**Week-04 HandsOn Solution**

* **WebApi**

**Exercise-1: Create a Web API with GET & POST**

* **Code:**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace WebApiDemo.Controllers

{

    [ApiController]

    [Route("[controller]")]

    public class ValuesController : ControllerBase

    {

        private static List<string> values = new() { "value1", "value2" };

        [HttpGet]

        public IEnumerable<string> Get() => values;

        [HttpPost]

        public IActionResult Post([FromBody] string value)

        {

            values.Add(value);

            return Ok(values);

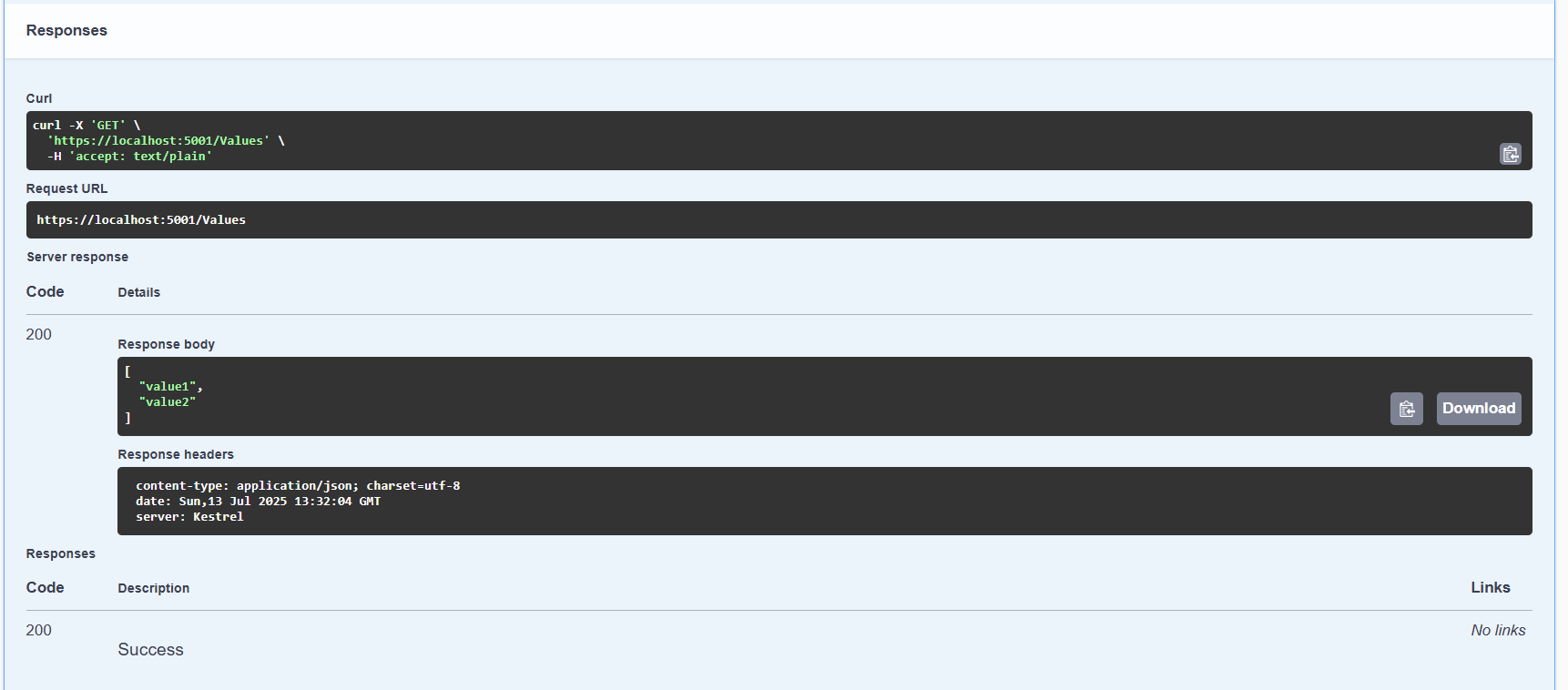
        }

    }

}

* **Output:**

**//GET/Values**

****

**//POST/Values**

****

**Exercise-2: Enable Swagger in Web API**

* **Code:**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

using WebApiDemo.Filters;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "WebApiDemo", Version = "v1" });

    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

    {

        Description = @"JWT Authorization header using the Bearer scheme.

                        Enter 'Bearer' followed by your token.

                        Example: Bearer eyJhbGciOiJIUzI1...",

        Name = "Authorization",

        In = ParameterLocation.Header,

        Type = SecuritySchemeType.ApiKey,

        Scheme = "Bearer"

});

    c.AddSecurityRequirement(new OpenApiSecurityRequirement

    {

        {

            new OpenApiSecurityScheme

            {

                Reference = new OpenApiReference

                {

                    Type = ReferenceType.SecurityScheme,

                    Id = "Bearer"

                },

                Scheme = "oauth2",

                Name = "Bearer",

                In = ParameterLocation.Header,

            },

            new List<string>()

        }

    });

});

var key = Encoding.ASCII.GetBytes("mysuperdupersecret");

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

    .AddJwtBearer(options =>

    {

        options.TokenValidationParameters = new TokenValidationParameters

        {

            ValidateIssuer = true,

            ValidateAudience = true,

            ValidateLifetime = true,

            ValidateIssuerSigningKey = true,

            ValidIssuer = "mySystem",

            ValidAudience = "myUsers",

            IssuerSigningKey = new SymmetricSecurityKey(key)

        };

});

builder.Services.AddScoped<CustomAuthFilter>();

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseHttpsRedirection();

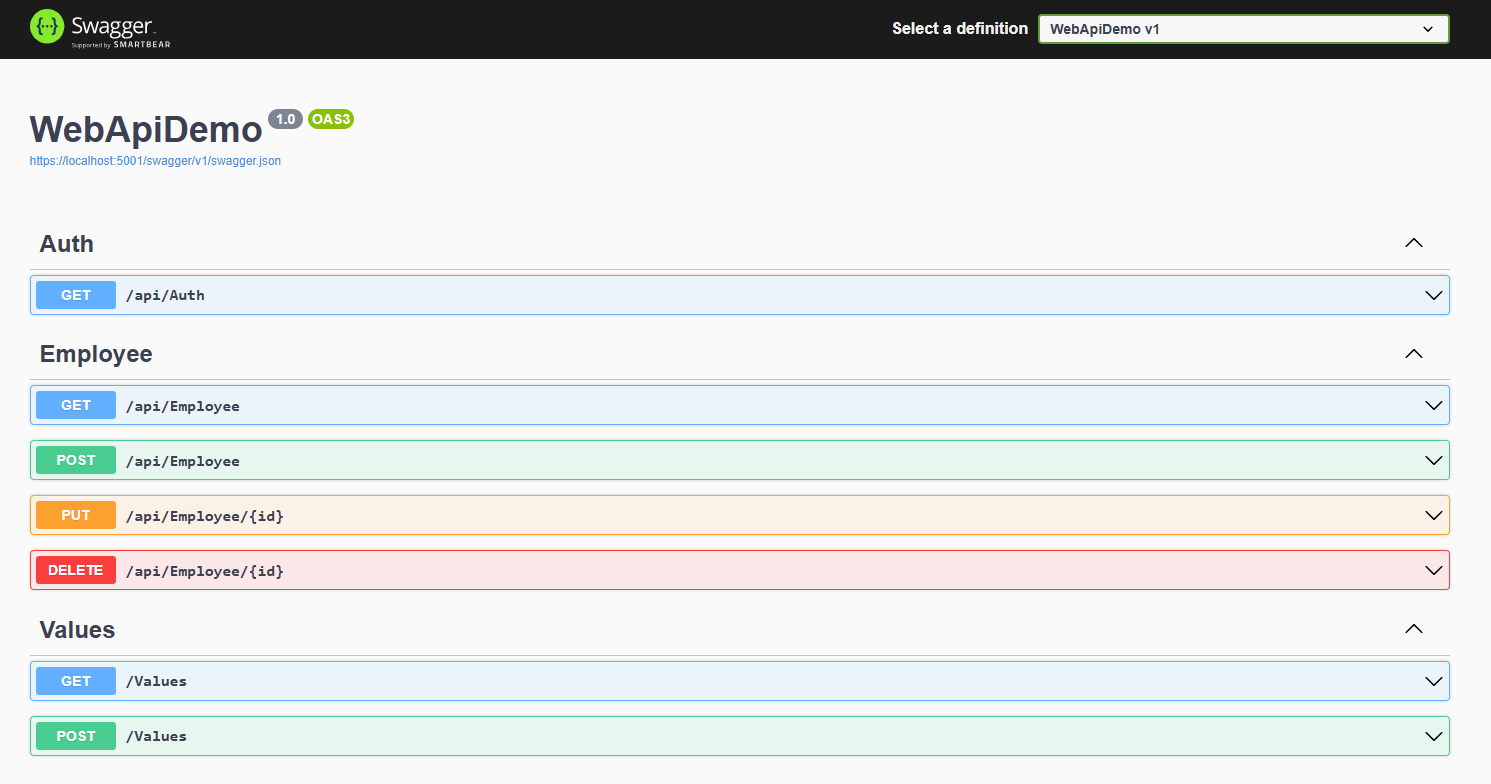
app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

* **Output:**

****

**Exercise-3: Create Model & Custom Filter**

* **Code:**

**//Models/Employee.cs**

using System;

using System.Collections.Generic;

namespace WebApiDemo.Models

{

    public class Skill { public string SkillName { get; set; } }

public class Department { public string DeptName { 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; }

    }

}

**//Filters/CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebApiDemo.Filters

{

    public class CustomAuthFilter : ActionFilterAttribute

    {

        public override void OnActionExecuting(ActionExecutingContext context)

        {

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

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

            {

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

            }

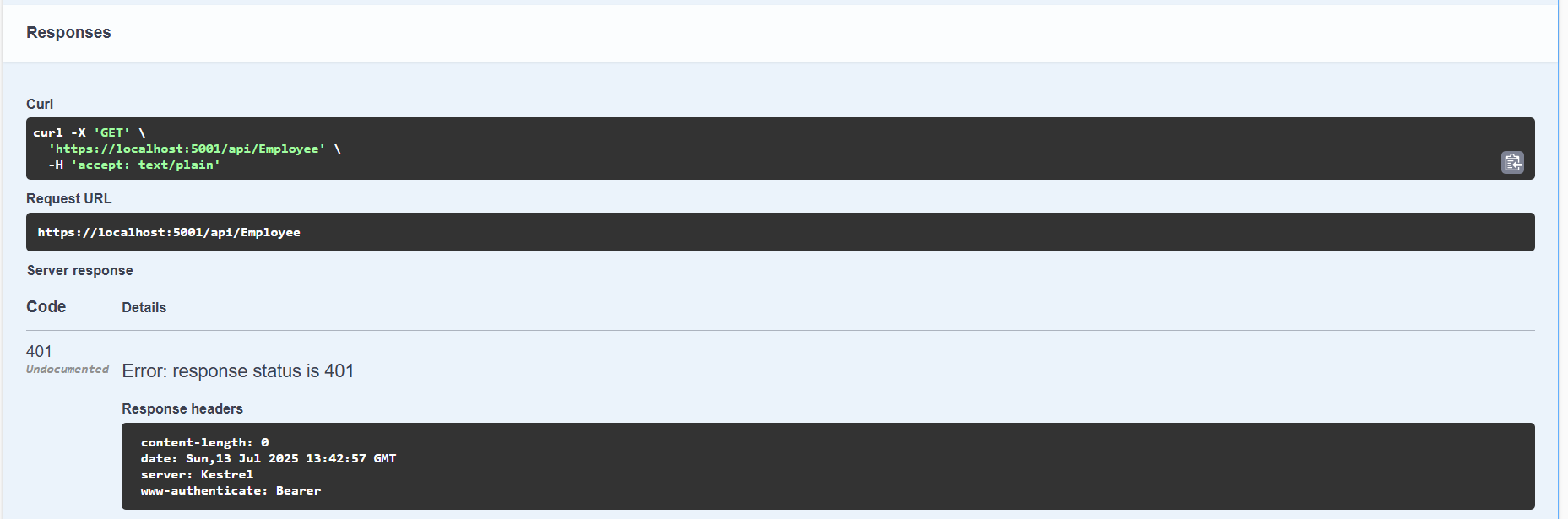
        }

    }

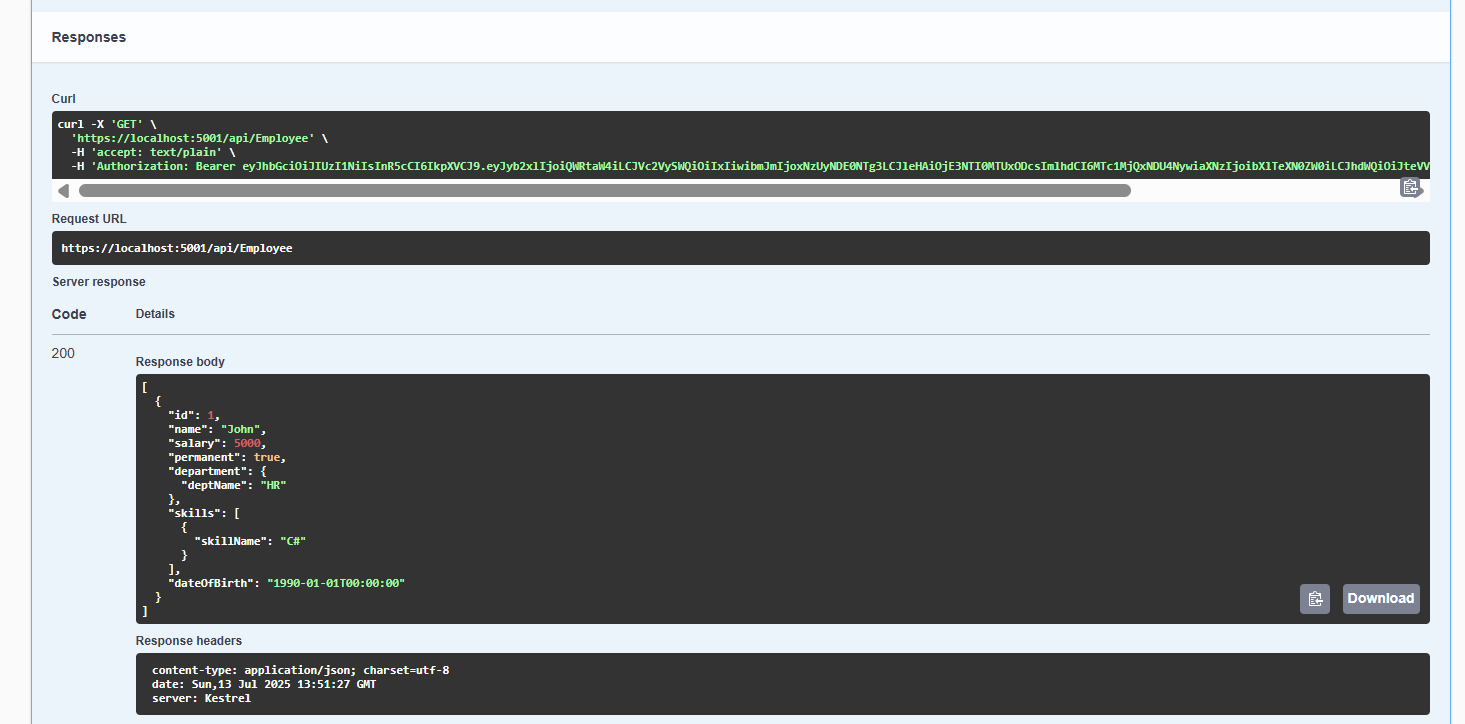
}

* **Output:**

**//Unauthorize** (Status Code: 401)

****

**//Authorize** (Status Code: 200)

****

**Exercise-4: Add PUT & DELETE for Employee API**

* **Code:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Authorization;

using WebApiDemo.Models;

using System.Collections.Generic;

using System.Linq;

using System;

namespace WebApiDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [Authorize(Roles = "Admin")]

    public class EmployeeController : ControllerBase

    {

        private static List<Employee> employees = new()

        {

            new Employee

            {

                Id = 1,

                Name = "John",

                Salary = 5000,

                Permanent = true,

                Department = new Department { DeptName = "HR" },

                Skills = new List<Skill> { new Skill { SkillName = "C#" } },

                DateOfBirth = new DateTime(1990, 1, 1)

            }

        };

        [HttpGet]

        public ActionResult<List<Employee>> Get() => employees;

        [HttpPost]

        public IActionResult Post([FromBody] Employee emp)

        {

            employees.Add(emp);

            return Ok(emp);

        }

        [HttpPut("{id}")]

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

        {

            if (id <= 0)

                return BadRequest("Invalid employee id");

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

            if (existing == null)

                return BadRequest("Invalid employee id");

            existing.Name = emp.Name;

            return Ok(existing);

        }

        [HttpDelete("{id}")]

        public IActionResult Delete(int id)

        {

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

            if (existing == null)

                return NotFound();

            employees.Remove(existing);

            return NoContent();

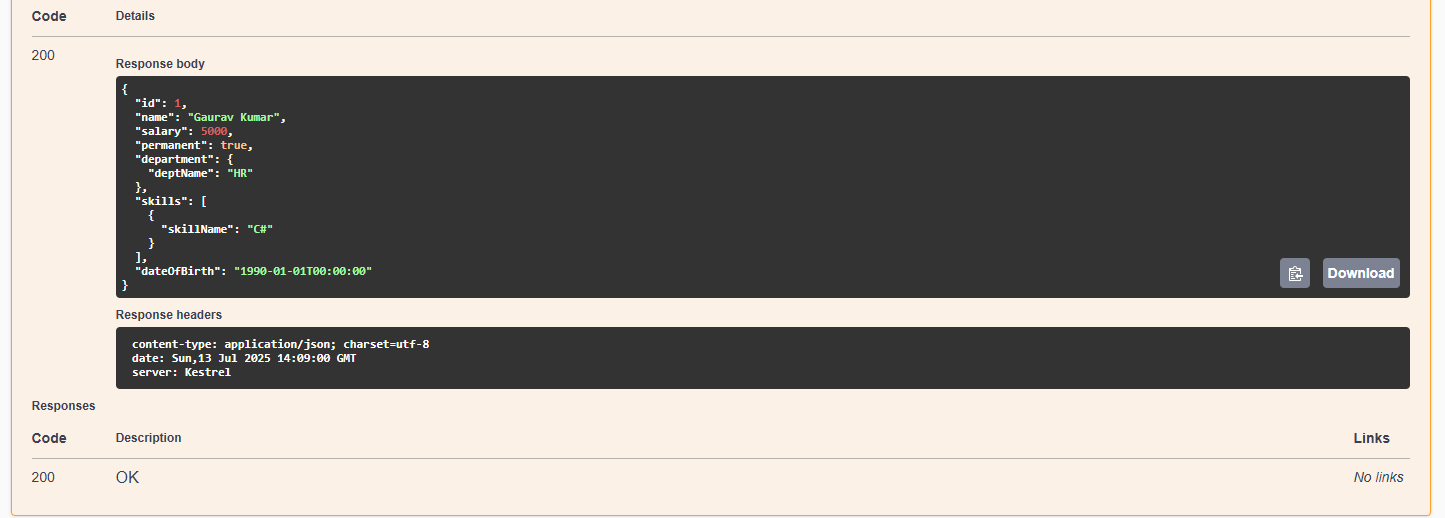
        }

    }

}

* **Output:**

**//PUT/api/Employee/{id}**

****

**//DELETE/api/Employee/{id}**

****

**Exercise-5: Implement JWT Token Authentication**

* **Code:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Authorization;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System;

namespace WebApiDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [AllowAnonymous]

    public class AuthController : ControllerBase

    {

        [HttpGet]

        public string GetToken()

        {

            var key = Encoding.ASCII.GetBytes("mysuperdupersecret");

            var tokenHandler = new JwtSecurityTokenHandler();

            var tokenDescriptor = new SecurityTokenDescriptor

            {

                Subject = new ClaimsIdentity(new[]

                {

                    new Claim(ClaimTypes.Role, "Admin"),

                    new Claim("UserId", "1")

                }),

                Expires = DateTime.UtcNow.AddMinutes(10),

                Issuer = "mySystem",

                Audience = "myUsers",

                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)

            };

            var token = tokenHandler.CreateToken(tokenDescriptor);

            return tokenHandler.WriteToken(token);

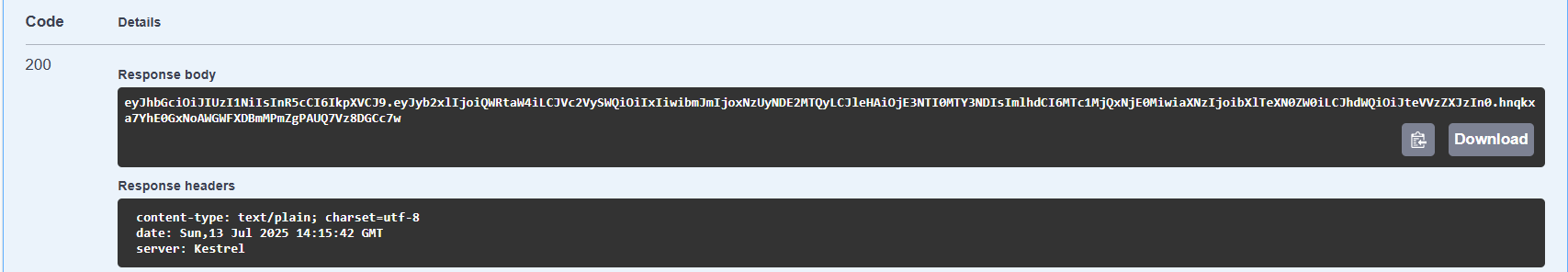
        }

    }

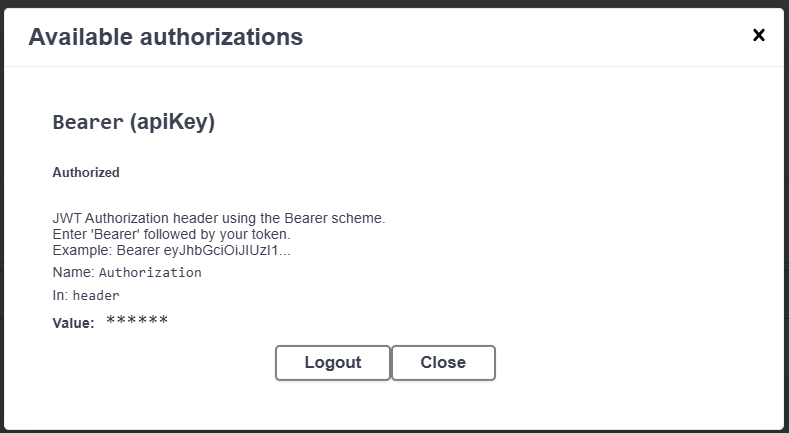
}

* **Output:**

**//Token Generation**

****

**//Authorization**

****

**Exercise-6: Kafka Producer and Consumer**

* **Code:**

**//Kafka/KafkaProducer.cs**

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

namespace WebApiDemo.Kafka

{

    public class KafkaProducer

    {

        public static async Task Produce(string topic, string message)

        {

            var config = new ProducerConfig { BootstrapServers = "localhost:9092" };

            using var producer = new ProducerBuilder<Null, string>(config).Build();

            var result = await producer.ProduceAsync(topic, new Message<Null, string> { Value = message });

            Console.WriteLine($"Delivered to: {result.TopicPartitionOffset}");

        }

    }

}

**//Kafka/KafkaConsumers.cs**

using Confluent.Kafka;

using System;

using System.Threading;

namespace WebApiDemo.Kafka

{

    public class KafkaConsumer

    {

        public static void Consume(string topic)

        {

            var config = new ConsumerConfig

            {

                BootstrapServers = "localhost:9092",

                GroupId = "test-consumer-group",

                AutoOffsetReset = AutoOffsetReset.Earliest

            };

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

            consumer.Subscribe(topic);

            CancellationTokenSource cts = new CancellationTokenSource();

            try

            {

                while (true)

                {

                    var cr = consumer.Consume(cts.Token);

                    Console.WriteLine($"Consumed message: {cr.Value}");

                }

            }

            catch (OperationCanceledException)

            {

                consumer.Close();

            }

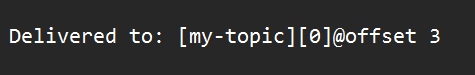
        }

    }

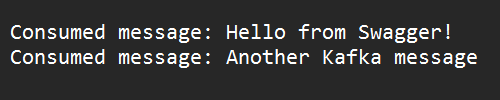
}

* **Output:**

**//Producer**

****

**//Consumer**

****