**Cognizant Deep Skilling Week-4**

ASP .NET Core 8.0 Web API

1. **Demonstrate creation of a simple WebAPI** **:-**

ValuesController.cs:

using Microsoft.AspNetCore.Mvc;

[ApiController]

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

public class ValuesController : ControllerBase

{

    [HttpGet]

    public IActionResult Get() => Ok(new[] { "value1", "value2" });

    [HttpPost]

    public IActionResult Post([FromBody] string value) => Ok($"Received: {value}");

}

Program.cs:

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

    options.SwaggerDoc("v1", new OpenApiInfo

    {

        Title = "MyFirstWebAPI",

        Version = "v1"

    });

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI(c =>

    {

        c.SwaggerEndpoint("/swagger/v1/swagger.json", "MyFirstWebAPI V1");

    });

}

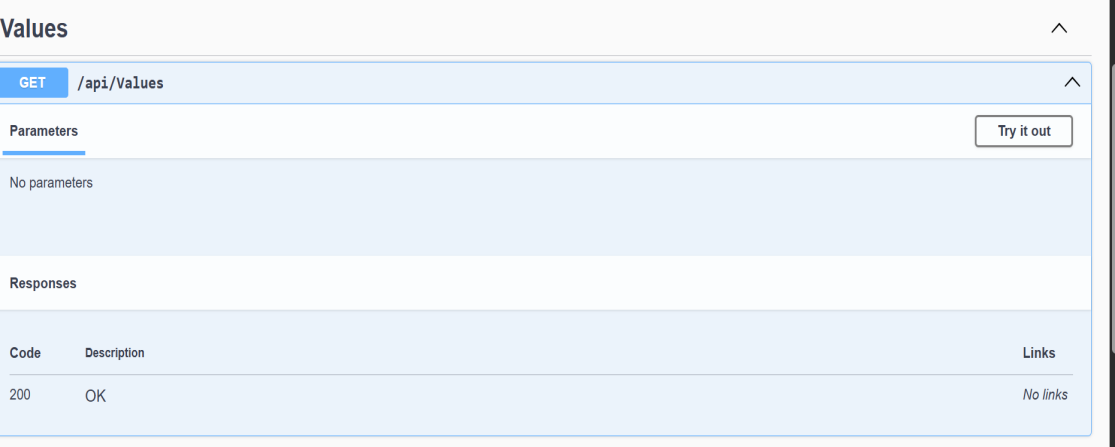
app.UseHttpsRedirection();

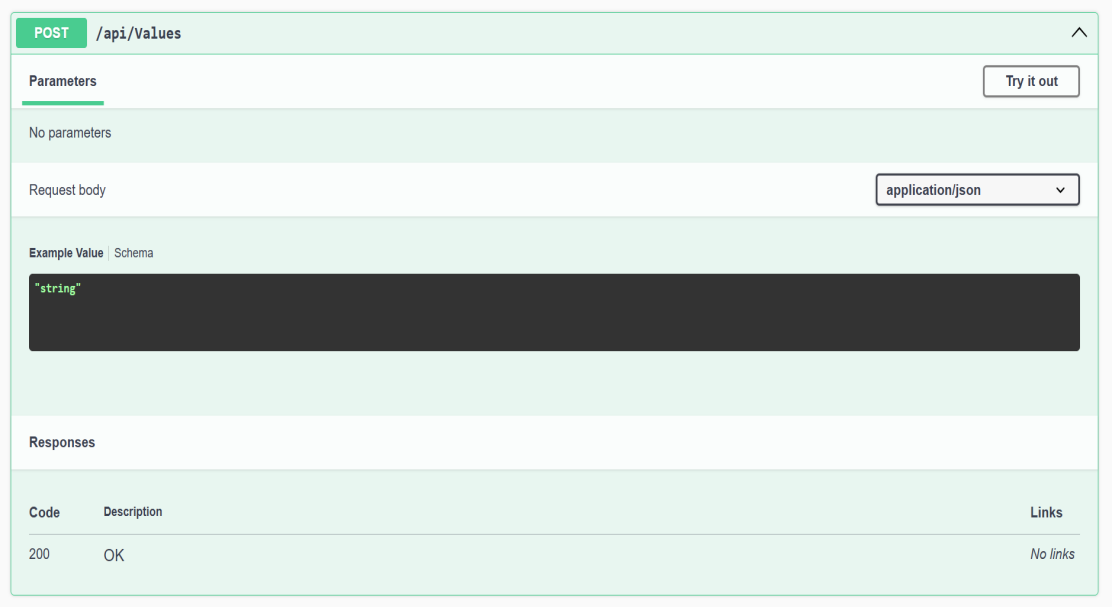
app.UseAuthorization();

app.MapControllers();

app.Run();

OUTPUT:-





**2.Demonstrate Swagger installation to WebAPI and WebAPI listing on browser:-**

1.Create a .Net core web application with API template.:-

EmplyeeController.cs:

using Microsoft.AspNetCore.Mvc;

namespace SwaggerDemoAPI.Controllers

{

    [ApiController]

    [Route("[controller]")]

    public class EmployeeController : ControllerBase

    {

        static List<string> employees = new List<string> { "Alice", "Bob", "Charlie" };

        [HttpGet]

        [ProducesResponseType(200)]

        public IActionResult GetEmployees()

        {

            return Ok(employees);

        }

        [HttpPost]

        [ProducesResponseType(201)]

        public IActionResult AddEmployee([FromBody] string name)

        {

            employees.Add(name);

            return Created("", name);

        }

        [HttpGet("{id}")]

        [ProducesResponseType(200)]

        [ProducesResponseType(404)]

        public IActionResult GetEmployee(int id)

        {

            if (id >= employees.Count) return NotFound();

            return Ok(employees[id]);

        }

    }

}

Program.cs:

using Microsoft.OpenApi.Models;

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://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.MapControllers();

var summaries = new[]

{

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

};

app.MapGet("/weatherforecast", () =>

{

    var forecast = Enumerable.Range(1, 5).Select(index =>

        new WeatherForecast

        (

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

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

            summaries[Random.Shared.Next(summaries.Length)]

        ))

        .ToArray();

    return forecast;

})

.WithName("GetWeatherForecast")

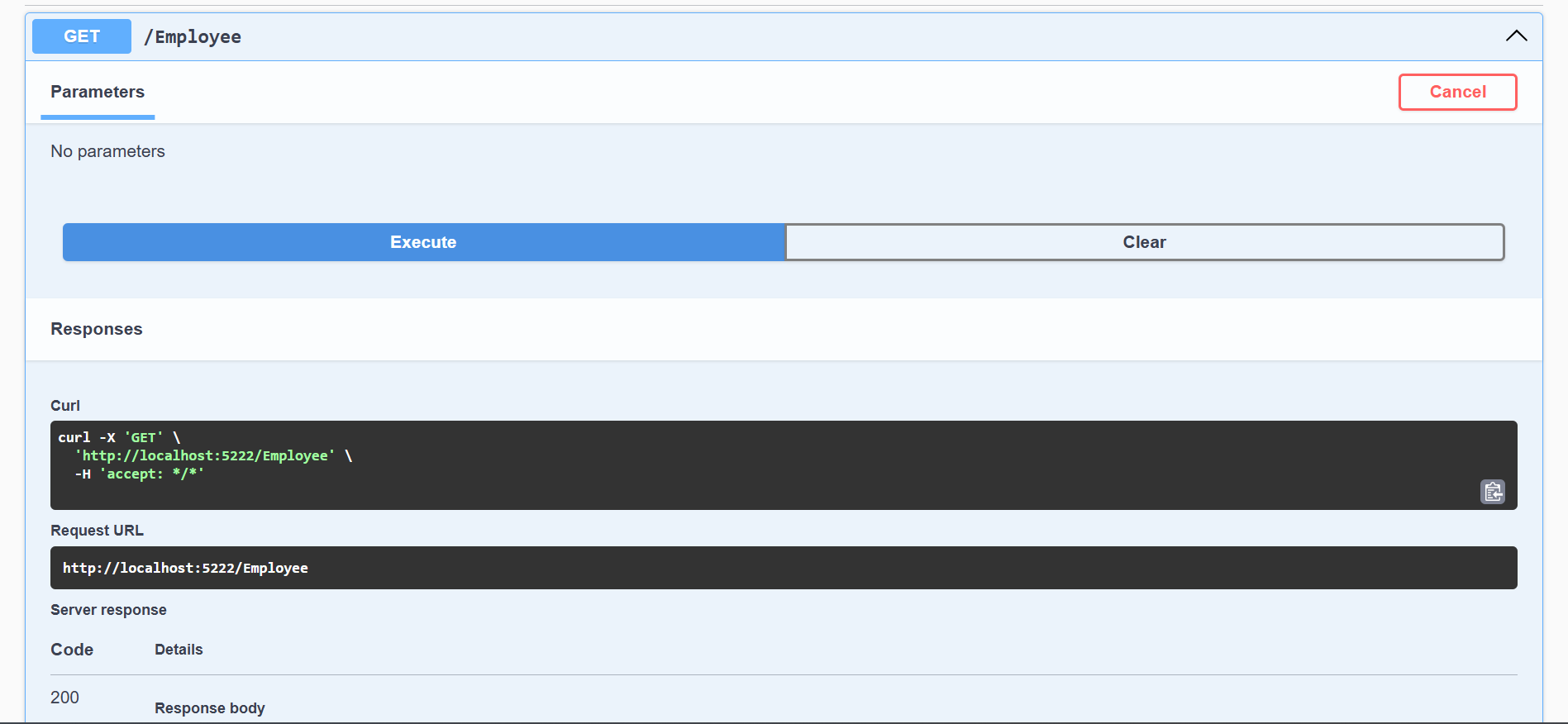
.WithOpenApi();

app.Run();record WeatherForecast(DateOnly Date, int TemperatureC, string? Summary)

{

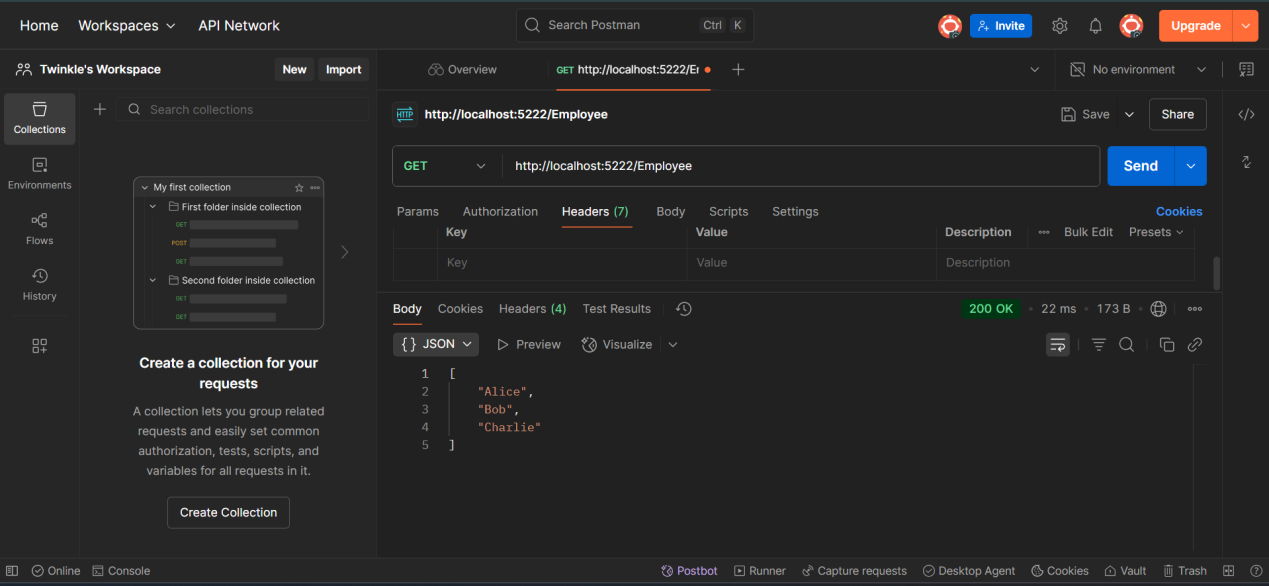
    public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);

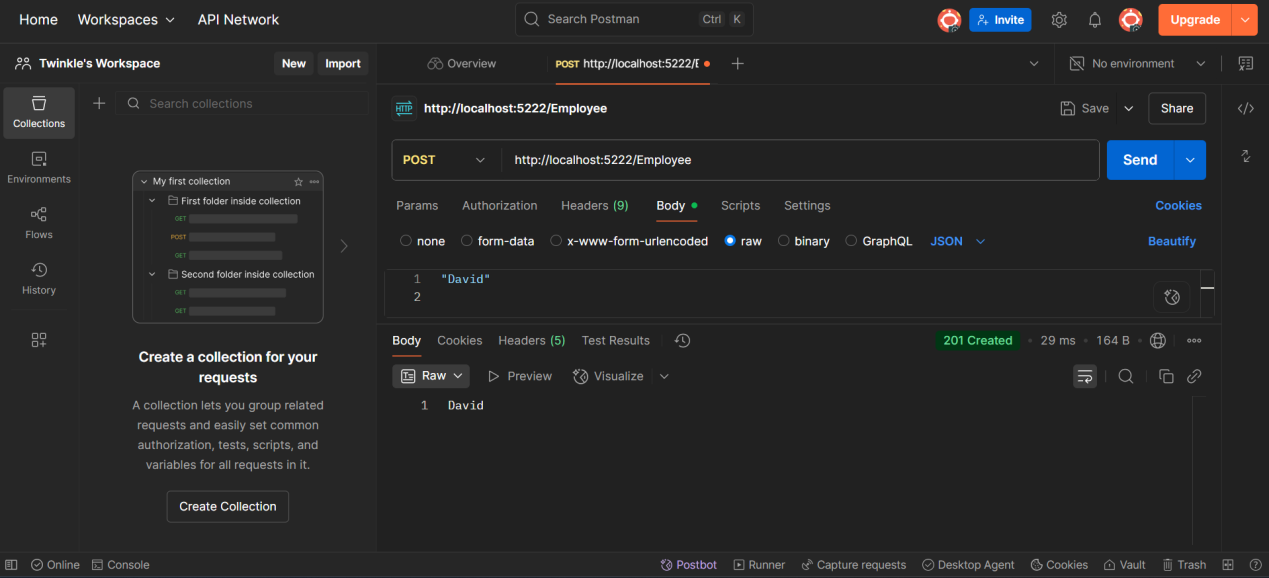
}

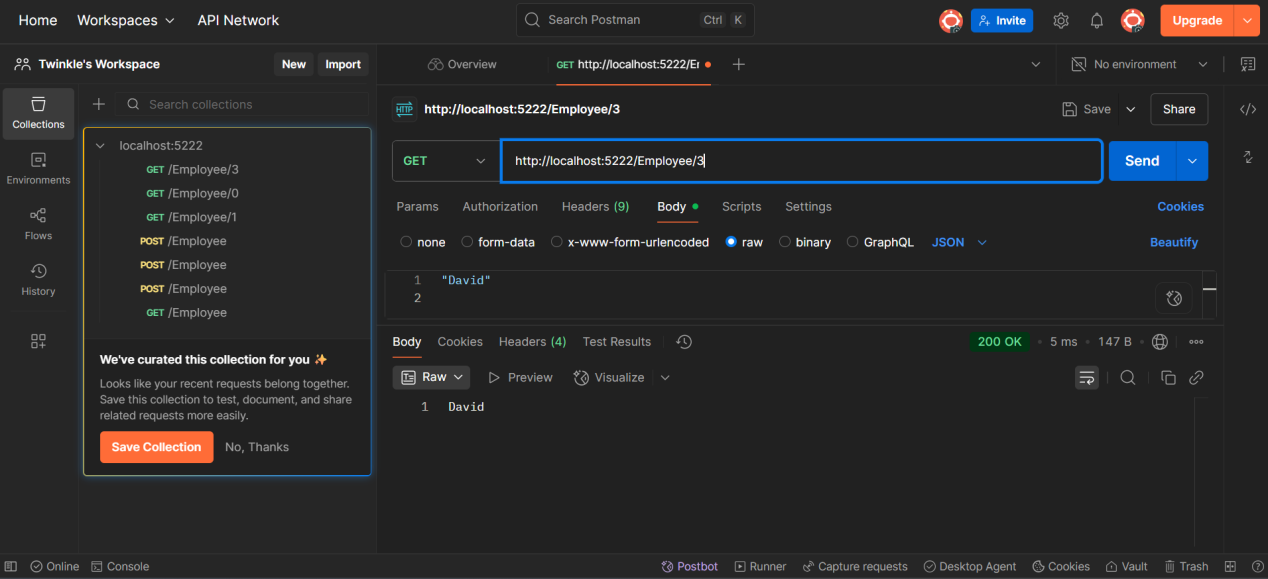




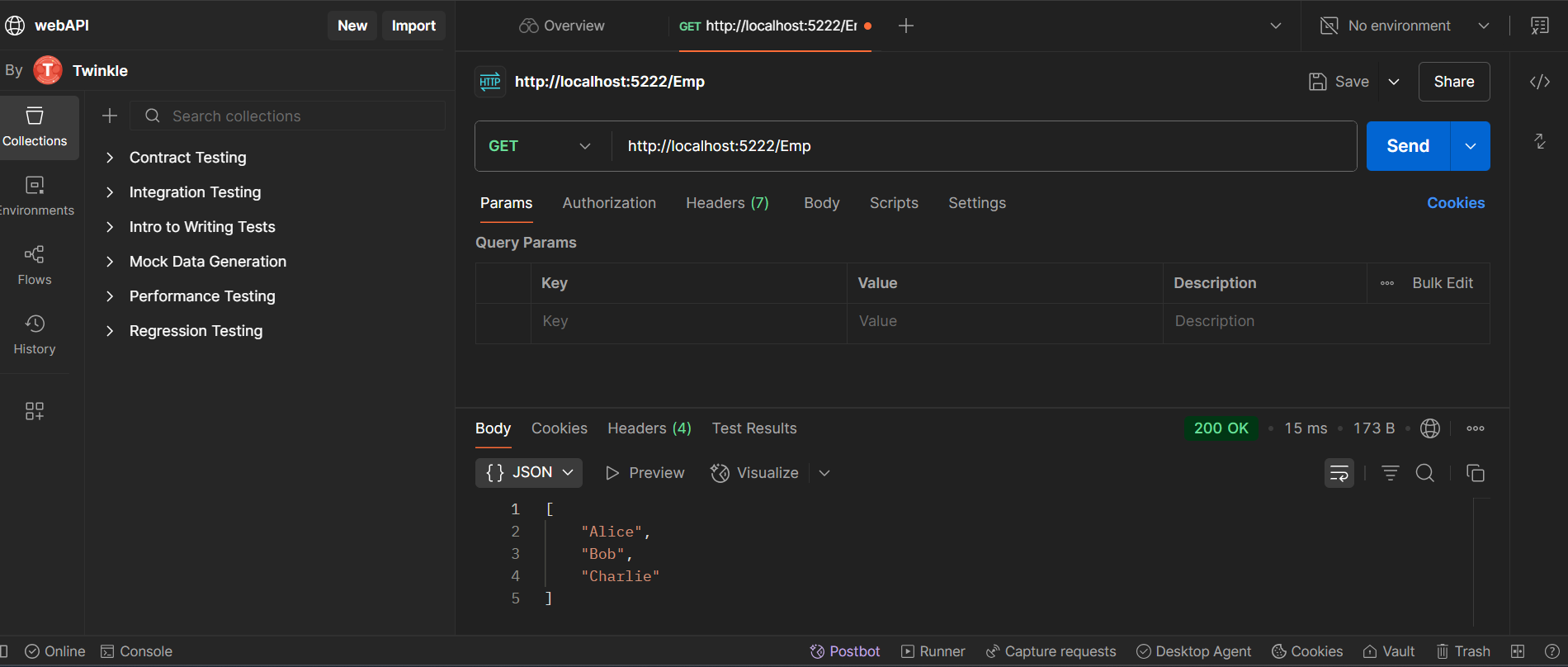
2.Use POSTMAN tool, to point to the local Web API that was created with Employee controller. Test the GET action method using POSTMAN.Verify the output if the List of employees are listed in the ‘Body’ part of the GET window on POSTMAN tool.

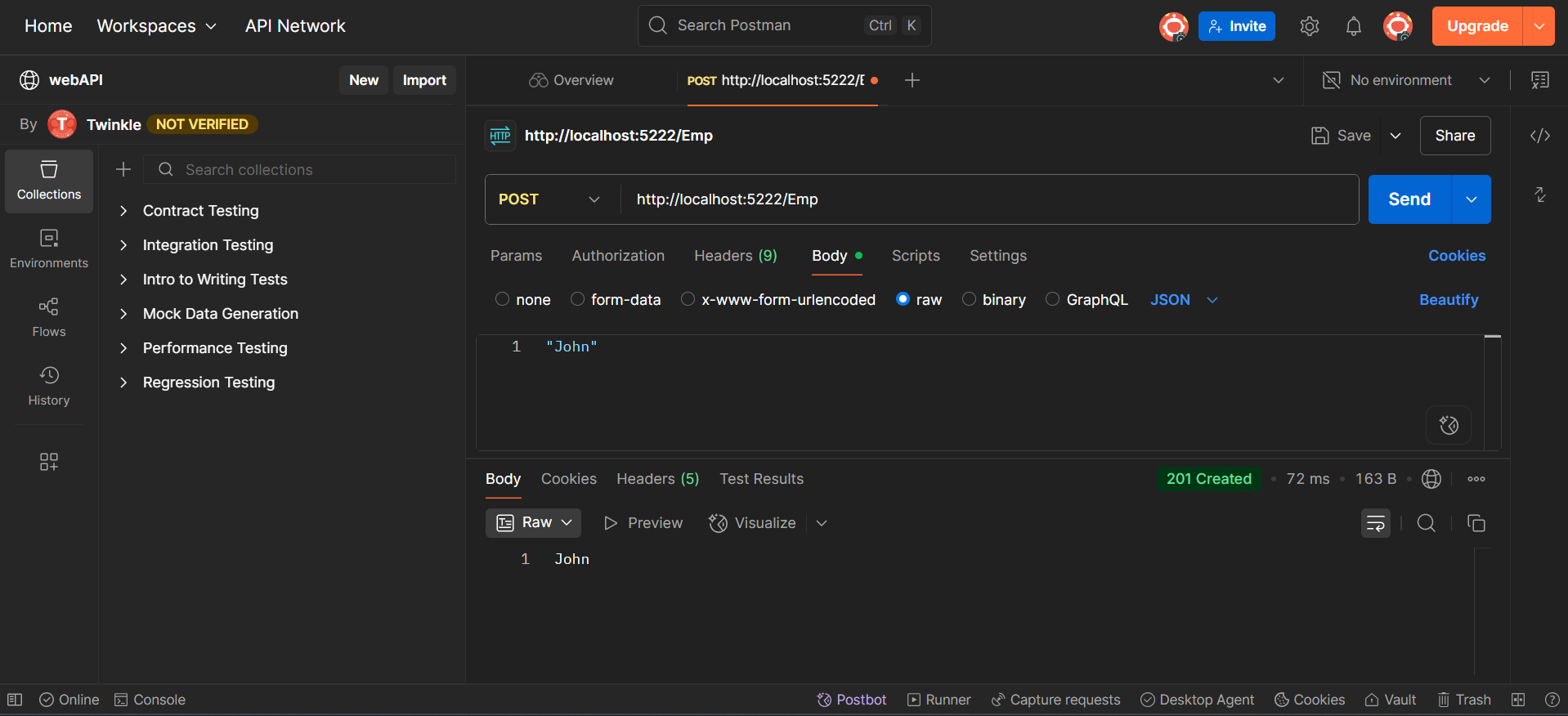


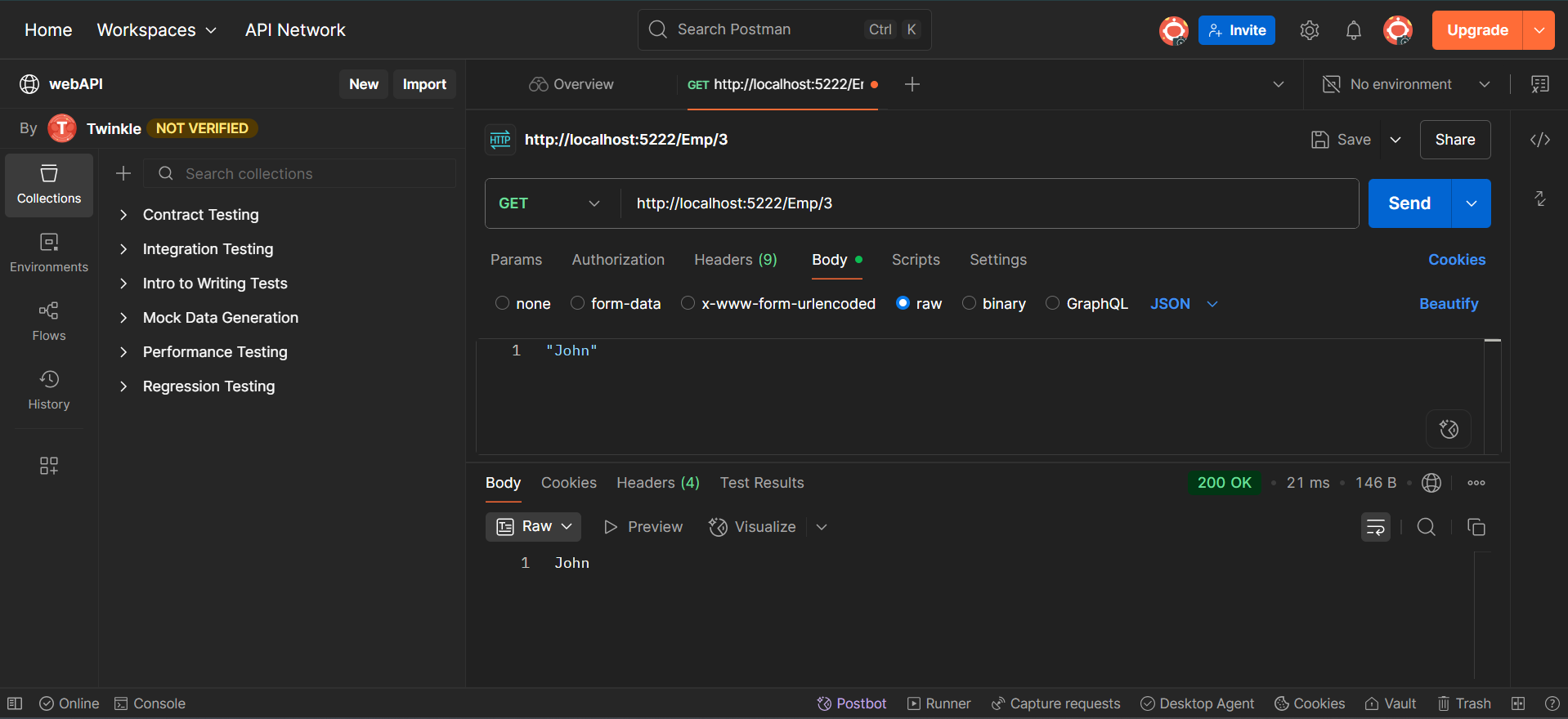




3.Modify the Controller name in the Route attribute of the Employee controller to ‘Emp’ and check its access thru POSTMAN.







**3.Demonstrate creation of an Action method to return list of custom class entity:-**

Employee.cs:

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; }

    }

}

EmployeeController.cs:

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using CustomApiDemo.Models;

using CustomApiDemo.Filters;

namespace CustomApiDemo.Controllers

{

    [ApiController]

    [Route("Emp")]

    [CustomAuthFilter] // Authorization filter applied here

    public class EmployeeController : ControllerBase

    {

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

        [HttpGet]

        [AllowAnonymous]

        [ProducesResponseType(200)]

        [ProducesResponseType(500)]

        public ActionResult<List<Employee>> GetStandard()

        {

            throw new Exception("Testing exception logging...");

            // return Ok(\_employees);

        }

        [HttpPost]

        [ProducesResponseType(201)]

        public IActionResult AddEmployee([FromBody] Employee emp)

        {

            \_employees.Add(emp);

            return Created("", emp);

        }

        private static List<Employee> GetStandardEmployeeList()

        {

            return new List<Employee>

            {

                new Employee

                {

                    Id = 1,

                    Name = "Alice",

                    Salary = 50000,

                    Permanent = true,

                    DateOfBirth = new DateTime(1990, 1, 1),

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

                    Skills = new List<Skill>

                    {

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

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

                    }

                }

            };

        }

    }

}

Department.cs:

namespace CustomApiDemo.Models

{

    public class Department

    {

        public int Id { get; set; }

        public string Name { get; set; }

    }

}

Skill.cs:

namespace CustomApiDemo.Models

{

    public class Skill

    {

        public int Id { get; set; }

        public string Name { get; set; }

    }

}

CustomAuthFilter.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");

                return;

            }

        }

    }

}

CustomExceptionFilter.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace CustomApiDemo.Filters

{

    public class CustomExceptionFilter : IExceptionFilter

    {

        public void OnException(ExceptionContext context)

        {

            string path = "error-log.txt";

            File.AppendAllText(path, $"[{DateTime.Now}] {context.Exception.Message}\n");

            context.Result = new ObjectResult("Internal Server Error Occurred")

            {

                StatusCode = 500

            };

        }

    }

}

Program.cs:

using CustomApiDemo.Filters;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers(options =>

{

    options.Filters.Add<CustomExceptionFilter>(); // Registers global exception handler

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

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

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

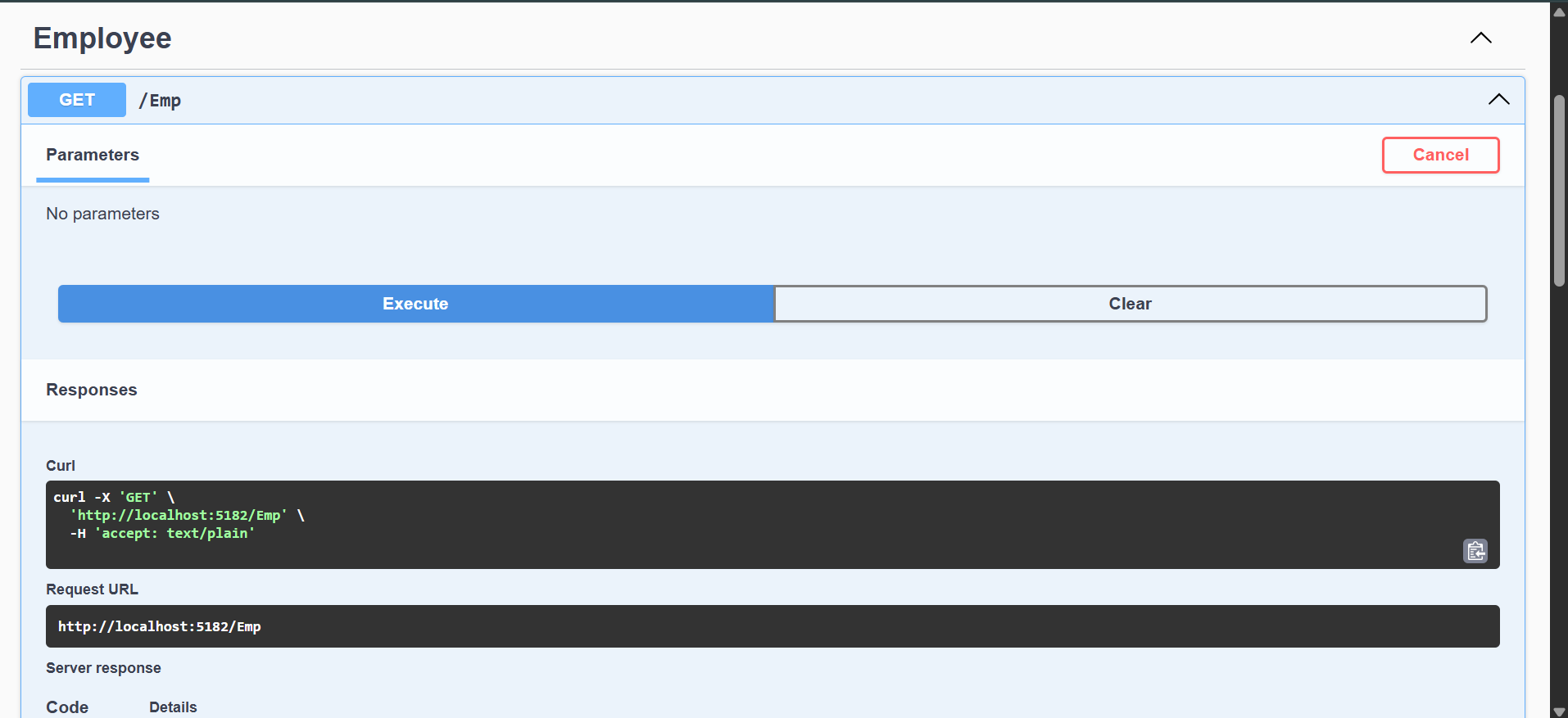
}

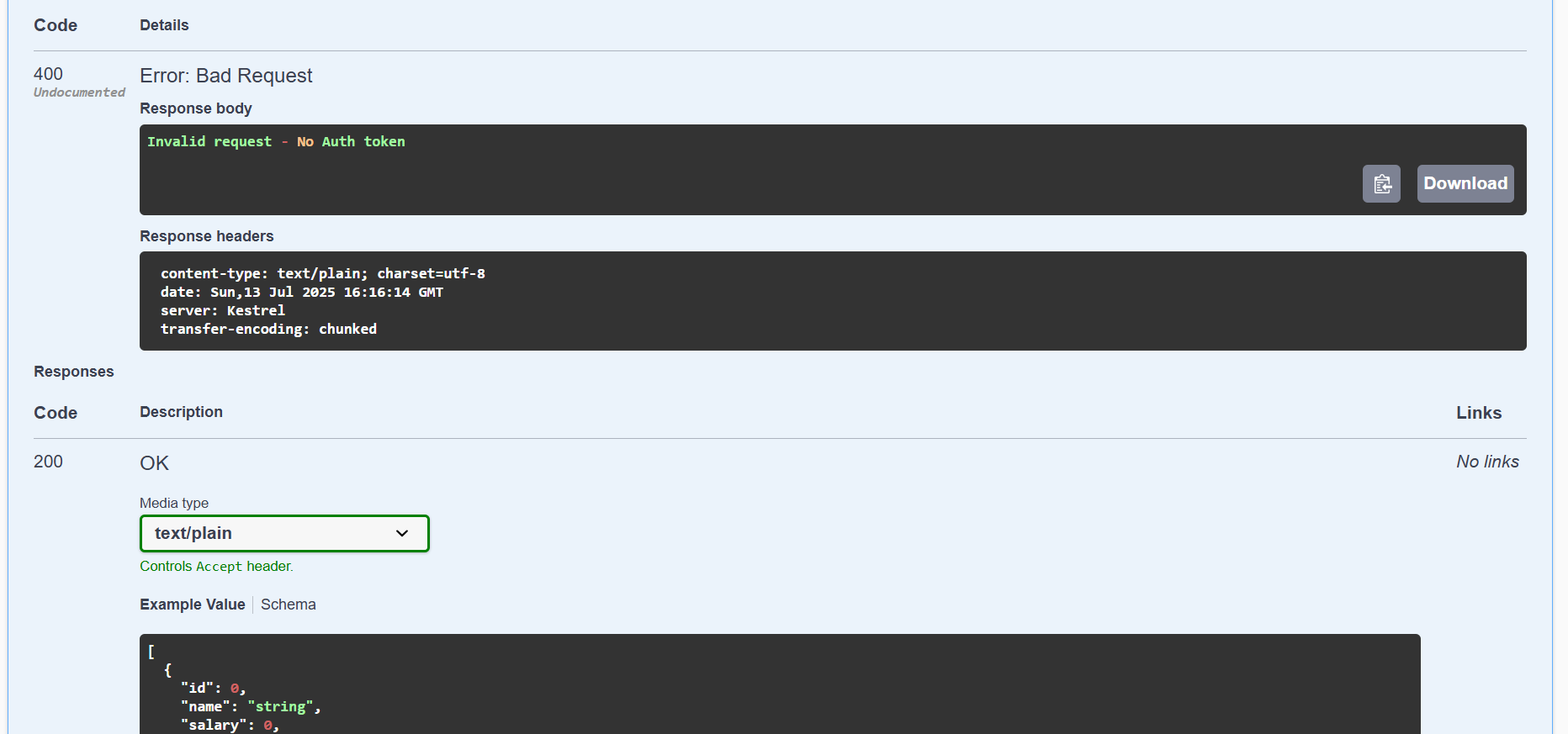
app.UseHttpsRedirection();

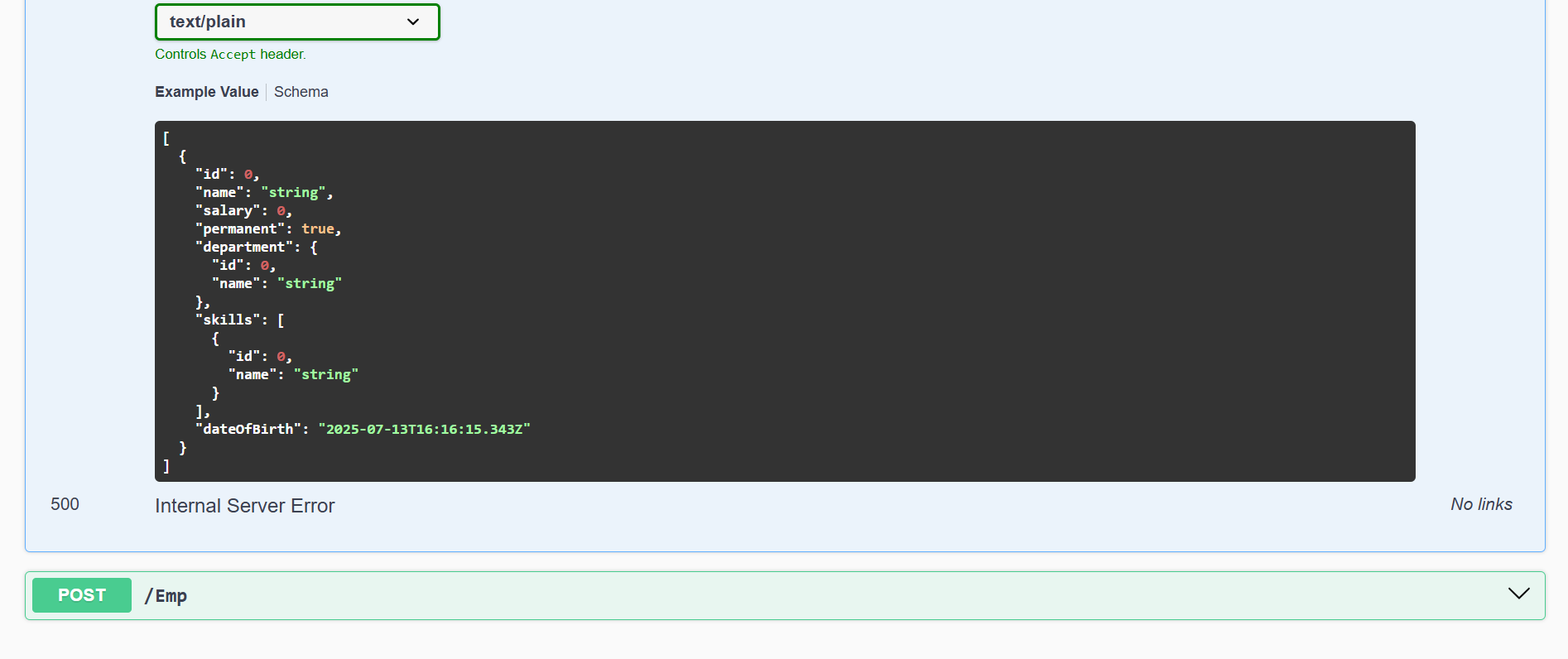
app.MapControllers();

app.Run();

OUTPUT:-







**4.Demonstrate creation of an Action method to perform data create, update & delete operation:-**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using CustomApiDemo.Models;

using CustomApiDemo.Filters;

namespace CustomApiDemo.Controllers

{

    [ApiController]

    [Route("Emp")]

    [CustomAuthFilter] // Authorization filter applied here

    public class EmployeeController : ControllerBase

    {

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

        [HttpGet]

        [AllowAnonymous]

        [ProducesResponseType(200)]

        [ProducesResponseType(500)]

        public ActionResult<List<Employee>> GetStandard()

        {

            throw new Exception("Testing exception logging...");

            // return Ok(\_employees);

        }

        [HttpPost]

        [ProducesResponseType(201)]

        public IActionResult AddEmployee([FromBody] Employee emp)

        {

            \_employees.Add(emp);

            return Created("", emp);

        }

        [HttpPut]

[ProducesResponseType(200)]

[ProducesResponseType(400)]

public ActionResult<Employee> UpdateEmployee([FromBody] Employee updatedEmp)

{

    if (updatedEmp.Id <= 0)

        return BadRequest("Invalid employee id");

    var existingEmp = \_employees.FirstOrDefault(e => e.Id == updatedEmp.Id);

    if (existingEmp == null)

        return BadRequest("Invalid employee id");

    existingEmp.Name = updatedEmp.Name;

    existingEmp.Salary = updatedEmp.Salary;

    existingEmp.Permanent = updatedEmp.Permanent;

    existingEmp.Department = updatedEmp.Department;

    existingEmp.Skills = updatedEmp.Skills;

    existingEmp.DateOfBirth = updatedEmp.DateOfBirth;

    return Ok(existingEmp);

}

        private static List<Employee> GetStandardEmployeeList()

        {

            return new List<Employee>

            {

                new Employee

                {

                    Id = 1,

                    Name = "Alice",

                    Salary = 50000,

                    Permanent = true,

                    DateOfBirth = new DateTime(1990, 1, 1),

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

                    Skills = new List<Skill>

                    {

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

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

                    }

                }

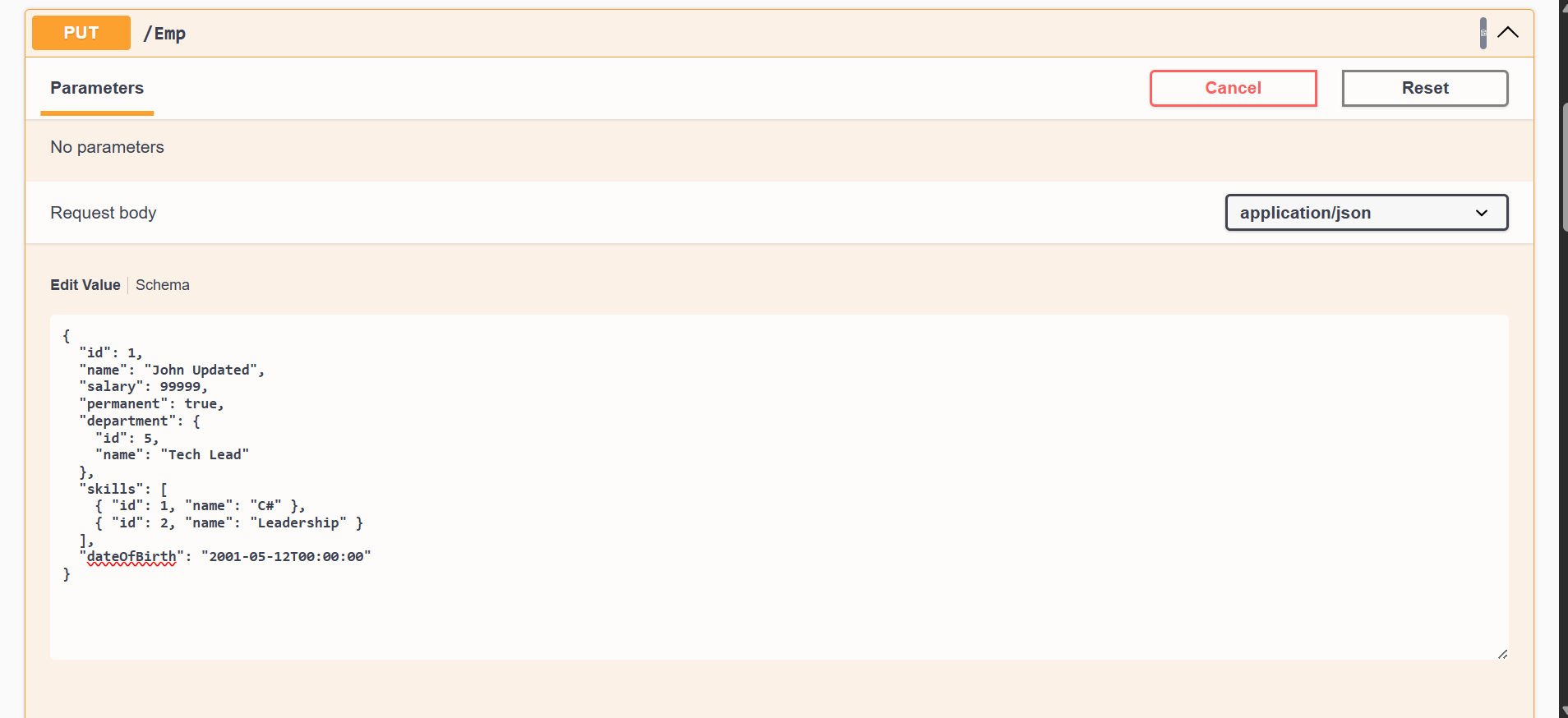
            };

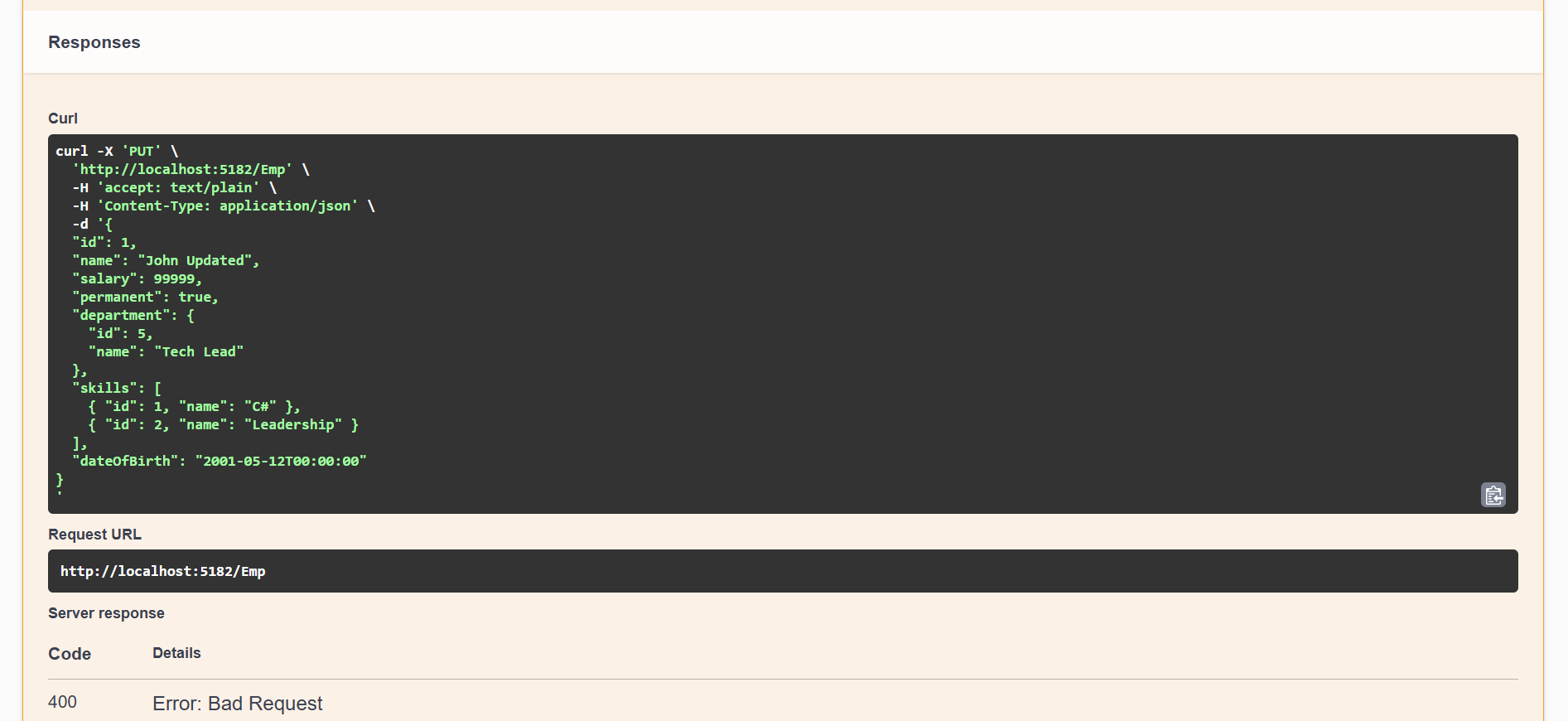
        }

    }

}

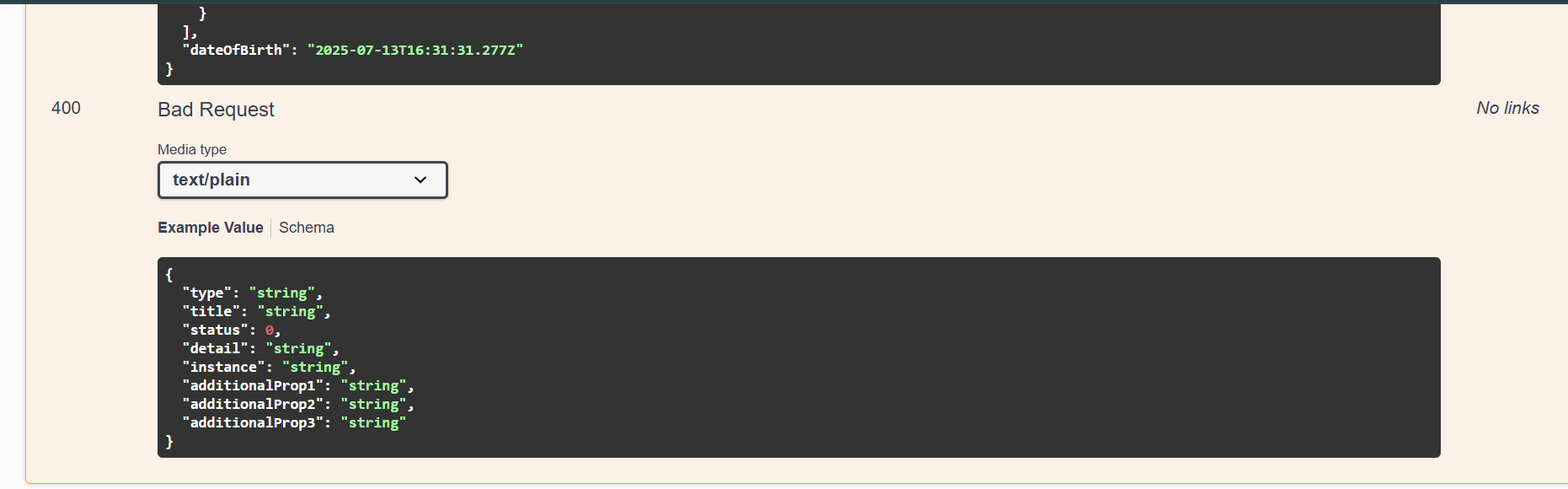
OUTPUT:-











**5.Demonstrate security in WebAPI:-**

AuthController.cs:

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace CustomApiDemo.Controllers

{

    [AllowAnonymous]

    [ApiController]

    [Route("Auth")]

    public class AuthController : ControllerBase

    {

        [HttpGet("token")]

        public IActionResult GetToken()

        {

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

            return Ok(token);

        }

        private string GenerateJSONWebToken(int userId, string userRole)

        {

            var key = "mysuperdupersecure\_secret\_key\_for\_jwt\_123456"; // 256-bit key

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

            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.UtcNow.AddMinutes(10),

                signingCredentials: credentials

            );

            return new JwtSecurityTokenHandler().WriteToken(token);

        }

    }

}

Program.cs:

using CustomApiDemo.Filters;

using Microsoft.OpenApi.Models;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Register global exception filter

builder.Services.AddControllers(options =>

{

    options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

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

});

builder.Services.AddAuthentication(options =>

{

    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

    options.TokenValidationParameters = new TokenValidationParameters

    {

        ValidateIssuer = true,

        ValidateAudience = true,

        ValidateLifetime = true,

        ValidateIssuerSigningKey = true,

        ValidIssuer = "mySystem",

        ValidAudience = "myUsers",

        IssuerSigningKey = new SymmetricSecurityKey(

            Encoding.UTF8.GetBytes("mysuperdupersecure\_secret\_key\_for\_jwt\_123456")),

    };

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

}

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

OUTPUT:

