**DN 4.0 Dotnet FSE**

**Name: Aditya Mohanty**

**Superset ID:6361043**

**Week 4-**

**Web API**

**LAB 1: First Web Api using .Net core**

Standard controller code:

[HttpGet]

public IEnumerable<string> Get() => new string[] { "value1", "value2" };

[HttpGet("{id}")]

public string Get(int id) => "value";

[HttpPost]

public void Post([FromBody] string value) { }

[HttpPut("{id}")]

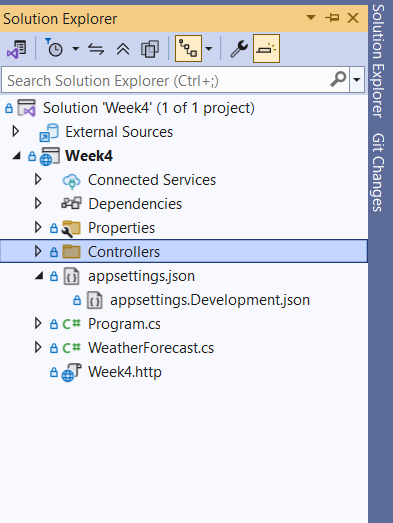
public void Put(int id, [FromBody] string value) { }

[HttpDelete("{id}")]

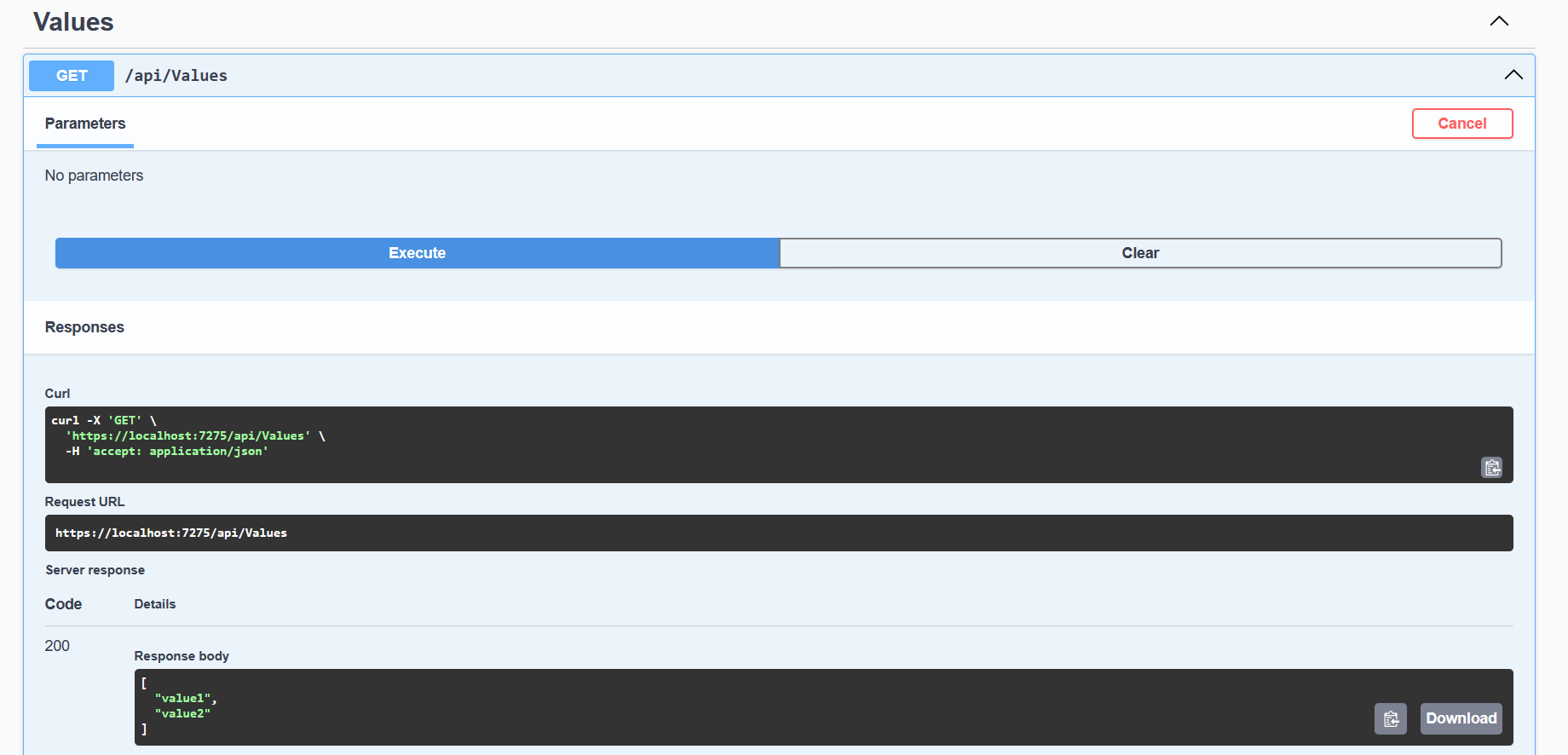
public void Delete(int id) { }

OUTPUT:

Web Api files after creation



Testing GET on Swagger



**LAB 2:Web Api using .Net core with Swagger**

Code for configuring Swagger(including middleware):

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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

Contact = new Microsoft.OpenApi.Models.OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

},

License = new Microsoft.OpenApi.Models.OpenApiLicense

{

Name = "License Terms",

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

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

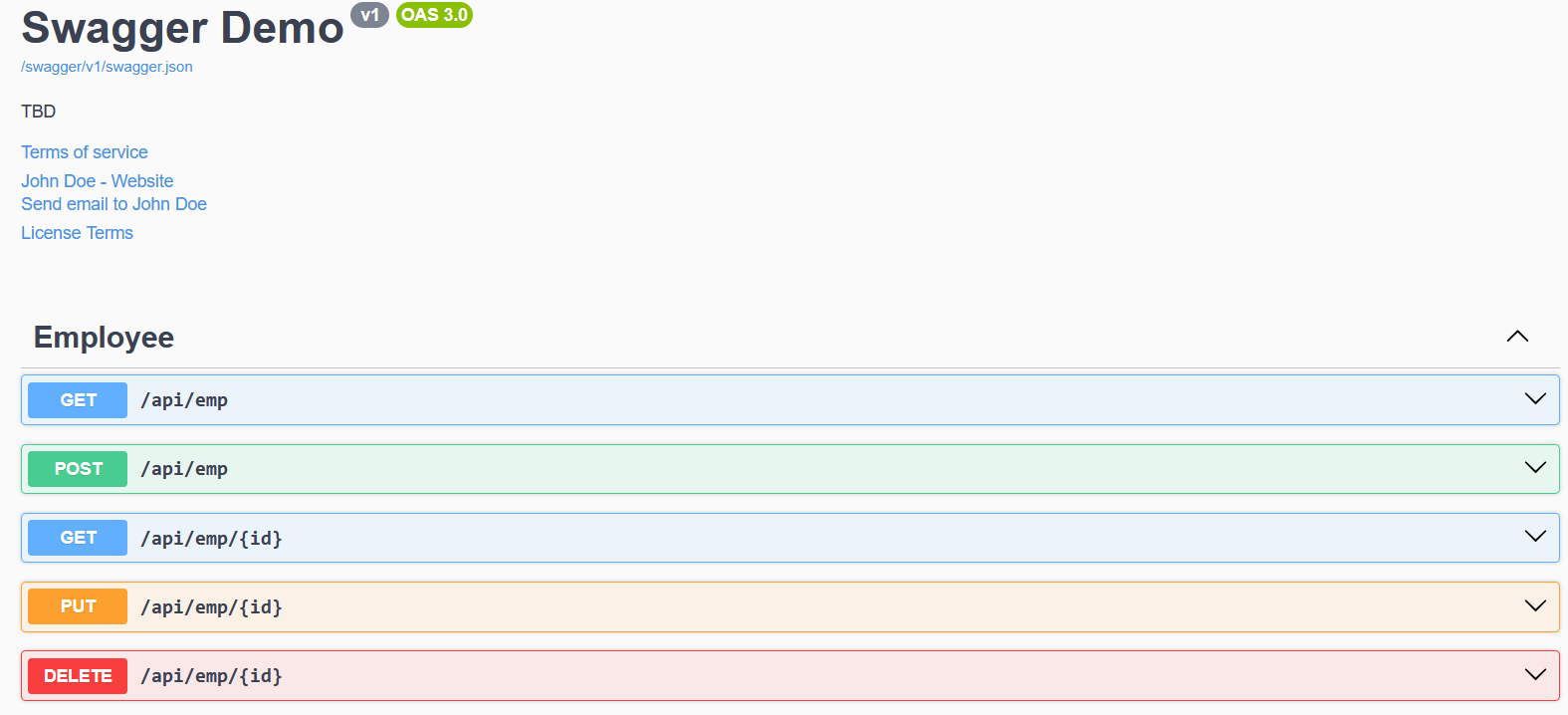
app.UseSwaggerUI(c =>

{

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

});

}

OUTPUT:

**LAB 3:Web Api using custom model class**

Code for classes in Models folder:

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

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; }

}

Code for Employee Controller:

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using Week4.Models;

[Authorize]

[ApiController]

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

public class EmployeeController : ControllerBase

{

private readonly List<Employee> \_employees;

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John Smith",

Salary = 50000,

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, 1, 1)

},

new Employee

{

Id = 2,

Name = "Jane Doe",

Salary = 60000,

Permanent = false,

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

Skills = new List<Skill>

{

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

},

DateOfBirth = new DateTime(1992, 5, 12)

}

};

}

[HttpGet]

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

[ProducesResponseType(500)]

public ActionResult<List<Employee>> Get()

{

throw new Exception("Test exception");

}

[HttpGet("standard")]

[ProducesResponseType(typeof(Employee), 200)]

public ActionResult<Employee> GetStandard()

{

return Ok(\_employees[0]);

}

[HttpPost]

public IActionResult Post([FromBody] Employee employee)

{

\_employees.Add(employee);

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

}

[HttpPut("{id}")]

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

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

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

if (existingEmployee == null)

{

return BadRequest("Invalid employee id");

}

existingEmployee.Name = employee.Name;

existingEmployee.Salary = employee.Salary;

existingEmployee.Permanent = employee.Permanent;

existingEmployee.Department = employee.Department;

existingEmployee.Skills = employee.Skills;

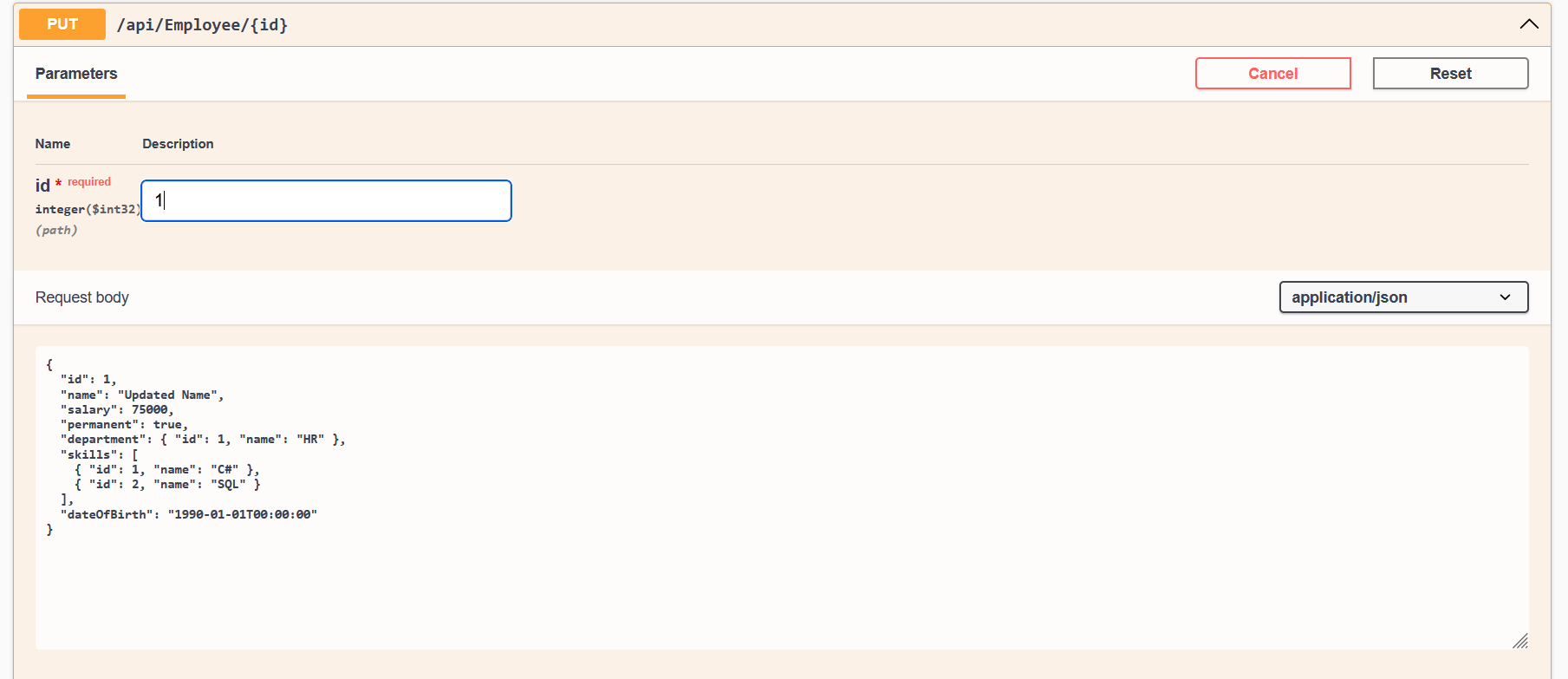
existingEmployee.DateOfBirth = employee.DateOfBirth;

return Ok(existingEmployee);

}

}

**OUPUT:**

****

**LAB 4:Web Api CRUD OPERATION**

EmployeeController Code for CRUD:

[HttpPut("{id}")]

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

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

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

if (existingEmployee == null)

{

return BadRequest("Invalid employee id");

}

existingEmployee.Name = employee.Name;

existingEmployee.Salary = employee.Salary;

existingEmployee.Permanent = employee.Permanent;

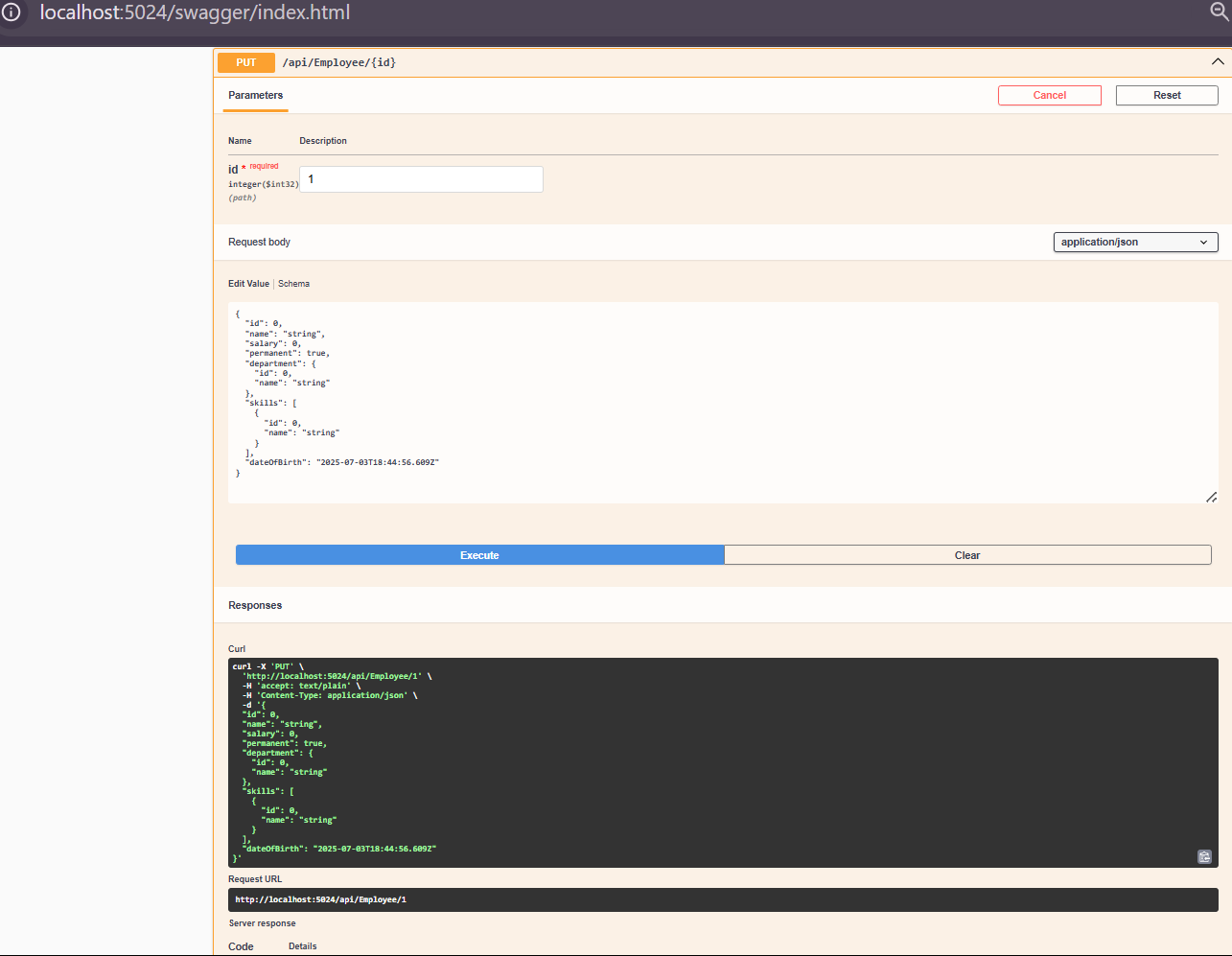
existingEmployee.Department = employee.Department;

existingEmployee.Skills = employee.Skills;

existingEmployee.DateOfBirth = employee.DateOfBirth;

return Ok(existingEmployee);

OUTPUT:



**LAB 5:CORS and Security in WEB API**

**Code for authcontroller:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

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(10), // Set to 2 for expiration test

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

[HttpGet("token")]

public IActionResult GetToken()

{

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

return Ok(new { token });

}

}

CODE for program.cs

using Microsoft.IdentityModel.Tokens;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

string securityKey = "mysuperdupersecret";

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

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultSignInScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, x =>

{

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

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

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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

Contact = new Microsoft.OpenApi.Models.OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

},

License = new Microsoft.OpenApi.Models.OpenApiLicense

{

Name = "License Terms",

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

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

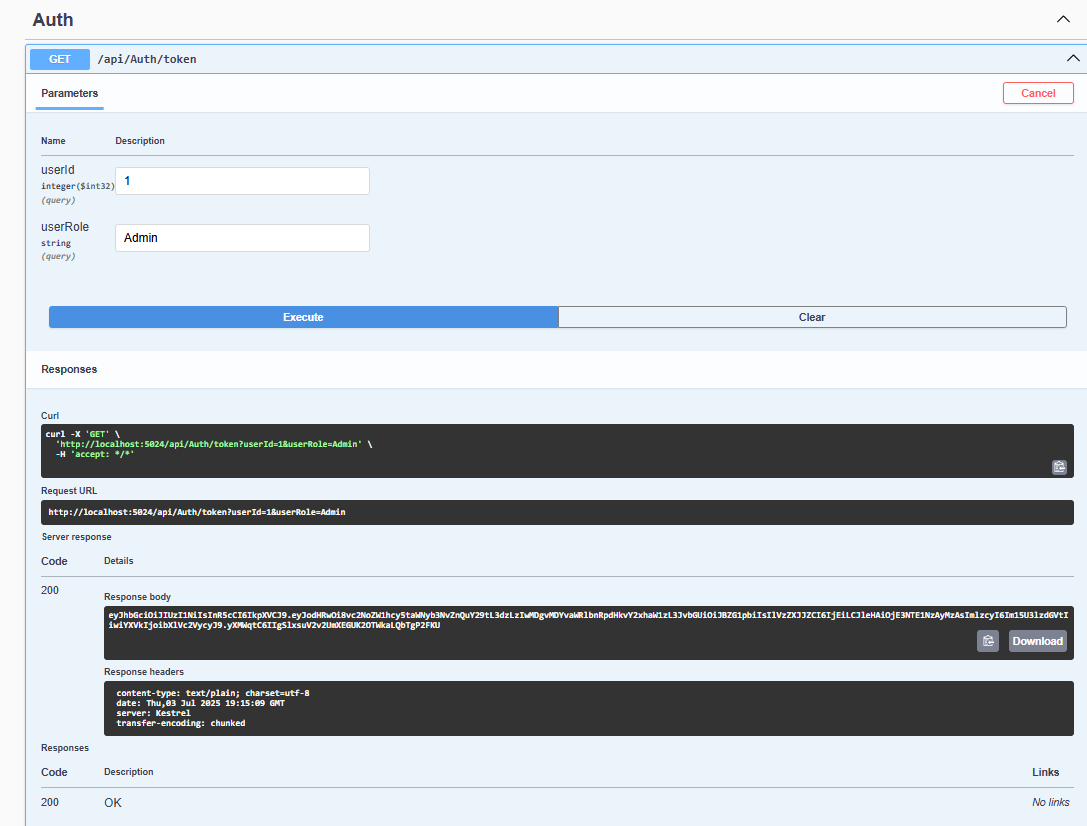
app.UseCors();

app.UseAuthentication();

app.MapControllers();

app.Run();

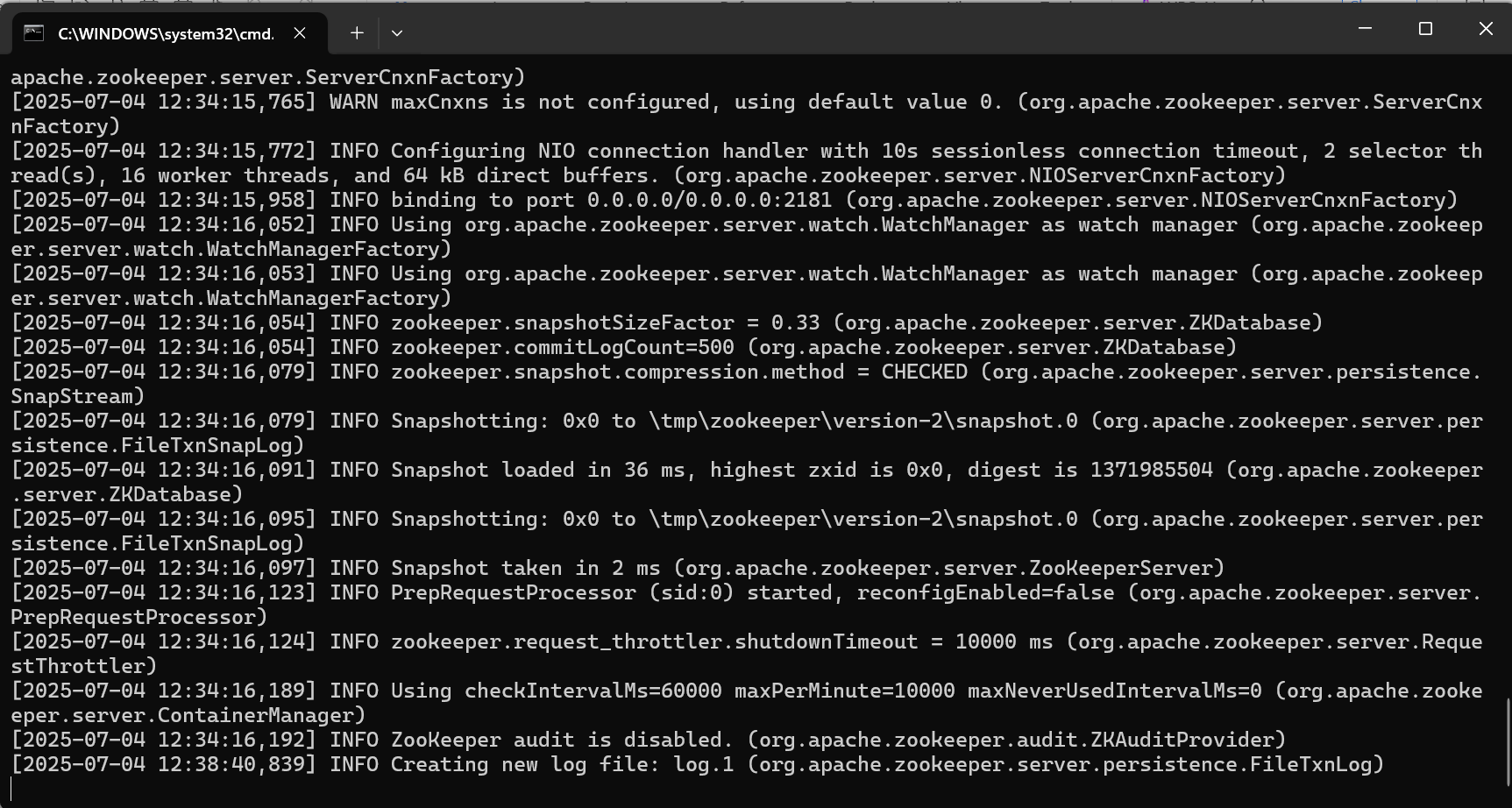
OUTPUT:



**LAB 6:KAFKA INTEGRATION WITH C#**

**1  
Start ZooKeeper**  
cd C:\kafka\_2.13-3.7.2  
bin\windows\zookeeper-server-start.bat config\zookeeper.properties  
  
**| 2 | Start Kafka Broker |**  
cd C:\kafka\_2.13-3.7.2  
bin\windows\kafka-server-start.bat config\server.properties  
  
**| 3 | Create Kafka Topic (if not already created) |**  
cd C:\kafka\_2.13-3.7.2  
bin\windows\kafka-topics.bat --create --topic chat-topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1  
 **| 4 | Run Kafka Consumer Console App |**  
cd path\_to\_KafkaChatConsumer\_project  
dotnet run  
 **| 5 | Run Kafka Producer Console App |**  
cd path\_to\_KafkaChatProducer\_project  
dotnet run  
  
**| 6 | Send and Receive Messages |**  
Type chat messages in the producer console and see them appear in the consumer console.

**Output:**

****