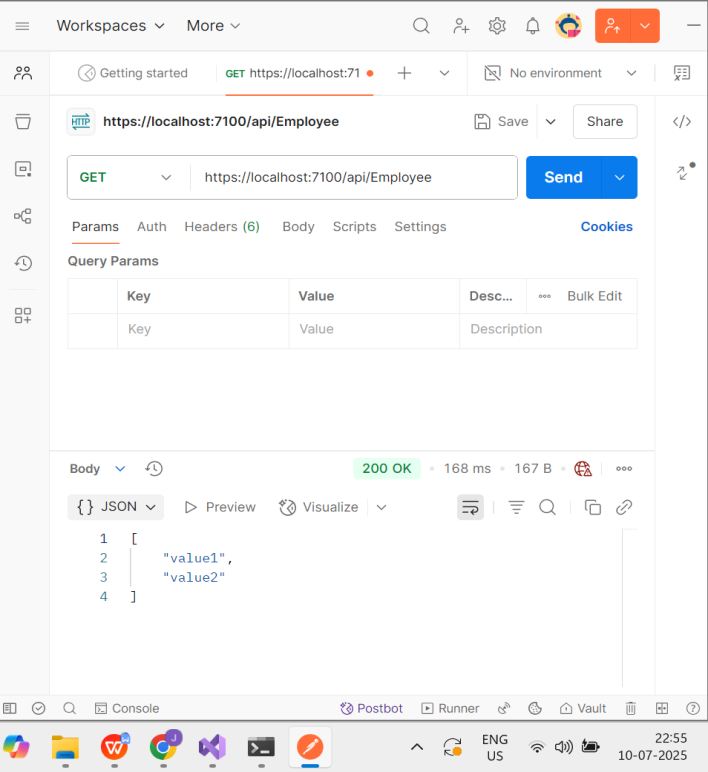
}

}

}

Output:





Lab 3:

1. Web Api using custom model class

Code:

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System.Linq;

using WebApplication3.Models;

[AllowAnonymous]

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees;

static EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

// GET: /Employee

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), 200)]

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

// GET: /Employee/standrad

[HttpGet("standrad")]

public ActionResult<Employee> GetStandrad()

{

return Ok(\_employees.FirstOrDefault());

}

// POST: /Employee

[HttpPost]

public ActionResult<Employee> Post(Employee emp)

{

\_employees.Add(emp);

return CreatedAtAction(nameof(Get), new { id = emp.Id }, emp);

}

// PUT: /Employee/{id}

[HttpPut("{id}")]

public IActionResult Put(int id, Employee emp)

{

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

if (existing == null) return NotFound();

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 NoContent();

}

private static List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John Doe",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skills> { new Skills { Id = 1, Name = "C#" } },

DateOfBirth = new DateTime(1990, 1, 1)

},

new Employee

{

Id = 2,

Name = "Jane Smith",

Salary = 60000,

Permanent = false,

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

Skills = new List<Skills>

{

new Skills { Id = 3, Name = "C#" },

new Skills { Id = 4, Name = "SQL" }

},

DateOfBirth = new DateTime(1985, 8, 15)

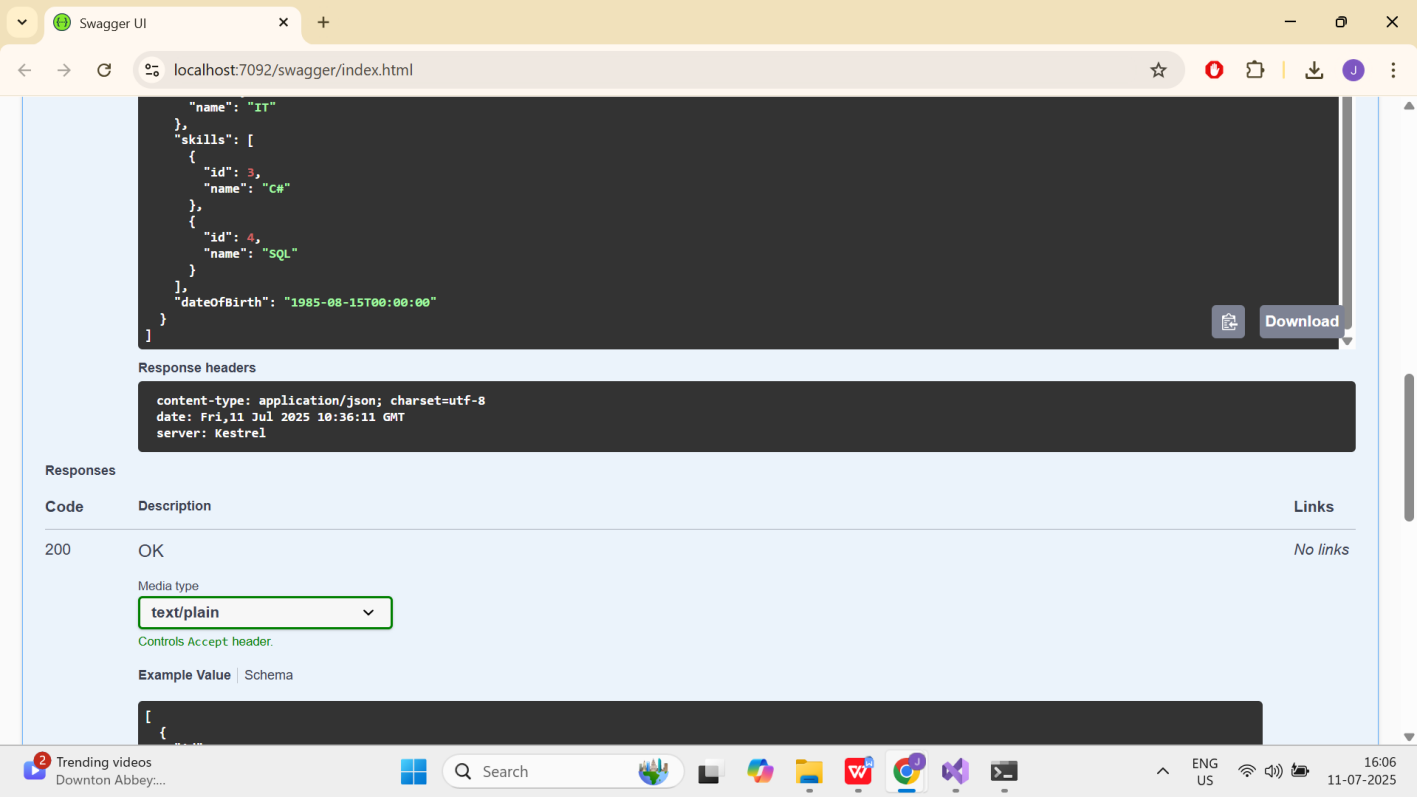
}

};

}

}

Output:



1. Create a Custom action filter for Authorization.

Code:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.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");

return;

}

base.OnActionExecuting(context);

}

}

1. Custom Exception filter

Code:

using Microsoft.AspNetCore.Mvc.Filters;

using Microsoft.AspNetCore.Mvc;

using System.IO;

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

var message = $"{DateTime.Now}: {exception.Message}{Environment.NewLine}";

File.AppendAllText("exception\_log.txt", message);

context.Result = new ObjectResult("Internal server error")

{

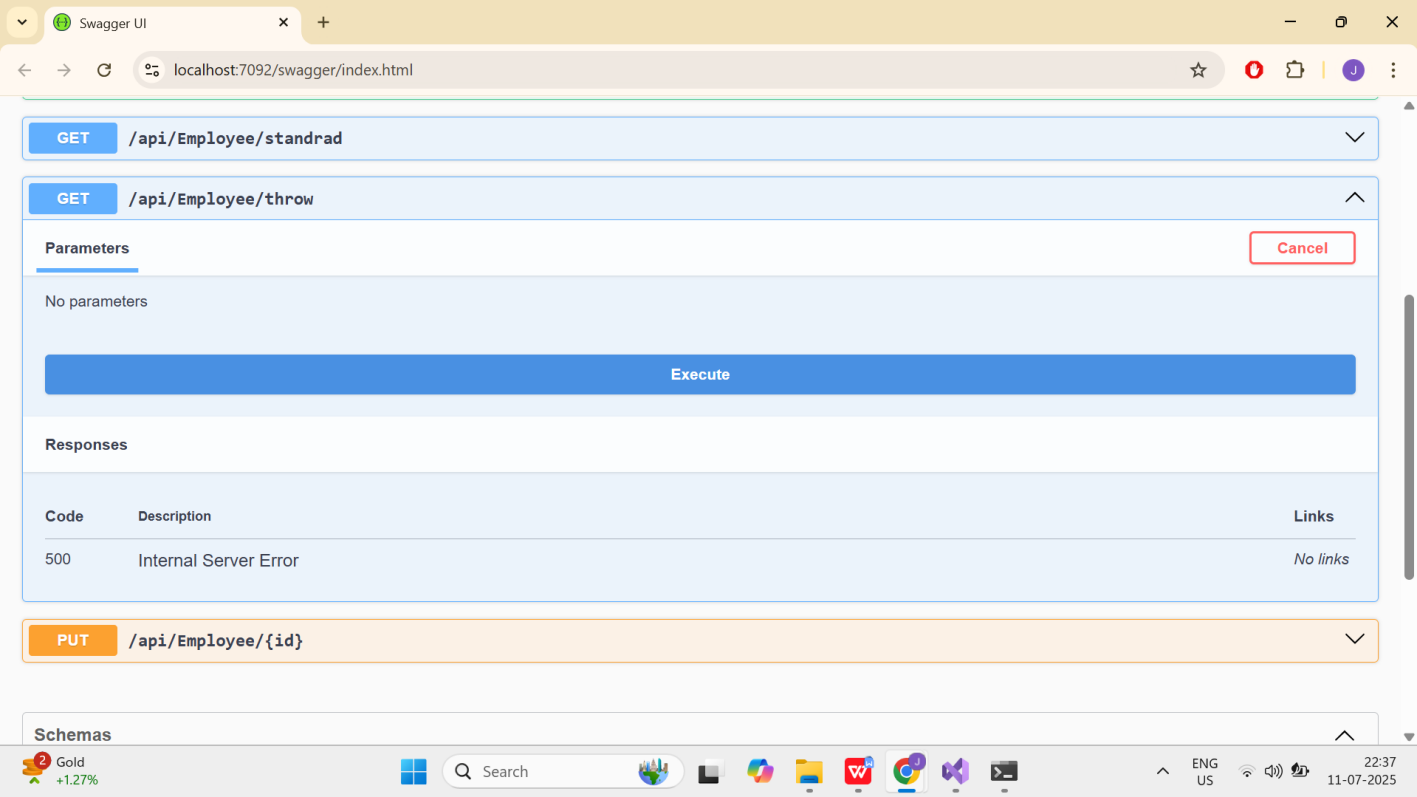
StatusCode = 500

};

context.ExceptionHandled = true;

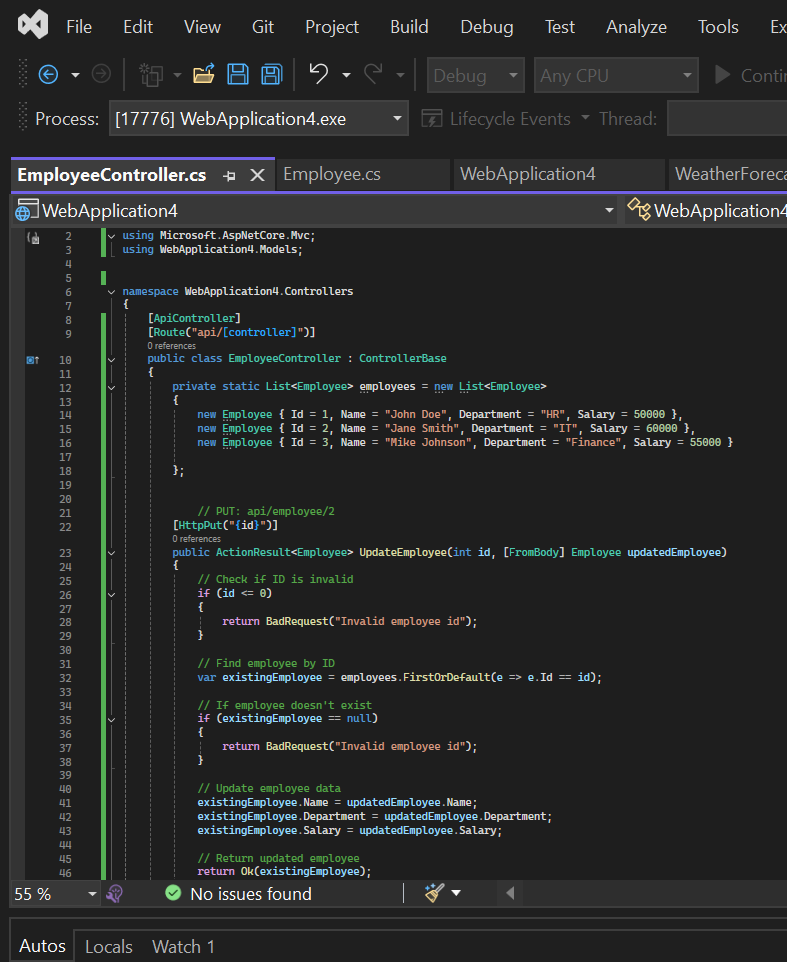
}

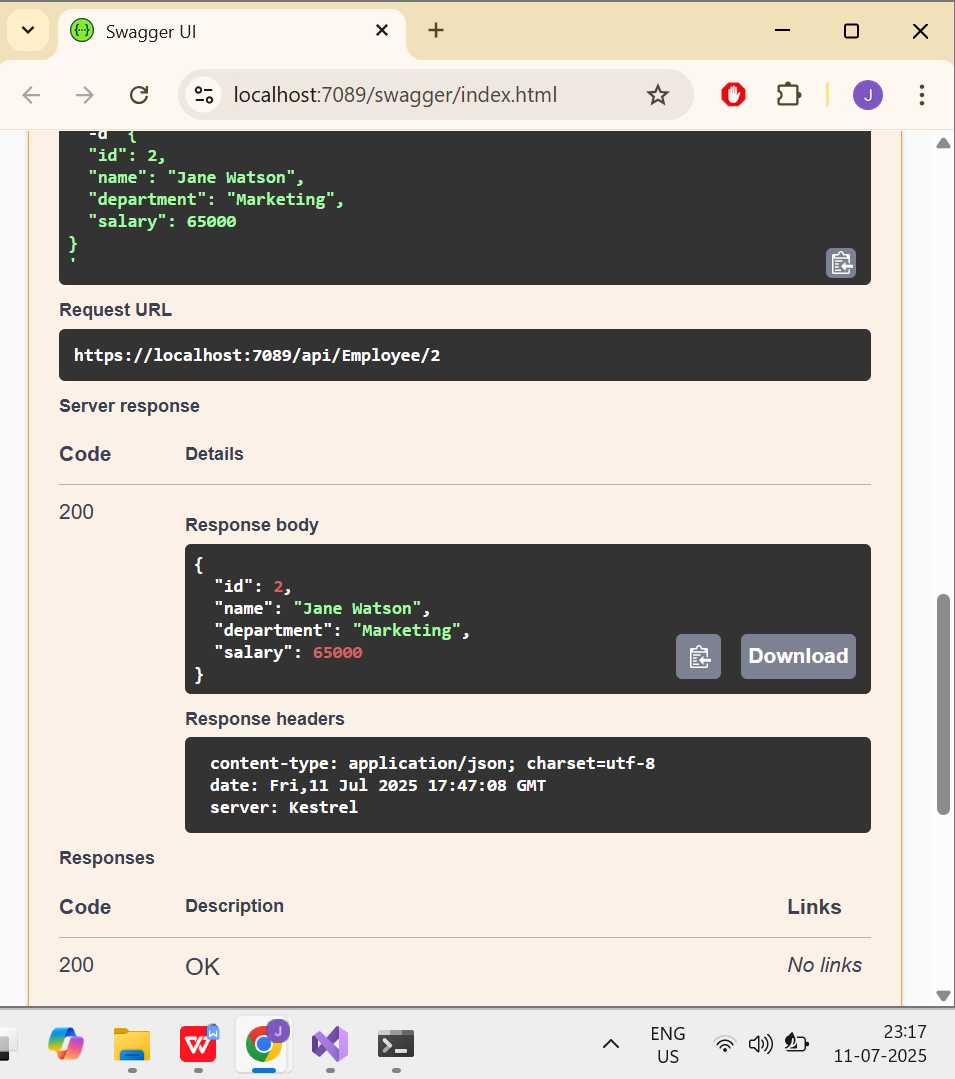
}

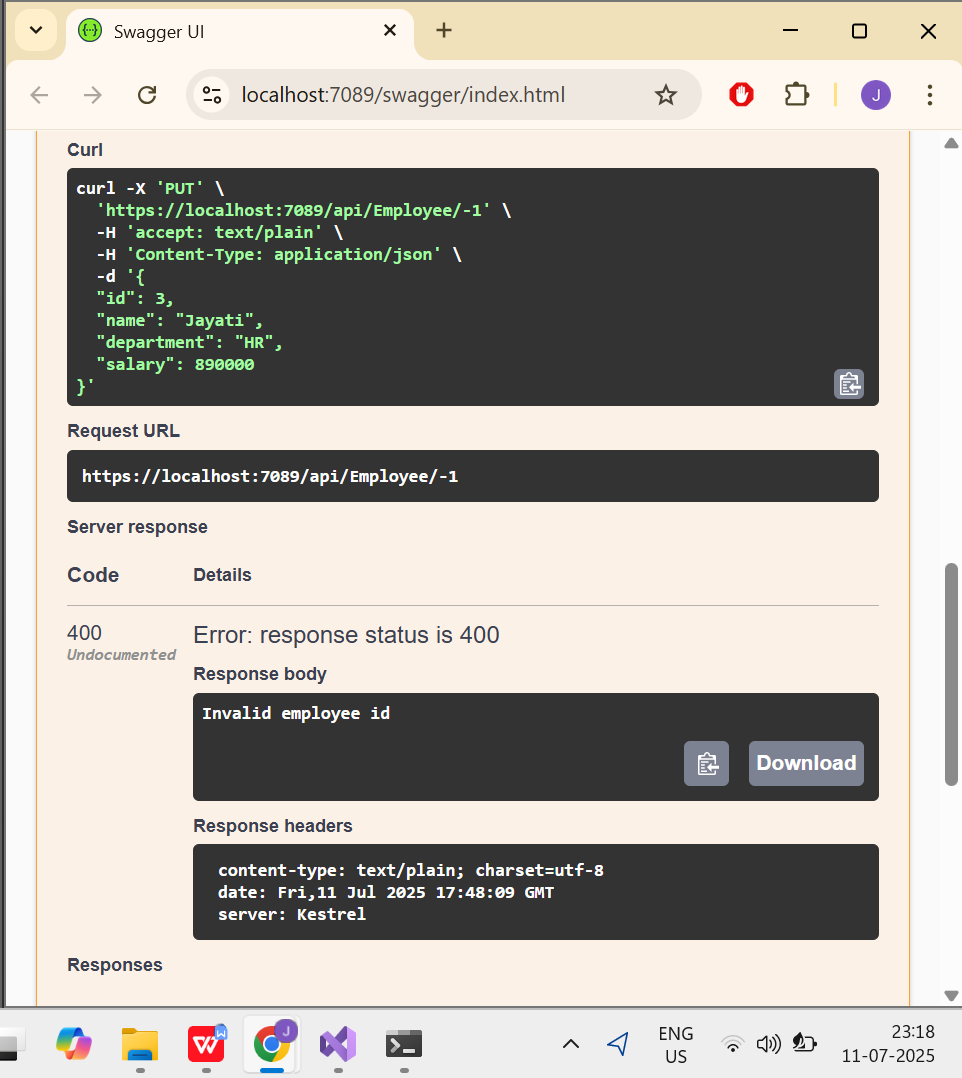


Lab 4:Web Api CRUD operation

Code and Output:

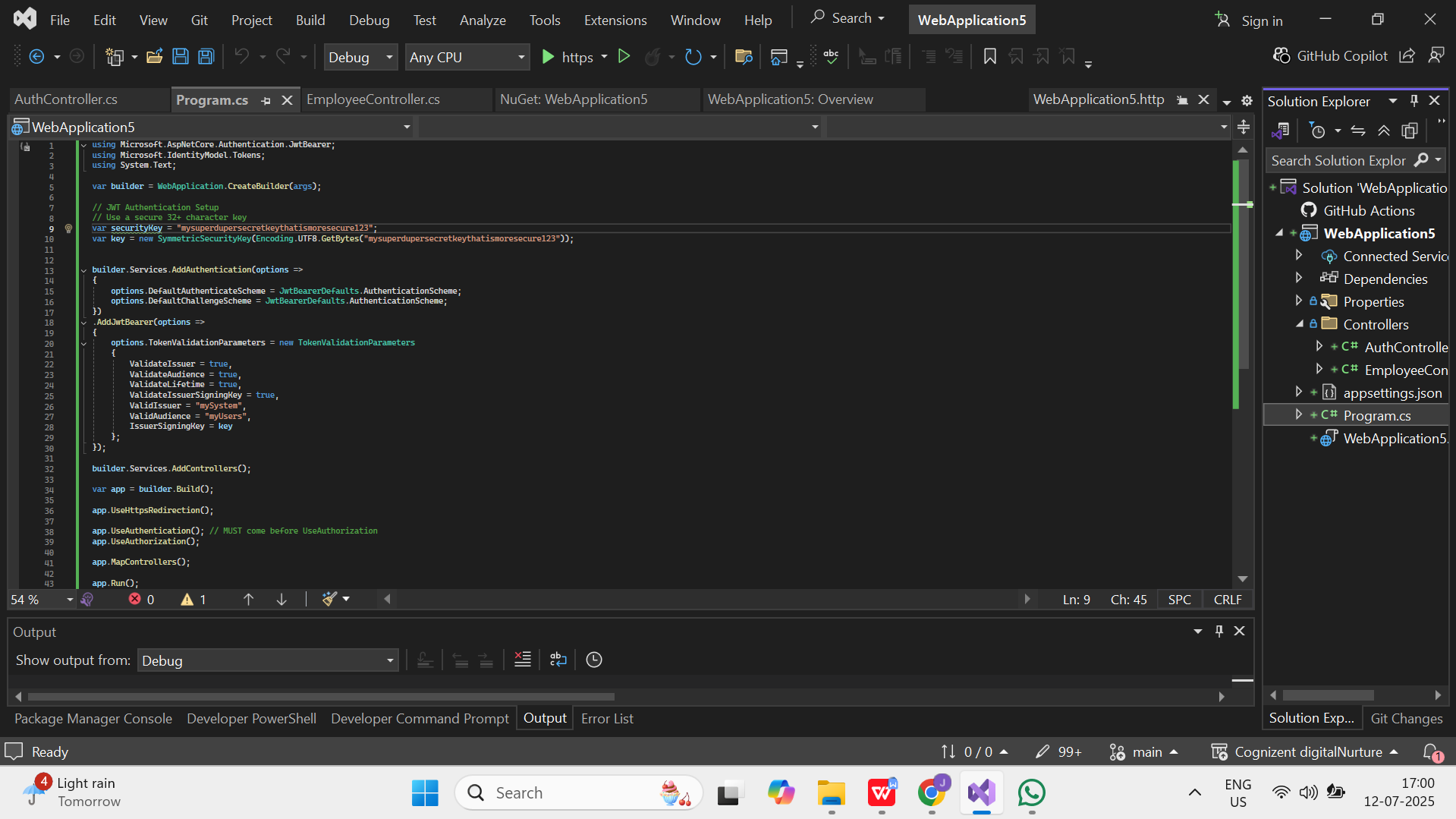


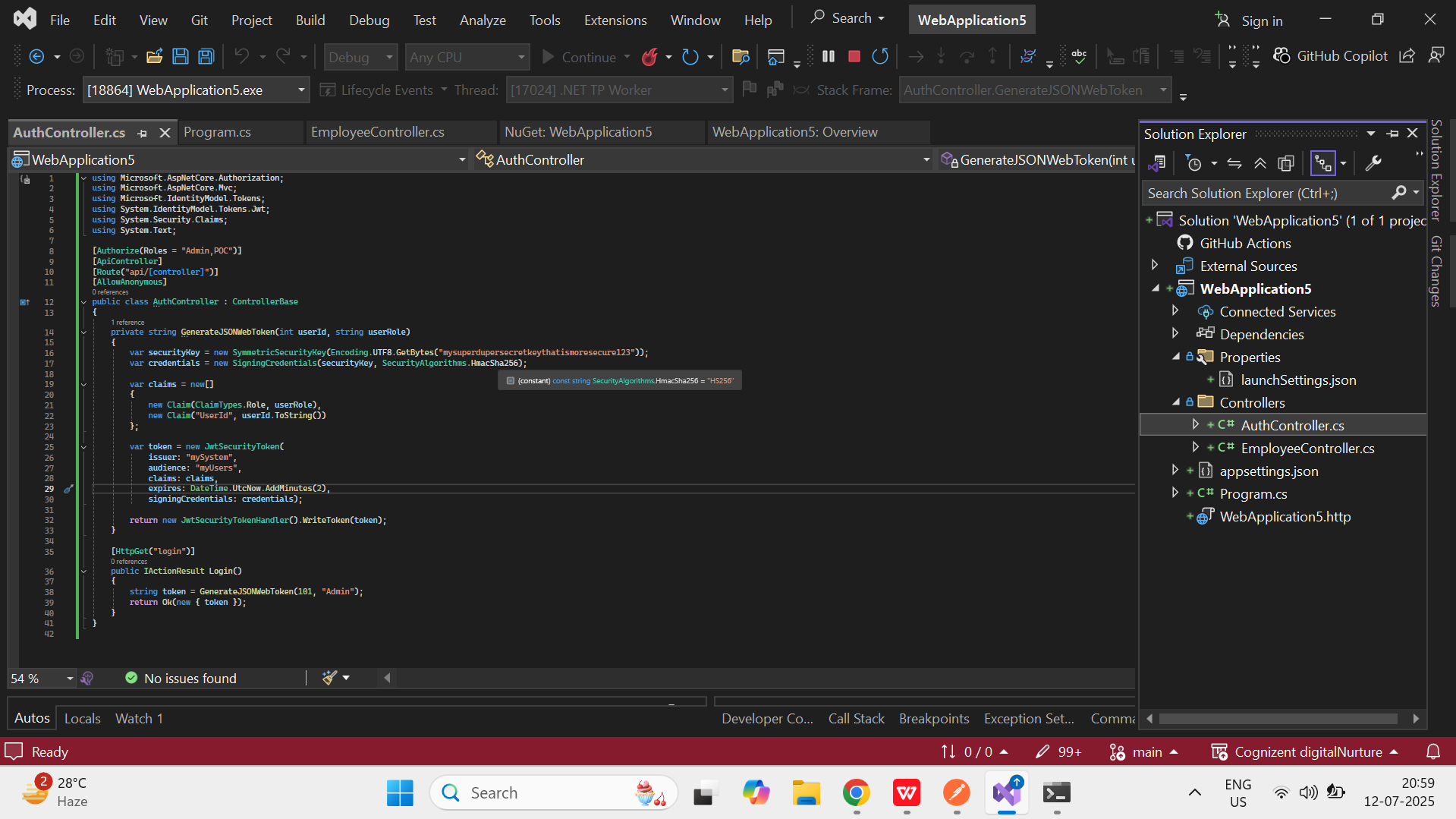




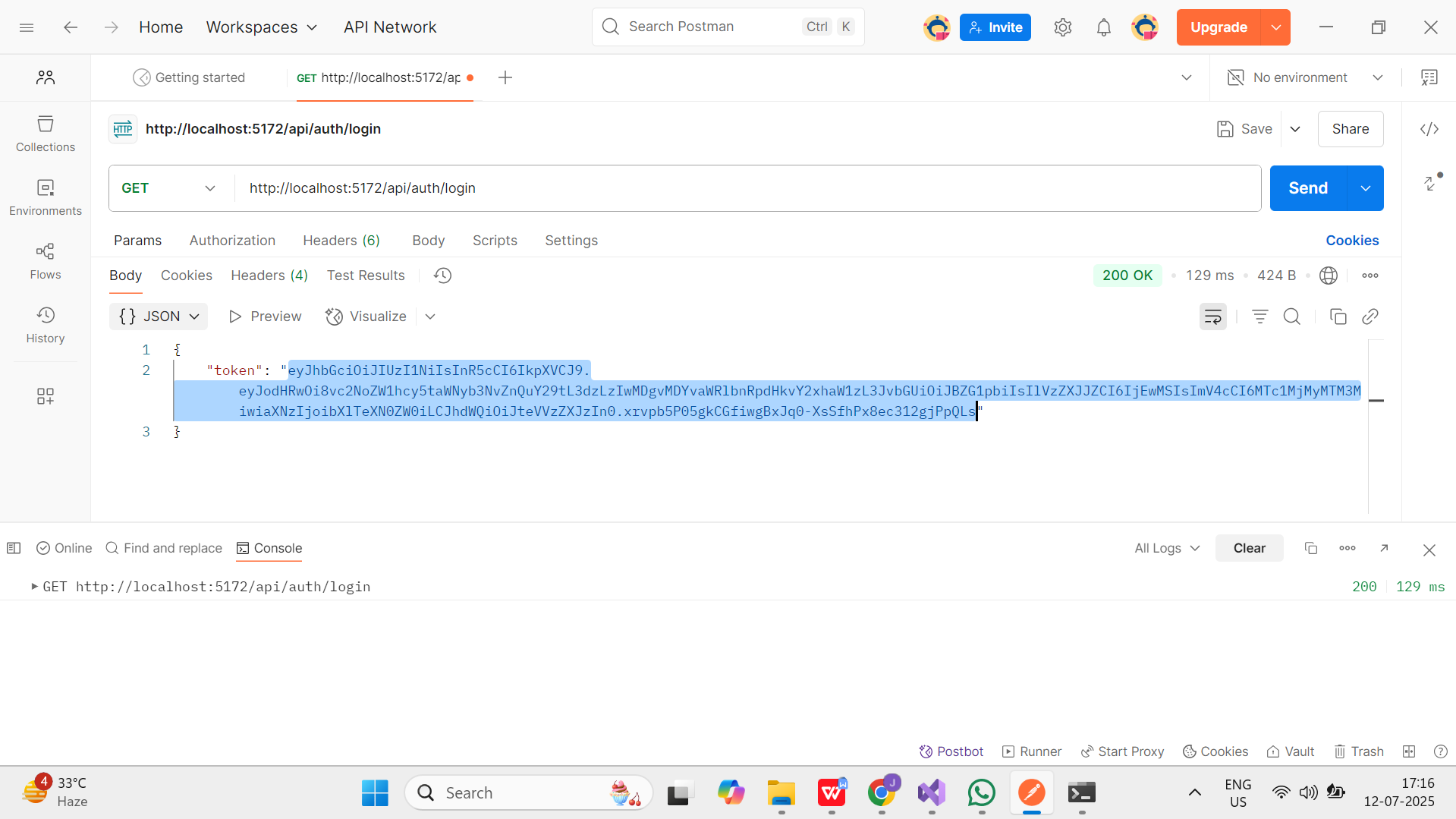
Lab 5:

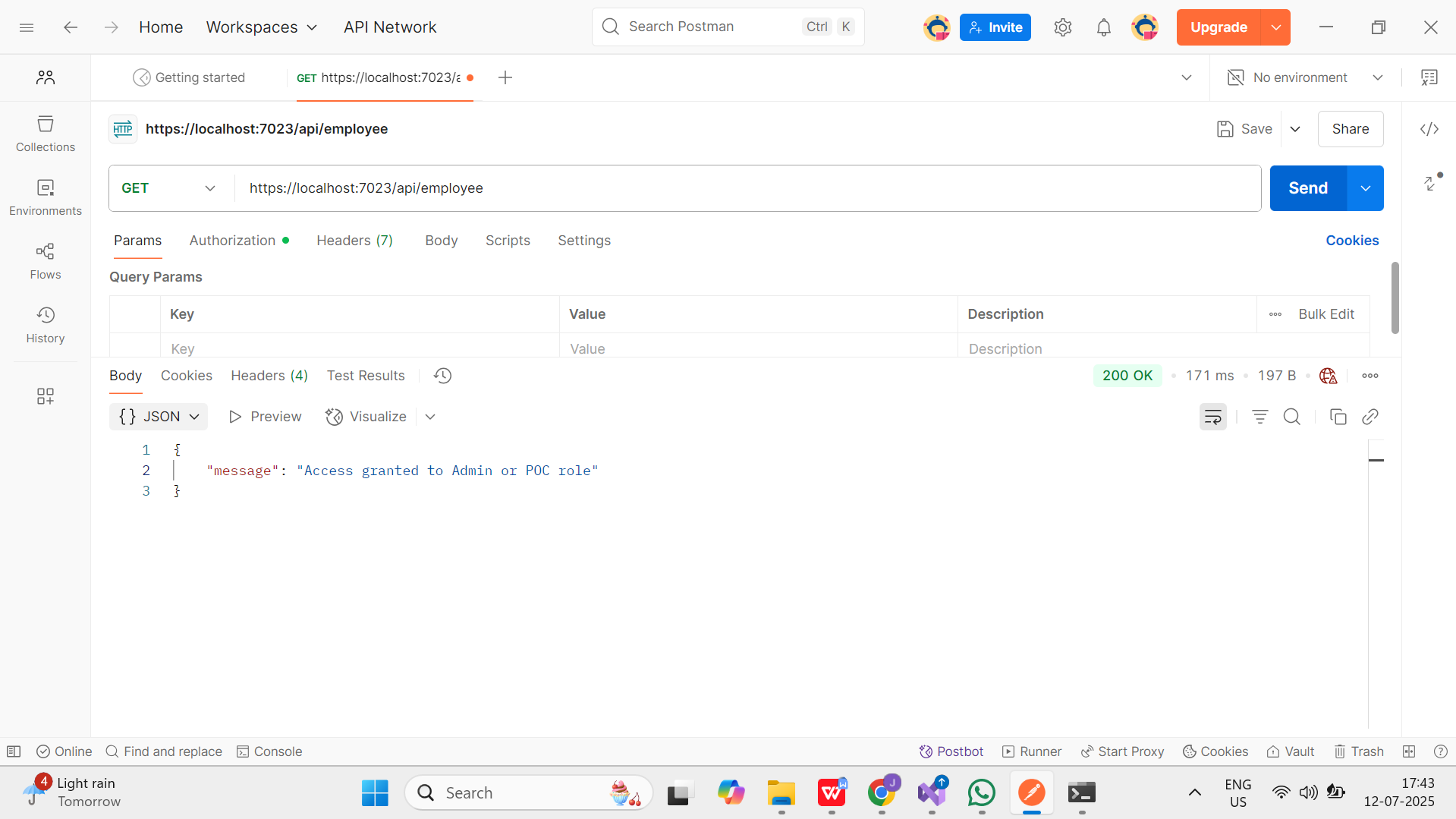
Code:

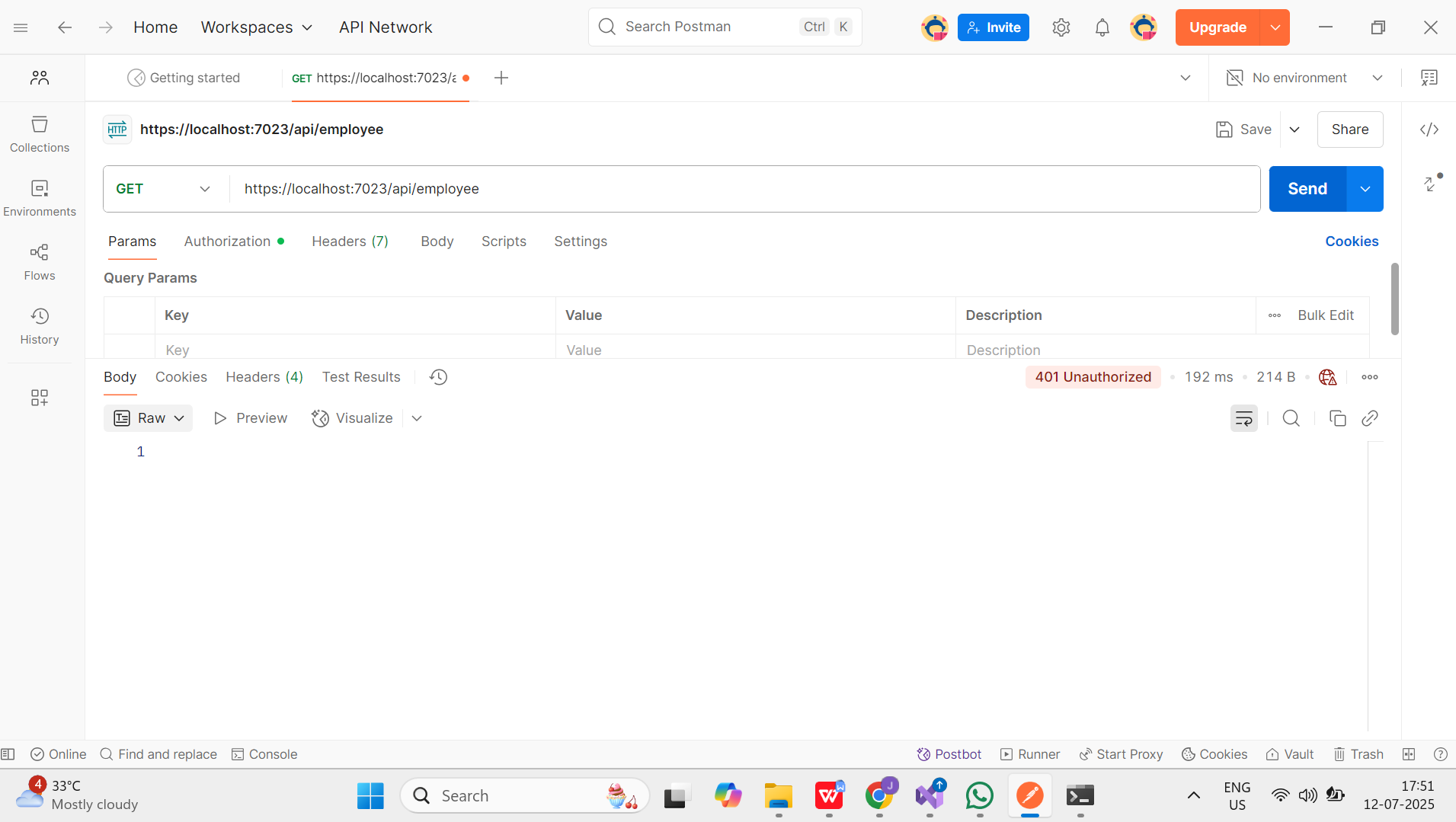




Output:

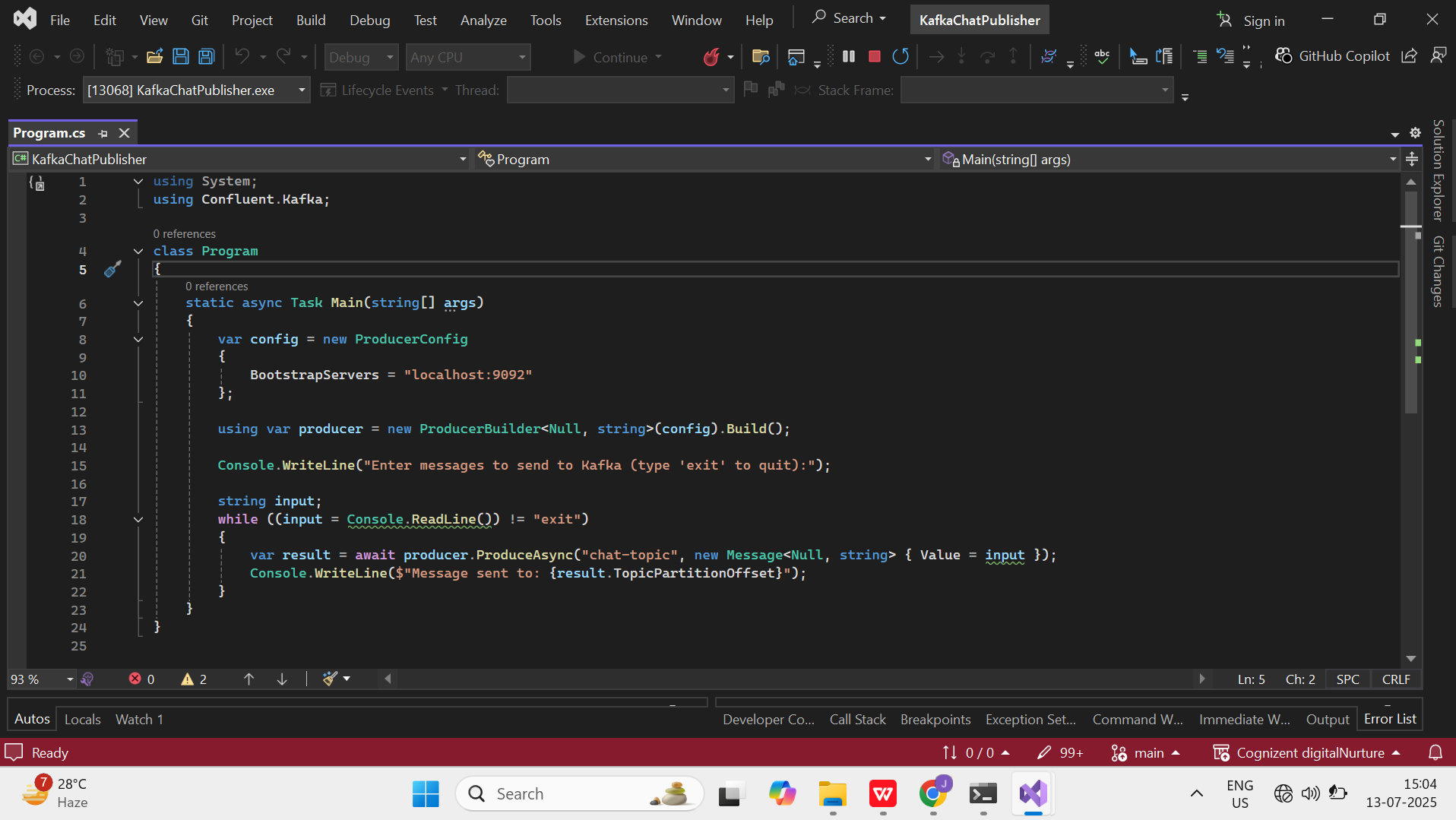


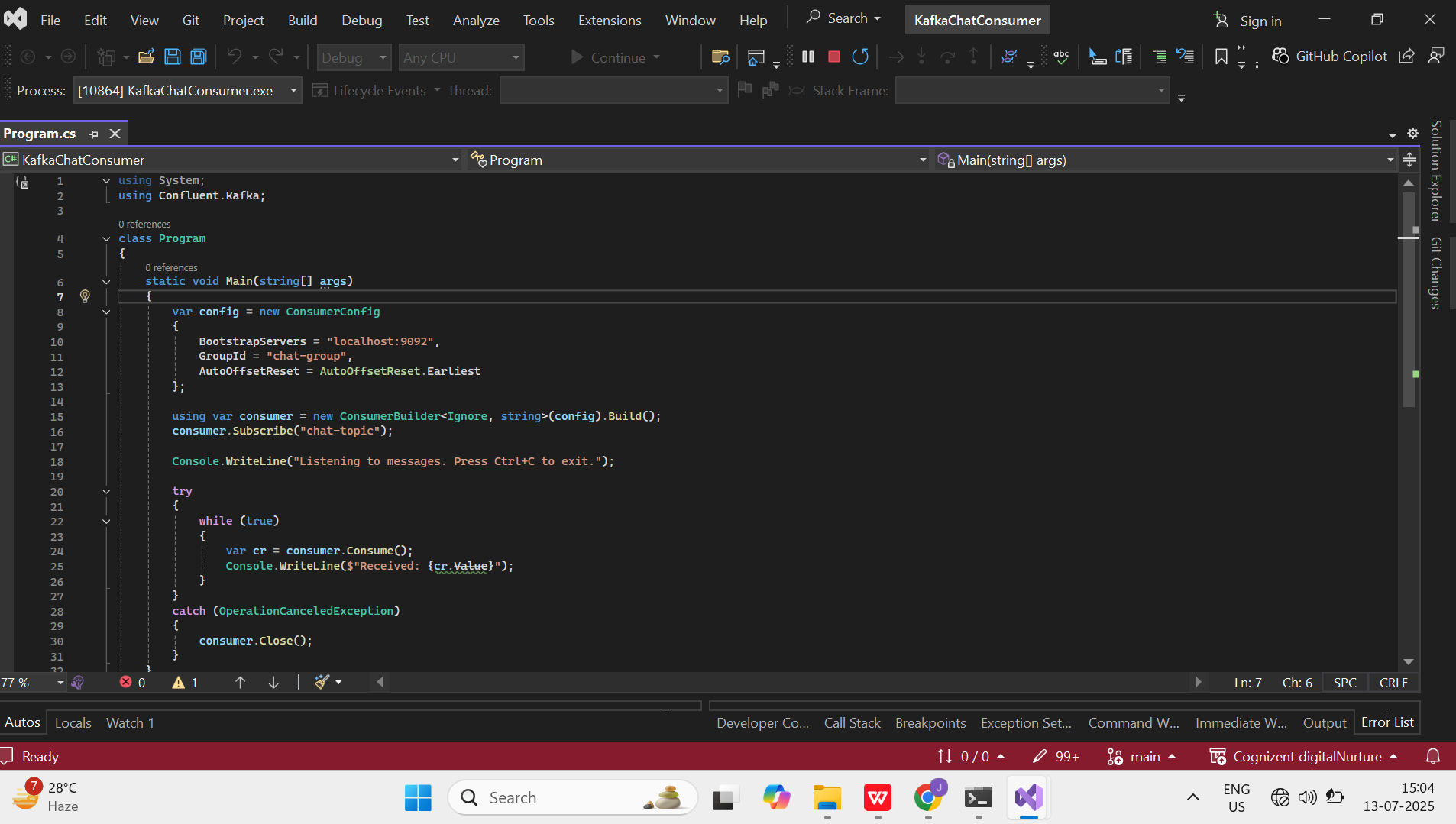




Lab 6:

Code:





Output:

