**NAME-MASHEERA AFRIN**

**SUPERSET ID: 6362071**

**WEBAPI HANDSON-1**

**ValueController.cs**

using Microsoft.AspNetCore.Mvc;

namespace FirstWebApi.Controllers

{

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

[ApiController]

public class ValuesController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

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

}

[HttpGet("{id}")]

public IActionResult Get(int id)

{

return Ok("value " + id);

}

[HttpPost]

public IActionResult Post([FromBody] string value)

{

return Ok("Created: " + value);

}

[HttpPut("{id}")]

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

{

return Ok($"Updated id {id} with value {value}");

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

return Ok($"Deleted id {id}");

}

}

}

**PROGRAM.CS**

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

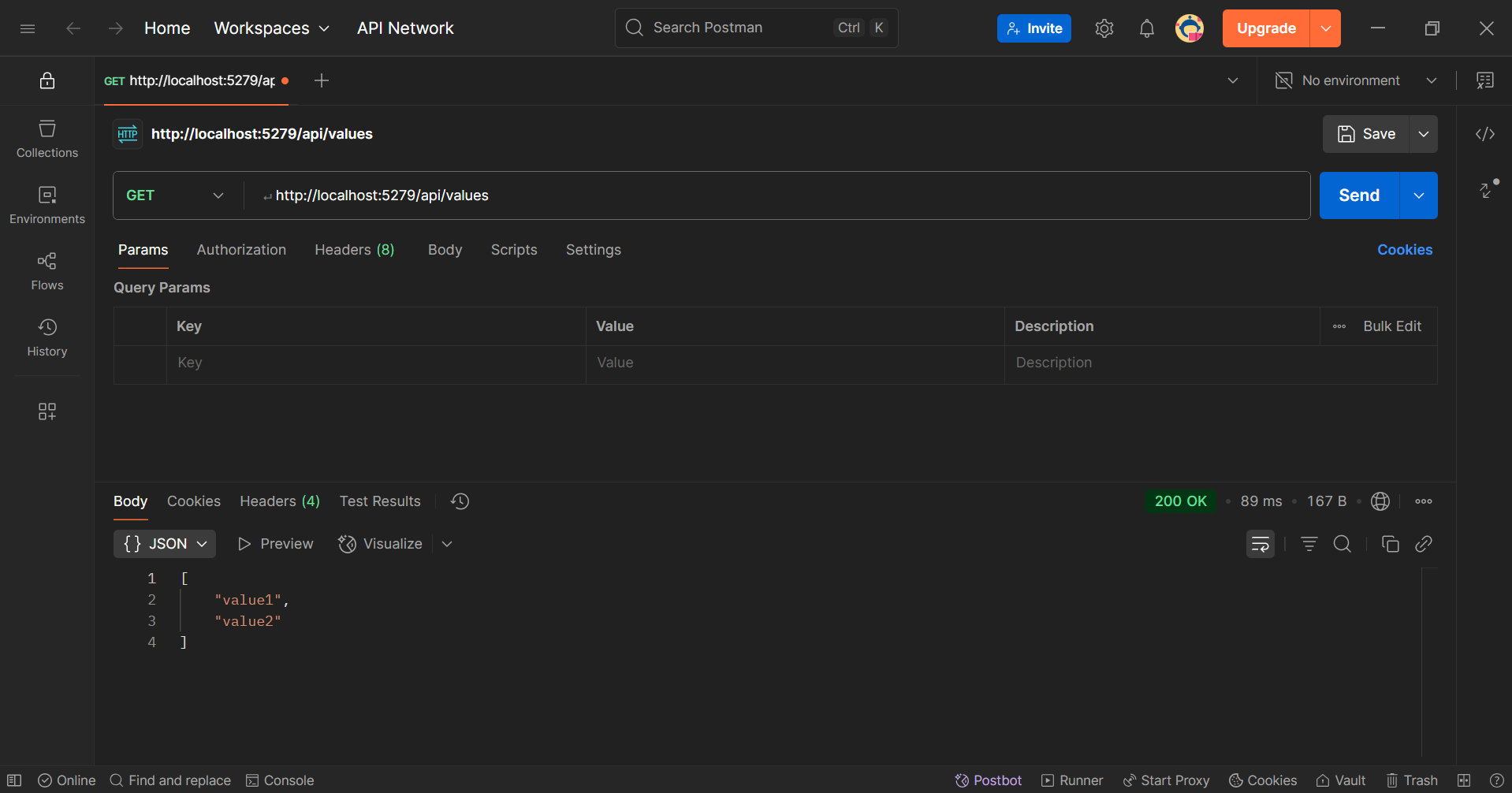
var app = builder.Build();

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();



**WEBAPI HANDSON-2**

**VALUECONTROLLER.CS**

using Microsoft.AspNetCore.Mvc;

namespace SwaggerDemo.Controllers

{

[ApiController]

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

public class ValuesController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

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

}

[HttpGet("{id}")]

public IActionResult Get(int id)

{

return Ok("value " + id);

}

}

}

**PROGRAM.CS**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

},

License = new OpenApiLicense

{

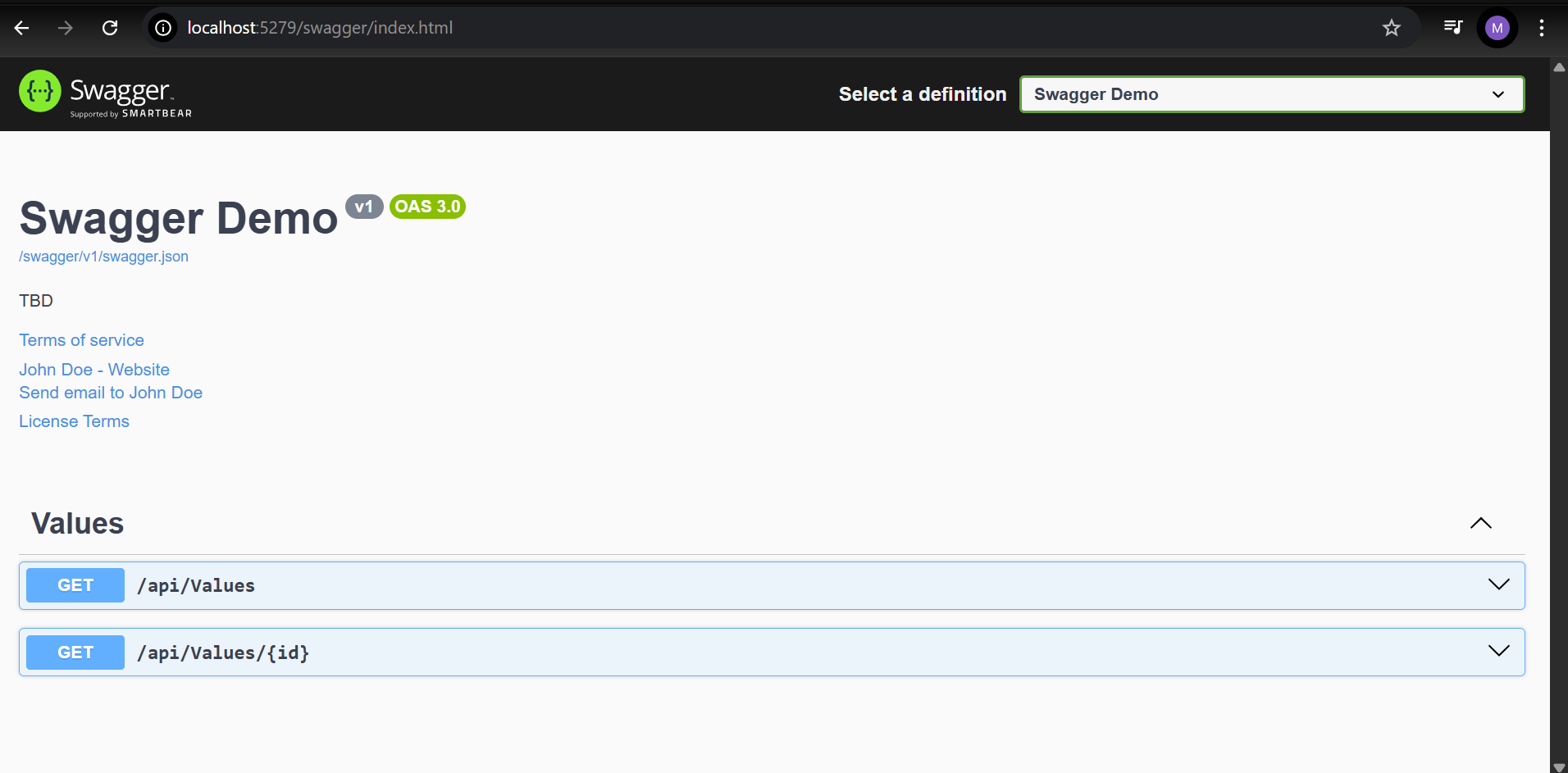
Name = "License Terms",

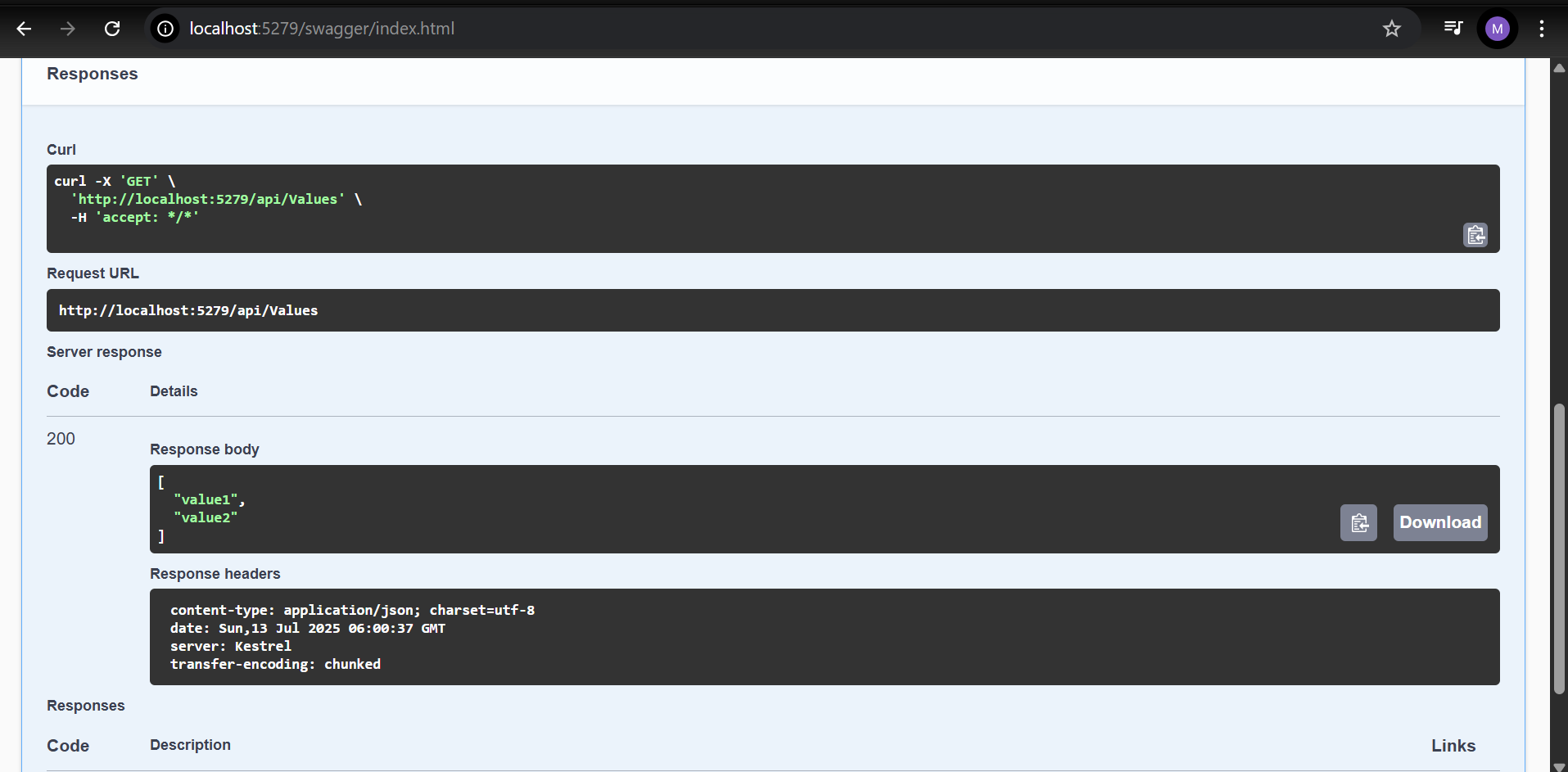
Url = new Uri("https://www.example.com")

}

});

});





**STEP 2.**

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

namespace FirstWebApi.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

var employees = new List<object>

{

new { Id = 1, Name = "Alice", Role = "Manager" },

new { Id = 2, Name = "Bob", Role = "Developer" },

new { Id = 3, Name = "Charlie", Role = "Designer" }

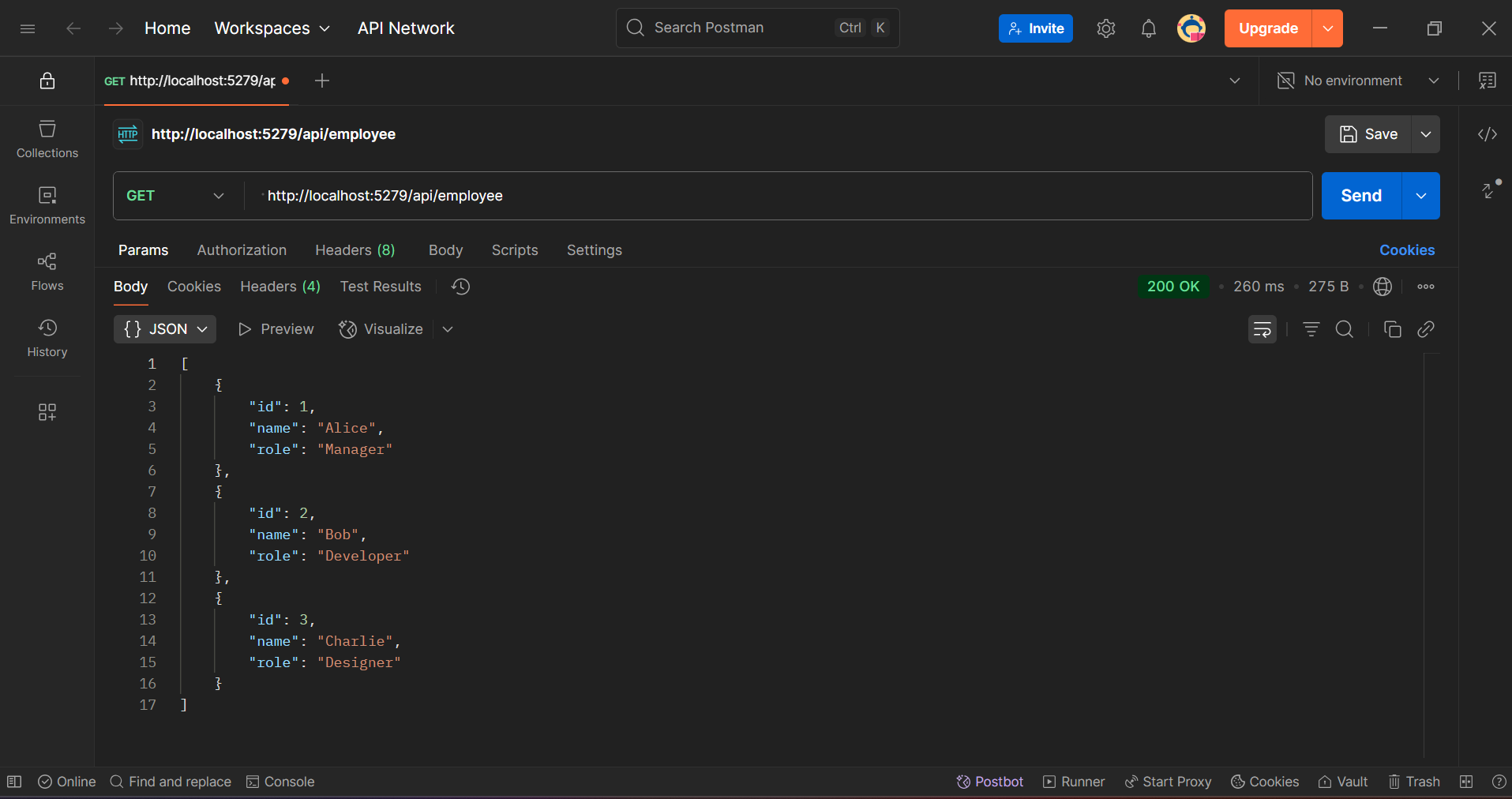
};

return Ok(employees);

}

}

}



**STEP3**

**EMPLOYEECONTROLLER.CS**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace FirstWebApi.Controllers

{

[ApiController]

[Route("api/Emp")]

public class EmployeeController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

var employees = new List<object>

{

new { Id = 1, Name = "Alice", Role = "Manager" },

new { Id = 2, Name = "Bob", Role = "Developer" },

new { Id = 3, Name = "Charlie", Role = "Designer" }

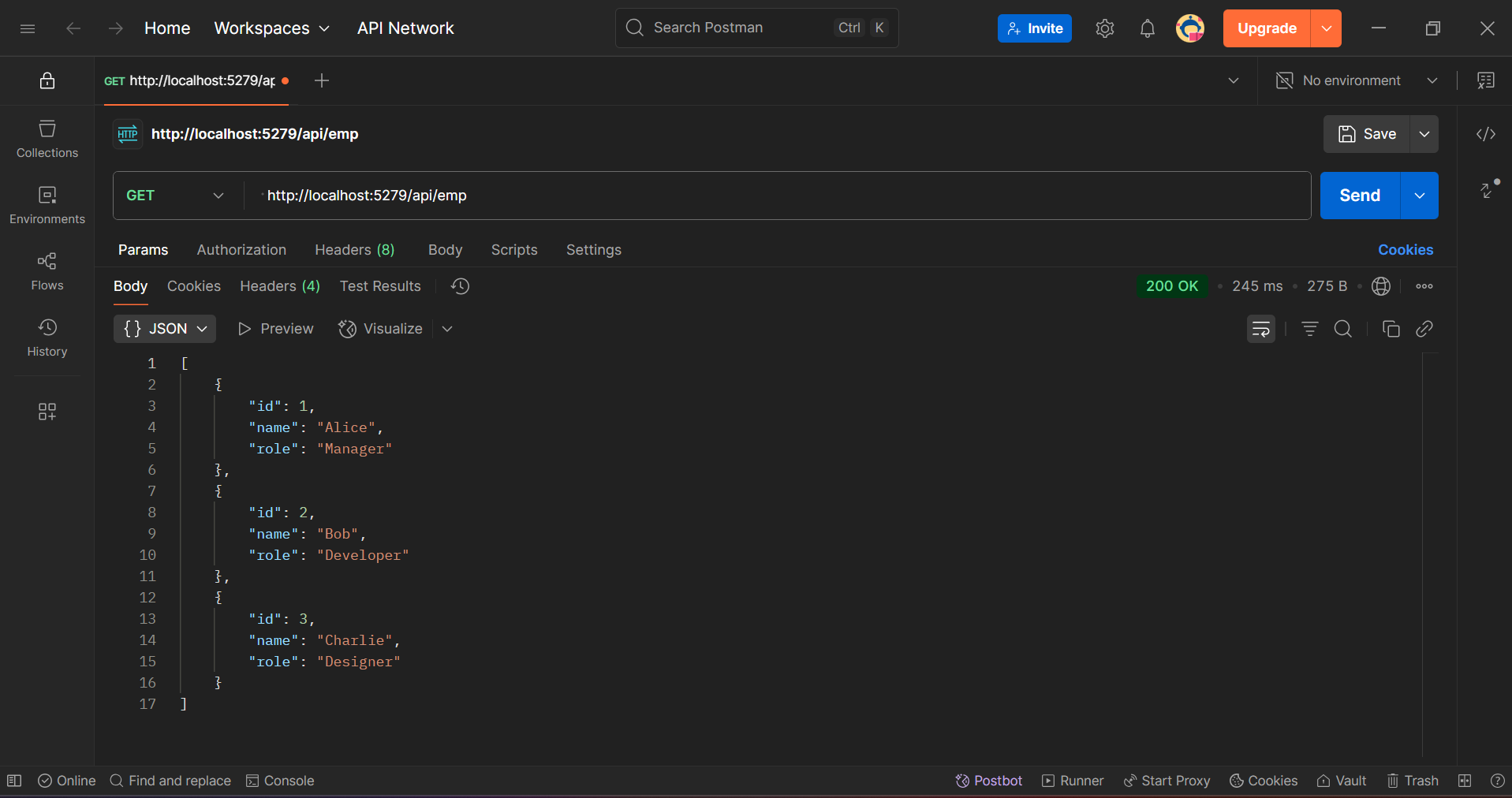
};

return Ok(employees);

}

}

}



**WEBAPI HANDSON-3**

**Employee.cs**

using System;

using System.Collections.Generic;

namespace FirstWebApi.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 FirstWebApi.Models

{

public class Skill

{

public int Id { get; set; }

public string? Name { get; set; }

}

}

**Department.cs**

using FirstWebApi.Models;

namespace FirstWebApi.Models

{

public class Department

{

public int Id { get; set; }

public string? Name { get; set; }

}

}

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using FirstWebApi.Models;

namespace FirstWebApi.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = new List<Employee>();

static EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

[HttpGet("standard")]

public ActionResult<Employee> GetStandard()

{

return Ok(GetStandardEmployeeList()[0]);

}

[HttpPost]

public IActionResult Post([FromBody] Employee newEmployee)

{

\_employees.Add(newEmployee);

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

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] Employee updatedEmployee)

{

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

if (emp == null)

return NotFound();

emp.Name = updatedEmployee.Name;

emp.Salary = updatedEmployee.Salary;

emp.Permanent = updatedEmployee.Permanent;

emp.Department = updatedEmployee.Department;

emp.Skills = updatedEmployee.Skills;

emp.DateOfBirth = updatedEmployee.DateOfBirth;

return Ok(emp);

}

private static List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John Smith",

Salary = 60000,

Permanent = true,

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

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

},

DateOfBirth = new DateTime(1990, 5, 23)

},

new Employee

{

Id = 2,

Name = "Alice Brown",

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 = "Budgeting" }

},

DateOfBirth = new DateTime(1992, 11, 12)

}

};

}

}

}

**Program.cs**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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

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();

app.UseHttpsRedirection();

app.UseAuthorization();

app.UseSwagger();

app.UseSwaggerUI(c =>

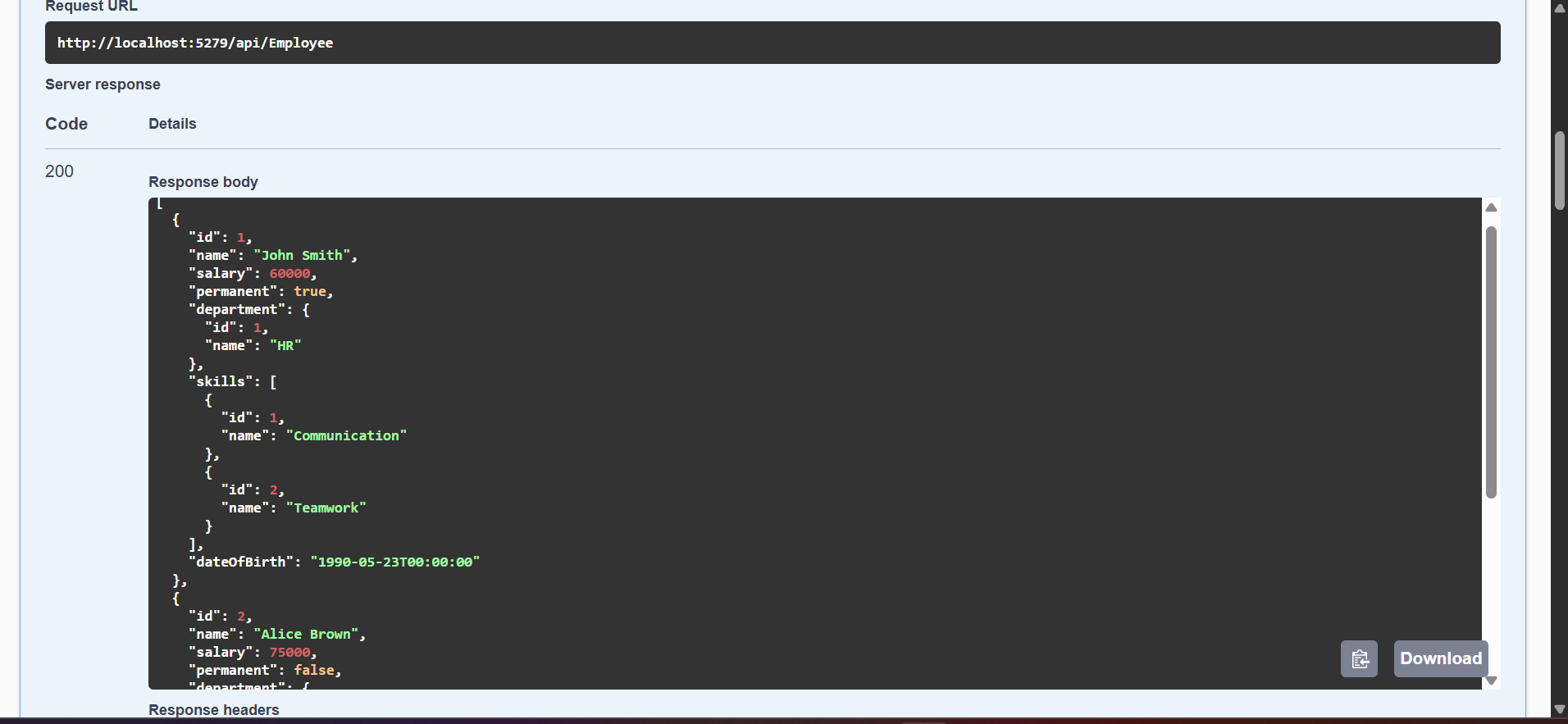
{

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

});

app.MapControllers();

app.Run();



**STEP2:**

**CustomerAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace FirstWebApi.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var hasAuthHeader = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var authHeader);

if (!hasAuthHeader)

{

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

return;

}

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

{

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

return;

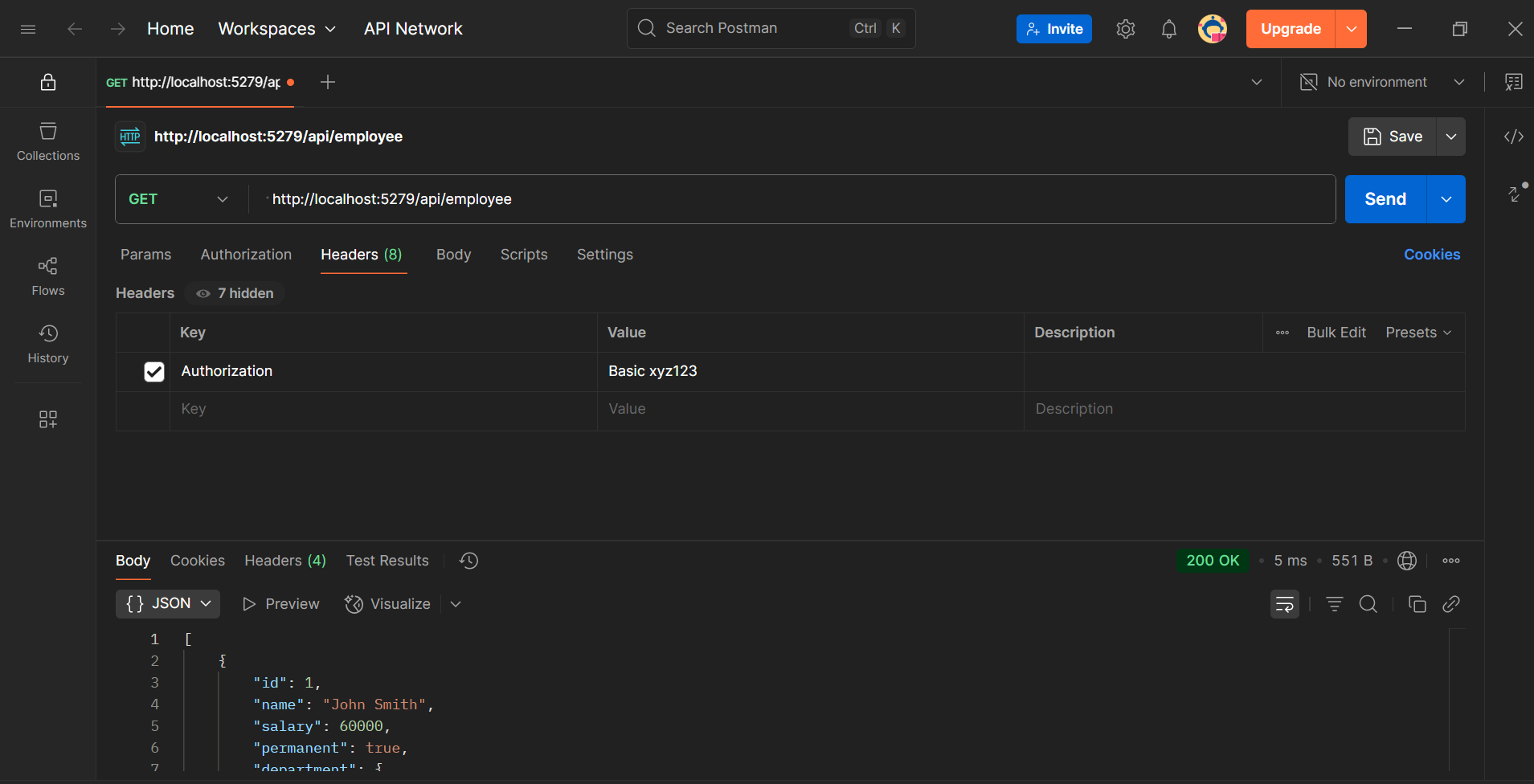
}

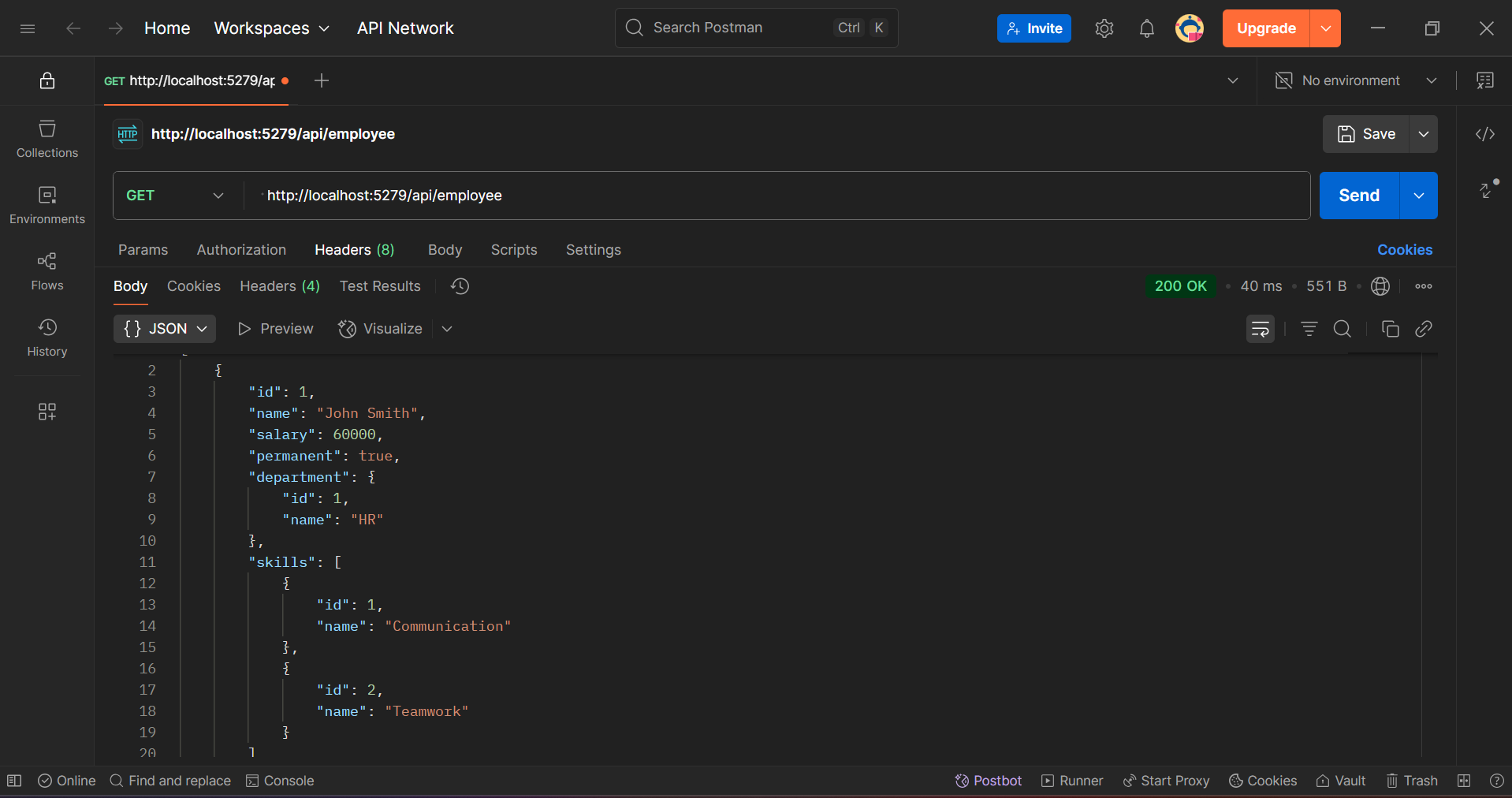
base.OnActionExecuting(context);

}

}

}





**STEP-3**

**CustomerAuthExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

namespace FirstWebApi.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

var logPath = "exception\_log.txt";

File.AppendAllText(logPath,

$"[{DateTime.Now}] Exception: {exception.Message}\nStackTrace: {exception.StackTrace}\n\n");

// Return custom 500 response using ObjectResult

context.Result = new ObjectResult(new

{

Message = "An error occurred while processing your request.",

Exception = exception.Message

})

{

StatusCode = 500

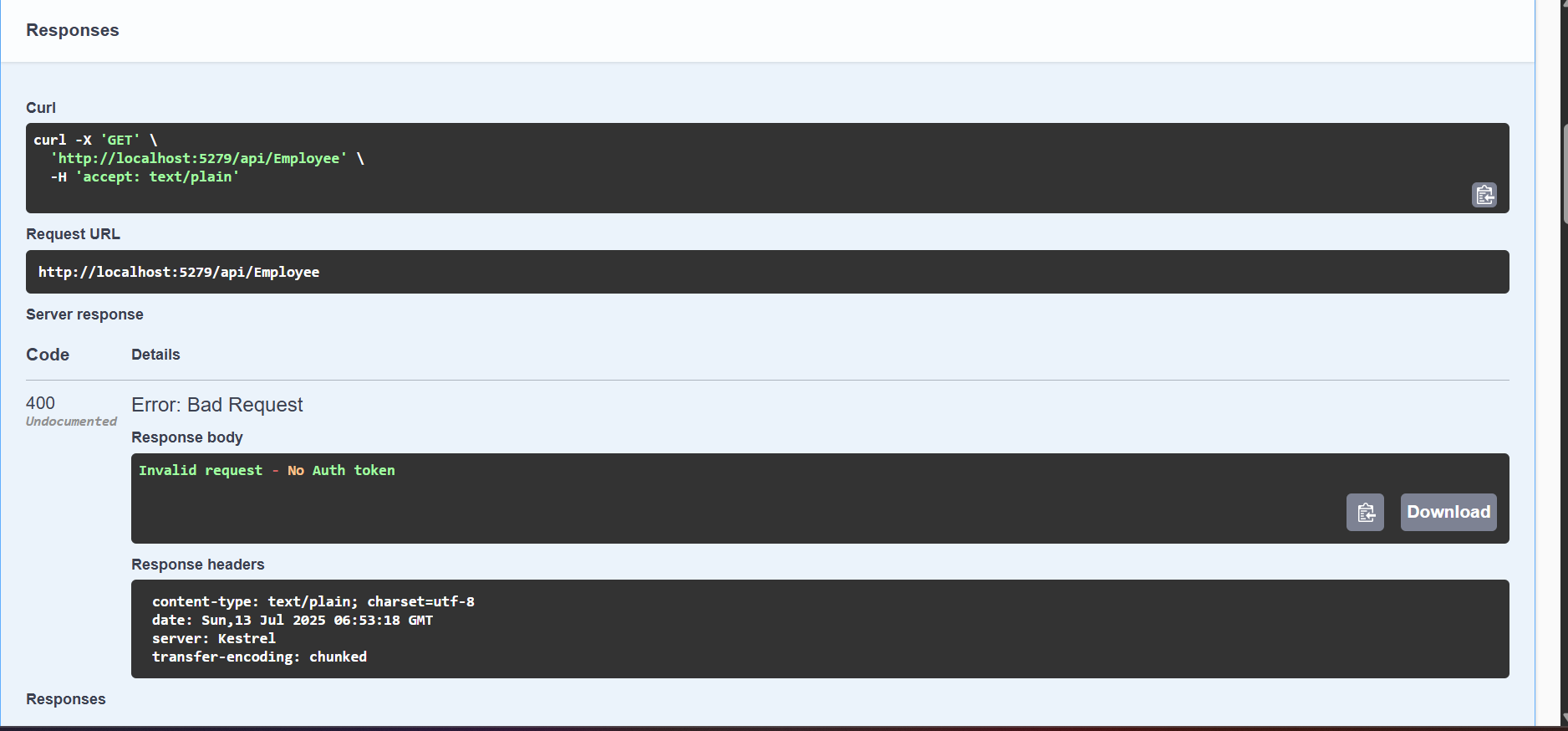
};

context.ExceptionHandled = true;

}

}

}



**WEBAPI HANDSON-4**

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using FirstWebApi.Models;

using FirstWebApi.Filters;

namespace FirstWebApi.Controllers

{

[ApiController]

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

[CustomAuthFilter]

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = new List<Employee>();

static EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

[HttpGet("test-exception")]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public IActionResult ThrowError()

{

throw new Exception("Intentional exception for testing");

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

[HttpGet("standard")]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<Employee> GetStandard()

{

return Ok(GetStandardEmployeeList()[0]);

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

public IActionResult Post([FromBody] Employee newEmployee)

{

\_employees.Add(newEmployee);

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

}

[HttpPut("{id}")]

[ProducesResponseType(typeof(Employee), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

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

{

if (id <= 0)

return BadRequest("Invalid employee id");

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

if (emp == null)

return BadRequest("Invalid employee id");

emp.Name = updatedEmployee.Name;

emp.Salary = updatedEmployee.Salary;

emp.Permanent = updatedEmployee.Permanent;

emp.Department = updatedEmployee.Department;

emp.Skills = updatedEmployee.Skills;

emp.DateOfBirth = updatedEmployee.DateOfBirth;

return Ok(emp);

}

private static List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John Smith",

Salary = 60000,

Permanent = true,

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

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

},

DateOfBirth = new DateTime(1990, 5, 23)

},

new Employee

{

Id = 2,

Name = "Alice Brown",

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 = "Budgeting" }

},

DateOfBirth = new DateTime(1992, 11, 12)

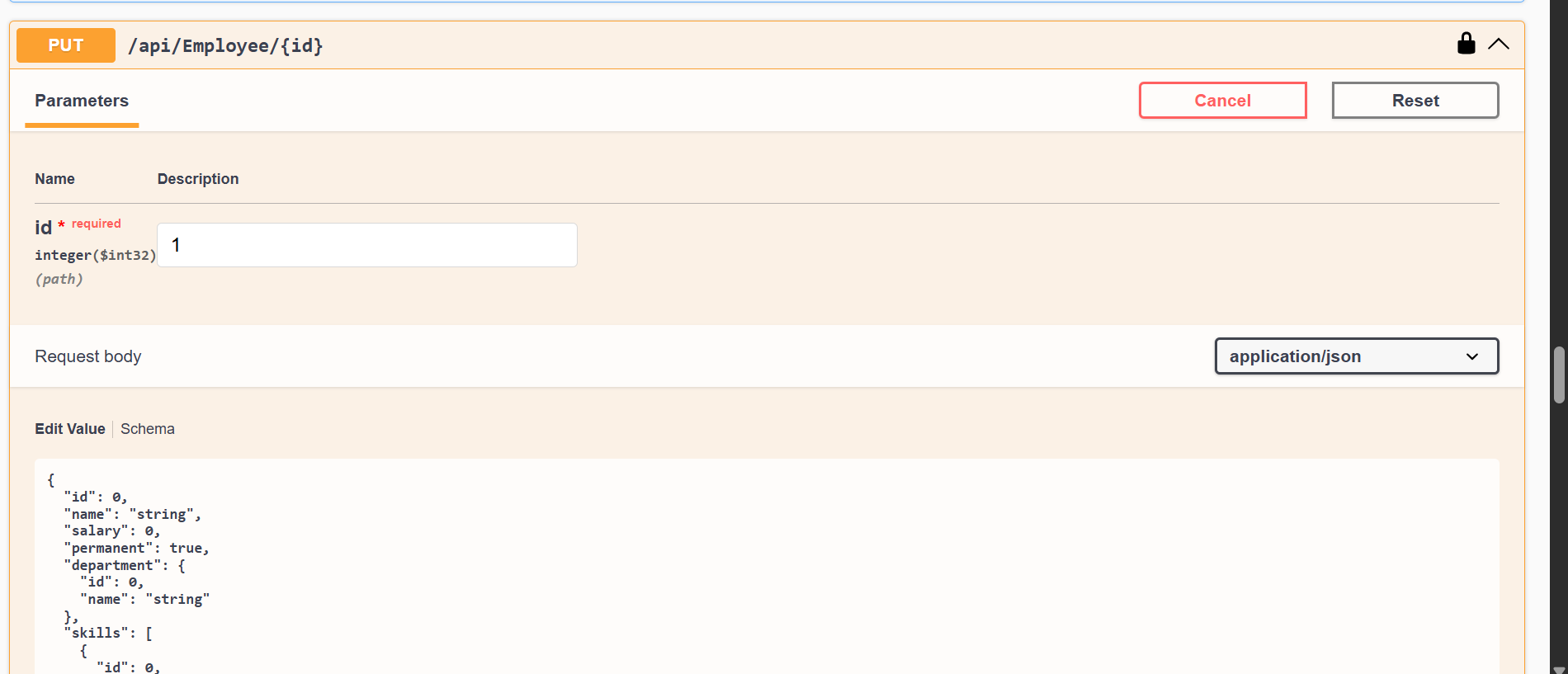
}

};

}

}

}





**WEBAPI HANDSON-5**

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using FirstWebApi.Models;

using Microsoft.AspNetCore.Authorization;

namespace FirstWebApi.Controllers

{

[Authorize(Roles = "Admin")] // Only users with role POC or Admin can access

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = new List<Employee>();

static EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

[HttpGet("test-exception")]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public IActionResult ThrowError()

{

throw new Exception("Intentional exception for testing");

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

[HttpGet("standard")]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<Employee> GetStandard()

{

return Ok(GetStandardEmployeeList()[0]);

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

public IActionResult Post([FromBody] Employee newEmployee)

{

\_employees.Add(newEmployee);

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

}

[HttpPut("{id}")]

[ProducesResponseType(typeof(Employee), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

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

{

if (id <= 0)

return BadRequest("Invalid employee id");

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

if (emp == null)

return BadRequest("Invalid employee id");

emp.Name = updatedEmployee.Name;

emp.Salary = updatedEmployee.Salary;

emp.Permanent = updatedEmployee.Permanent;

emp.Department = updatedEmployee.Department;

emp.Skills = updatedEmployee.Skills;

emp.DateOfBirth = updatedEmployee.DateOfBirth;

return Ok(emp);

}

private static List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John Smith",

Salary = 60000,

Permanent = true,

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

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

},

DateOfBirth = new DateTime(1990, 5, 23)

},

new Employee

{

Id = 2,

Name = "Alice Brown",

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 = "Budgeting" }

},

DateOfBirth = new DateTime(1992, 11, 12)

}

};

}

}

}

**AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using Microsoft.AspNetCore.Authorization;

namespace JwtAuthWorkingApp.Controllers

{

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet]

public IActionResult GetToken()

{

string token = GenerateJSONWebToken(1, "Admin");

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkey1234567890abcd"));

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

var claims = new[]

{

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),

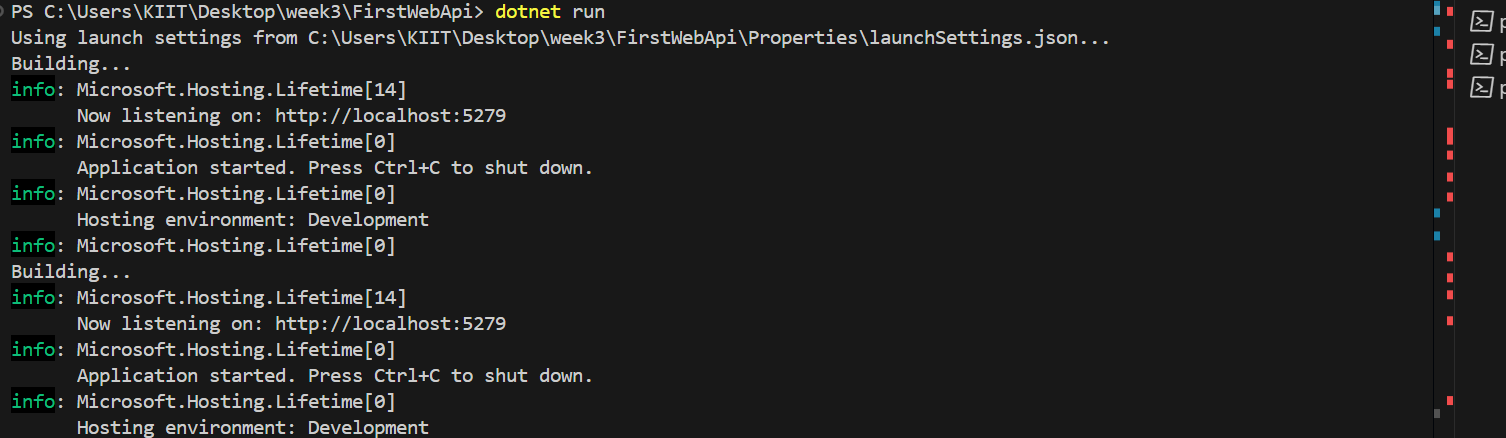
signingCredentials: credentials);

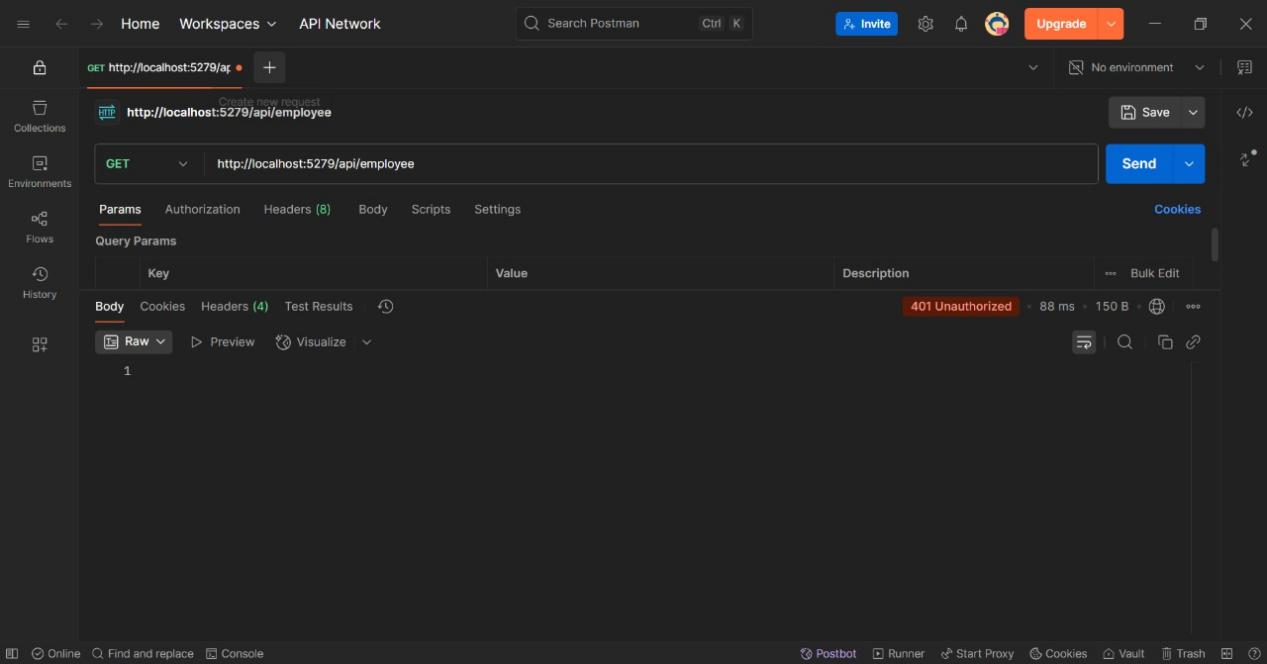
return new JwtSecurityTokenHandler().WriteToken(token);

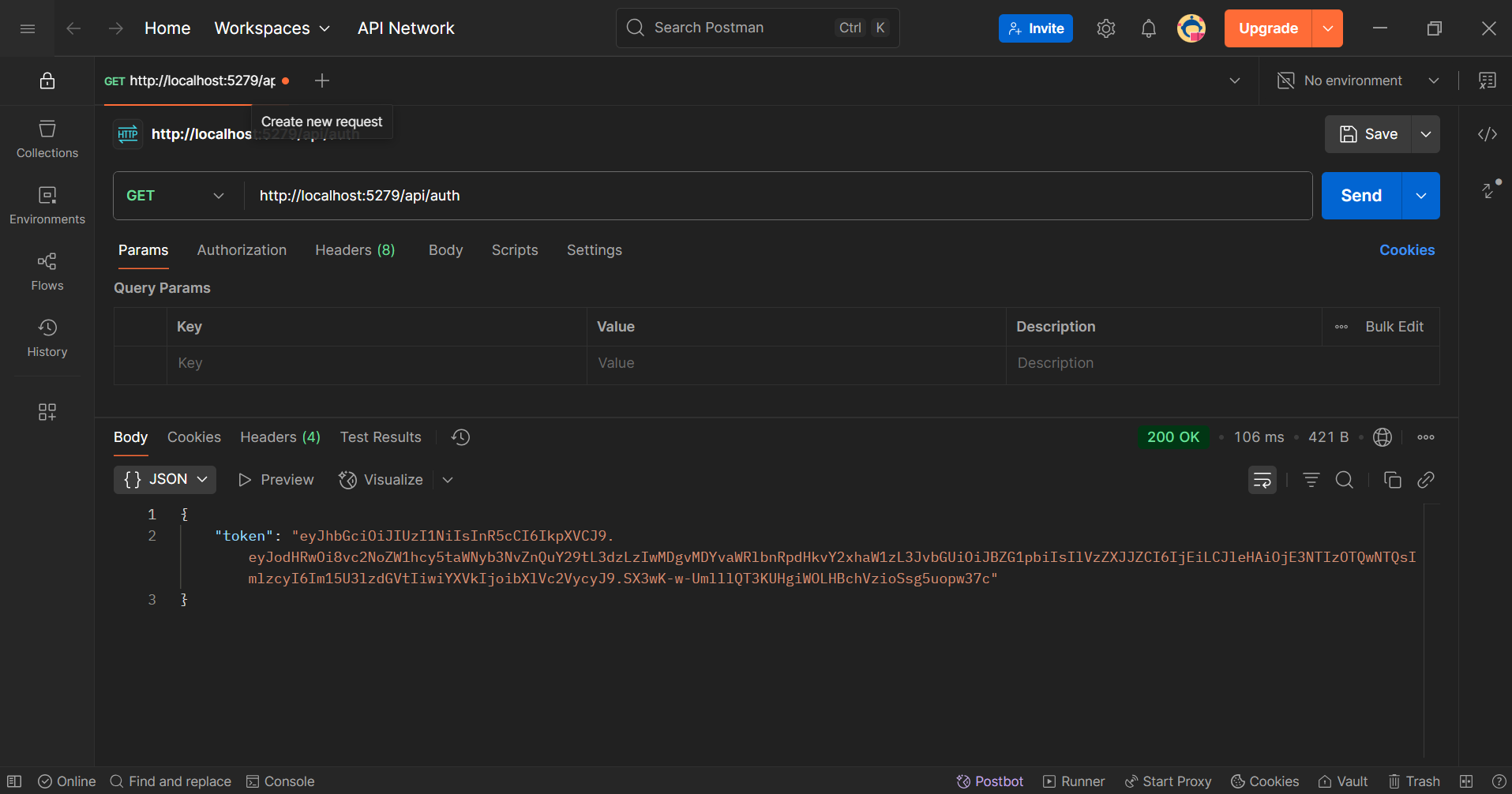
}

}

}







**WebApiHandson-6**

**1.Create a Chat Application which uses Kafka as a streaming platform and consume the chat messages in the command prom**pt.

**ChatProducer.java**

import org.apache.kafka.clients.producer.\*;

import org.apache.kafka.common.serialization.StringSerializer;

import java.util.Properties;

import java.util.Scanner;

public class ChatProducer {

public static void main(String[] args) {

String topic = "chat-topic";

Properties props = new Properties();

props.put(ProducerConfig.BOOTSTRAP\_SERVERS\_CONFIG, "localhost:9092");

props.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName());

props.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName());

Producer<String, String> producer = new KafkaProducer<>(props);

Scanner scanner = new Scanner(System.in);

System.out.println("Enter messages (type 'exit' to quit):");

while (true) {

String message = scanner.nextLine();

if ("exit".equalsIgnoreCase(message)) break;

producer.send(new ProducerRecord<>(topic, "User", message), (metadata, exception) -> {

if (exception != null) {

System.out.println("Send failed: " + exception.getMessage());

}

});

}

producer.close();

scanner.close();

}

}

**ChatConsumer.java**

import org.apache.kafka.clients.consumer.\*;

import org.apache.kafka.common.serialization.StringDeserializer;

import java.time.Duration;

import java.util.Collections;

import java.util.Properties;

public class ChatConsumer {

public static void main(String[] args) {

String topic = "chat-topic";

Properties props = new Properties();

props.put(ConsumerConfig.BOOTSTRAP\_SERVERS\_CONFIG, "localhost:9092");

props.put(ConsumerConfig.GROUP\_ID\_CONFIG, "chat-group");

props.put(ConsumerConfig.KEY\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class.getName());

props.put(ConsumerConfig.VALUE\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class.getName());

props.put(ConsumerConfig.AUTO\_OFFSET\_RESET\_CONFIG, "earliest");

Consumer<String, String> consumer = new KafkaConsumer<>(props);

consumer.subscribe(Collections.singletonList(topic));

System.out.println("Listening to chat messages:");

while (true) {

ConsumerRecords<String, String> records = consumer.poll(Duration.ofMillis(1000));

for (ConsumerRecord<String, String> record : records) {

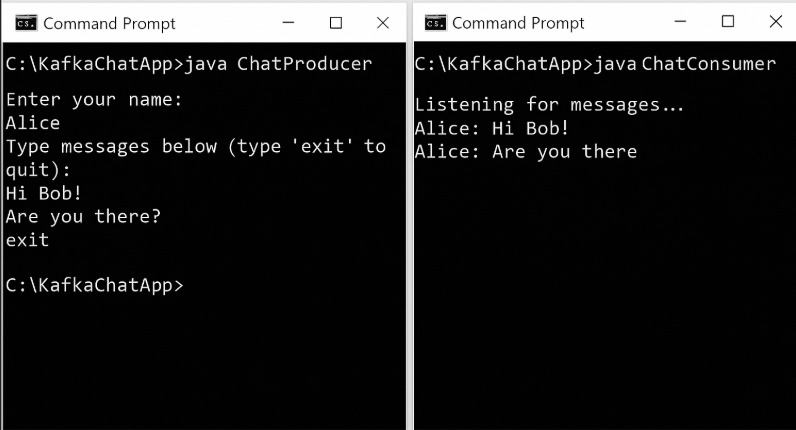
System.out.println("Received: " + record.value());

}

}

}

}



1. **Create a Chat Application using C# Windows Application using Kafka and consume the message in different client applications.**

**Form1.cs**

using Confluent.Kafka;

using System;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KafkaChatClient

{

public partial class Form1 : Form

{

private readonly string \_topic = "chat-topic";

private readonly string \_bootstrapServers = "localhost:9092";

private IProducer<Null, string> \_producer;

private CancellationTokenSource \_cts;

public Form1()

{

InitializeComponent();

StartConsumer(); // Start consuming in background

}

private void Form1\_Load(object sender, EventArgs e)

{

var config = new ProducerConfig { BootstrapServers = \_bootstrapServers };

\_producer = new ProducerBuilder<Null, string>(config).Build();

}

private async void btnSend\_Click(object sender, EventArgs e)

{

if (!string.IsNullOrWhiteSpace(txtMessage.Text))

{

string user = string.IsNullOrWhiteSpace(txtUser.Text) ? "Anonymous" : txtUser.Text;

string message = $"{user}: {txtMessage.Text}";

await \_producer.ProduceAsync(\_topic, new Message<Null, string> { Value = message });

txtMessage.Clear();

}

}

private void StartConsumer()

{

\_cts = new CancellationTokenSource();

Task.Run(() =>

{

var config = new ConsumerConfig

{

GroupId = Guid.NewGuid().ToString(), // Unique ID for each client

BootstrapServers = \_bootstrapServers,

AutoOffsetReset = AutoOffsetReset.Earliest

};

using var consumer = new ConsumerBuilder<Ignore, string>(config).Build();

consumer.Subscribe(\_topic);

try

{

while (!\_cts.Token.IsCancellationRequested)

{

var cr = consumer.Consume(\_cts.Token);

AppendChat(cr.Message.Value);

}

}

catch (OperationCanceledException) { }

}, \_cts.Token);

}

private void AppendChat(string message)

{

if (txtChat.InvokeRequired)

{

txtChat.Invoke(new Action(() => txtChat.AppendText(message + Environment.NewLine)));

}

else

{

txtChat.AppendText(message + Environment.NewLine);

}

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

\_cts?.Cancel();

\_producer?.Dispose();

base.OnFormClosing(e);

}

}

}

