**EX-1**

using Microsoft.AspNetCore.Mvc;

namespace FirstWebApi.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class ValuesController : ControllerBase

    {

        // GET: api/values

        [HttpGet]

        public IActionResult Get()

        {

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

        }

        // GET: api/values/5

        [HttpGet("{id}")]

        public IActionResult Get(int id)

        {

            return Ok($"value {id}");

        }

        // POST: api/values

        [HttpPost]

        public IActionResult Post([FromBody] string value)

        {

            return Ok($"Posted value: {value}");

        }

        // PUT: api/values/5

        [HttpPut("{id}")]

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

        {

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

        }

        // DELETE: api/values/5

        [HttpDelete("{id}")]

        public IActionResult Delete(int id)

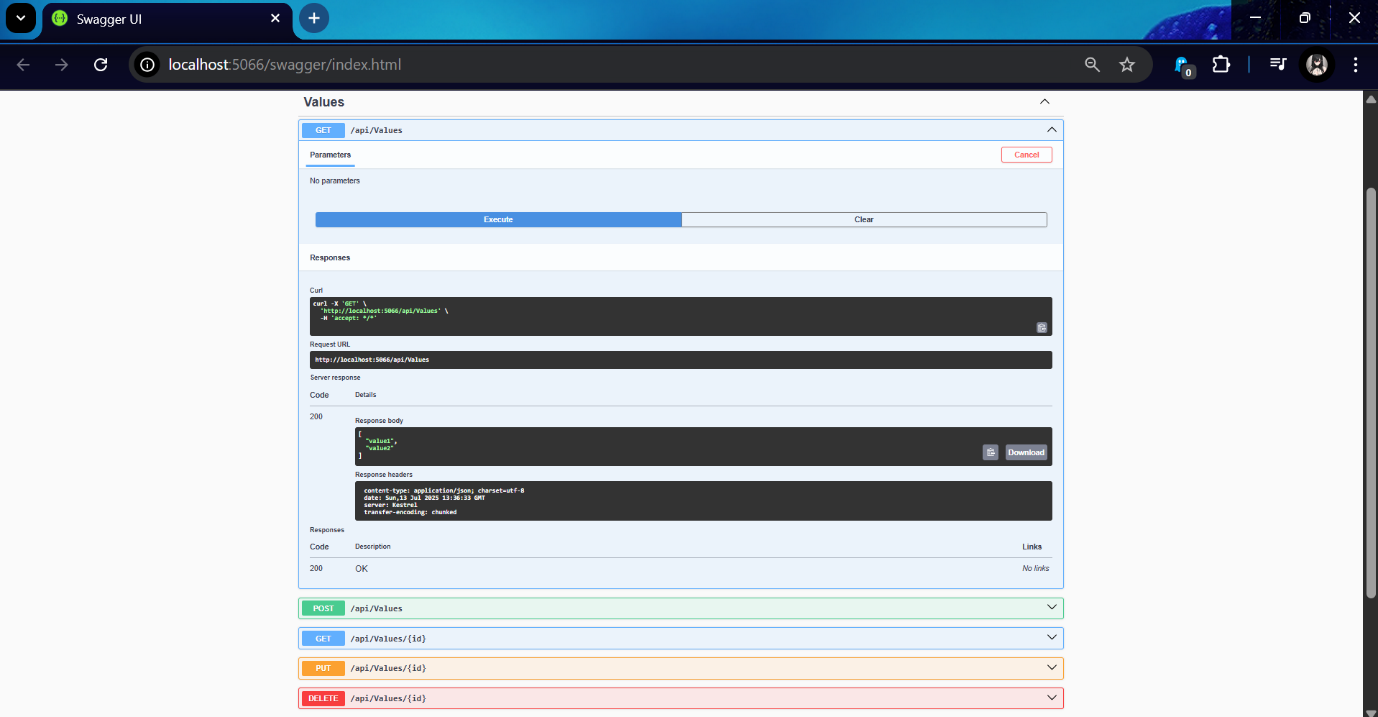
        {

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

        }

    }

}



**EX2**

using Microsoft.AspNetCore.Mvc;

namespace SwaggerDemoAPI.Controllers

{

    [Route("api/emp")]

    [ApiController]

    public class EmployeeController : ControllerBase

    {

        private static List<string> employees = new List<string> { "John", "Jane" };

        [HttpGet(Name = "GetAllEmployees")]

        public ActionResult<IEnumerable<string>> Get()

        {

            return employees;

        }

        [HttpPost]

        public IActionResult Post([FromBody] string name)

        {

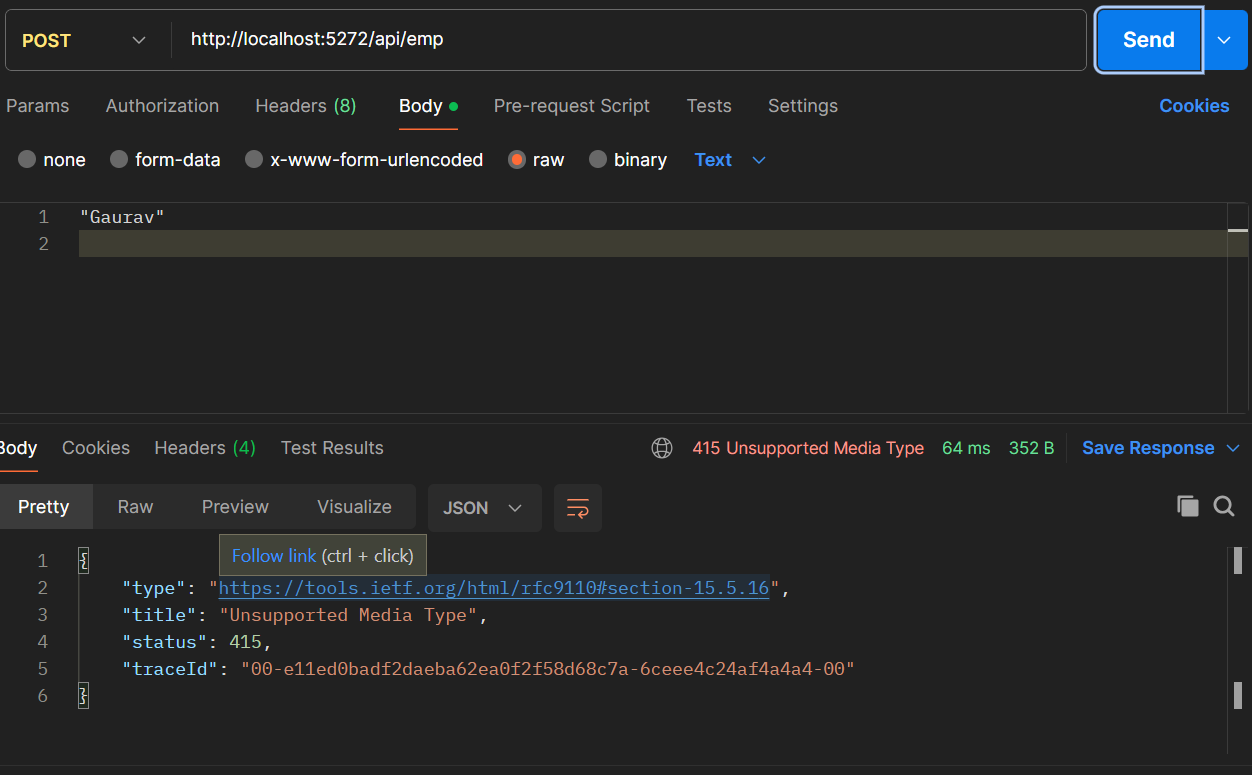
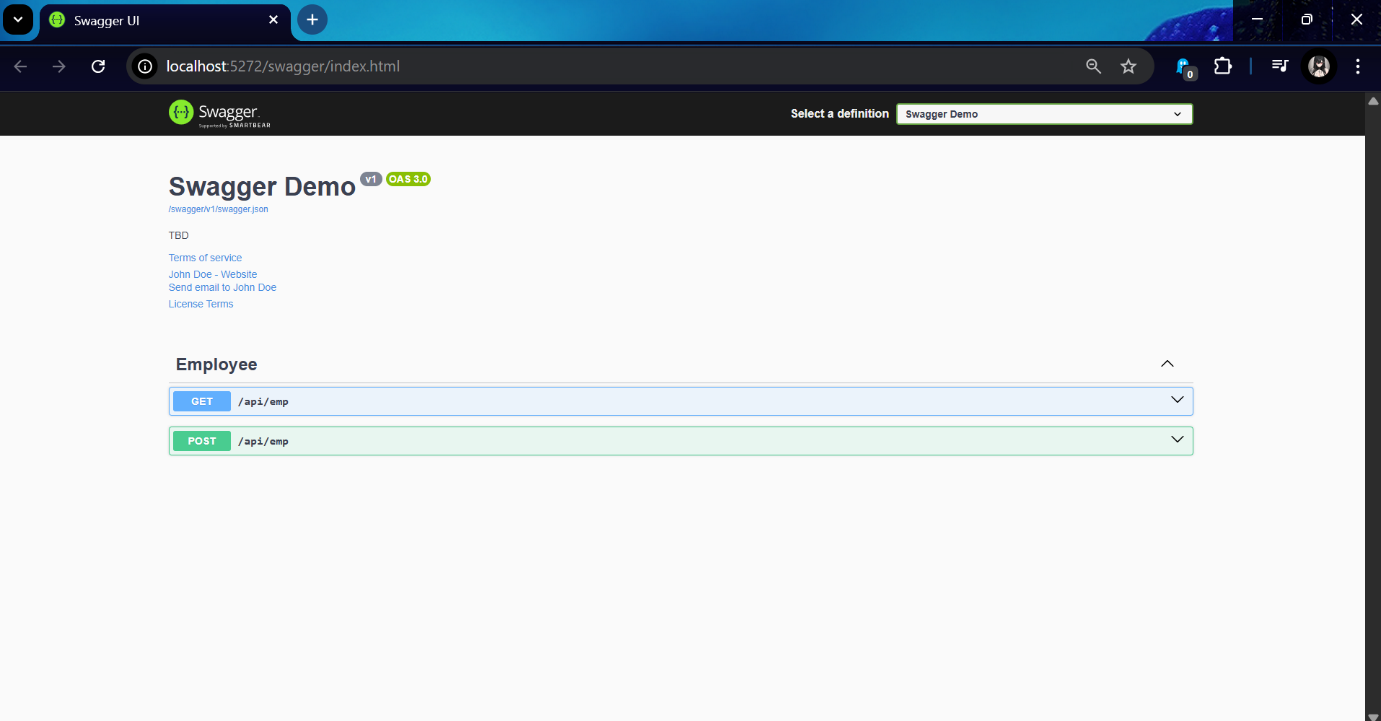
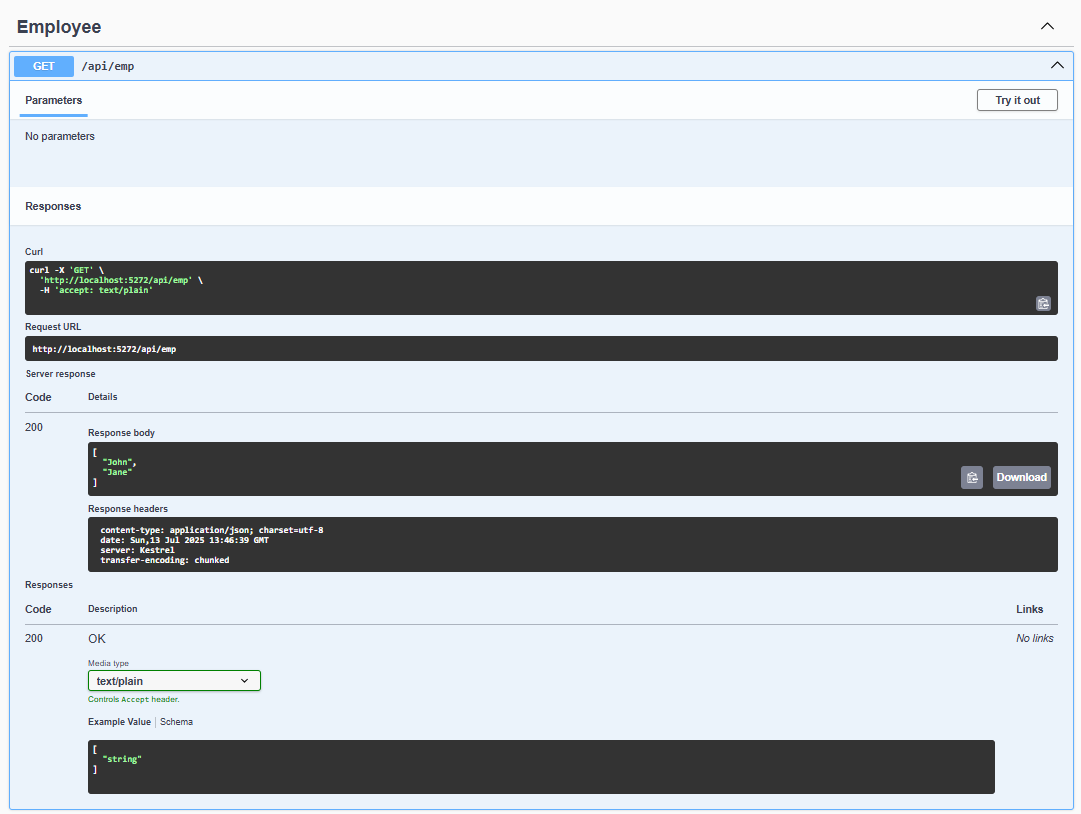
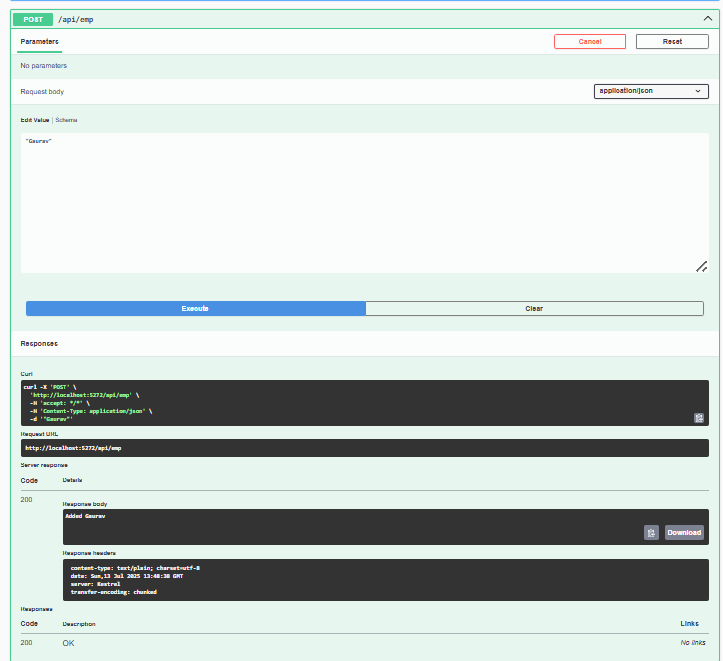
            employees.Add(name);

            return Ok("Added " + name);

        }

    }

}



**EX3**

using EmployeeApiDemo.Filters;

using EmployeeApiDemo.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace EmployeeApiDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [ServiceFilter(typeof(CustomAuthFilter))]

    public class EmployeeController : ControllerBase

    {

        private List<Employee> \_employees;

        public EmployeeController()

        {

            \_employees = GetStandardEmployeeList();

        }

        private List<Employee> GetStandardEmployeeList()

        {

            return new List<Employee>

            {

                new Employee

                {

                    Id = 1,

                    Name = "John",

                    Salary = 50000,

                    Permanent = true,

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

                    Skills = new List<Skill>

                    {

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

                        new Skill { Id = 2, Name = "ASP.NET" }

                    },

                    DateOfBirth = new DateTime(1990, 5, 23)

                }

            };

        }

        [HttpGet("GetStandard")]

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

        [ProducesResponseType(500)]

        [AllowAnonymous]

        public ActionResult<List<Employee>> GetStandard()

        {

            // Uncomment this line to test the exception filter:

            // throw new Exception("Simulated exception for testing");

            return Ok(\_employees);

        }

        [HttpPost]

        public ActionResult<Employee> PostEmployee([FromBody] Employee emp)

        {

            \_employees.Add(emp);

            return Ok(emp);

        }

    }

}

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeApiDemo.Filters

{

    public class CustomAuthFilter : ActionFilterAttribute

    {

        public override void OnActionExecuting(ActionExecutingContext context)

        {

            var headers = context.HttpContext.Request.Headers;

            if (!headers.ContainsKey("Authorization"))

            {

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

                return;

            }

            var token = headers["Authorization"].ToString();

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

            {

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

                return;

            }

            base.OnActionExecuting(context);

        }

    }

}

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeApiDemo.Filters

{

    public class CustomExceptionFilter : IExceptionFilter

    {

        public void OnException(ExceptionContext context)

        {

            string path = Path.Combine(Directory.GetCurrentDirectory(), "errorlog.txt");

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

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

            {

                StatusCode = 500

            };

        }

    }

}

namespace EmployeeApiDemo.Models

{

    public class Department

    {

        public int Id { get; set; }

        public string? Name { get; set; }

    }

}

namespace EmployeeApiDemo.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; }

    }

}

namespace EmployeeApiDemo.Models

{

    public class Skill

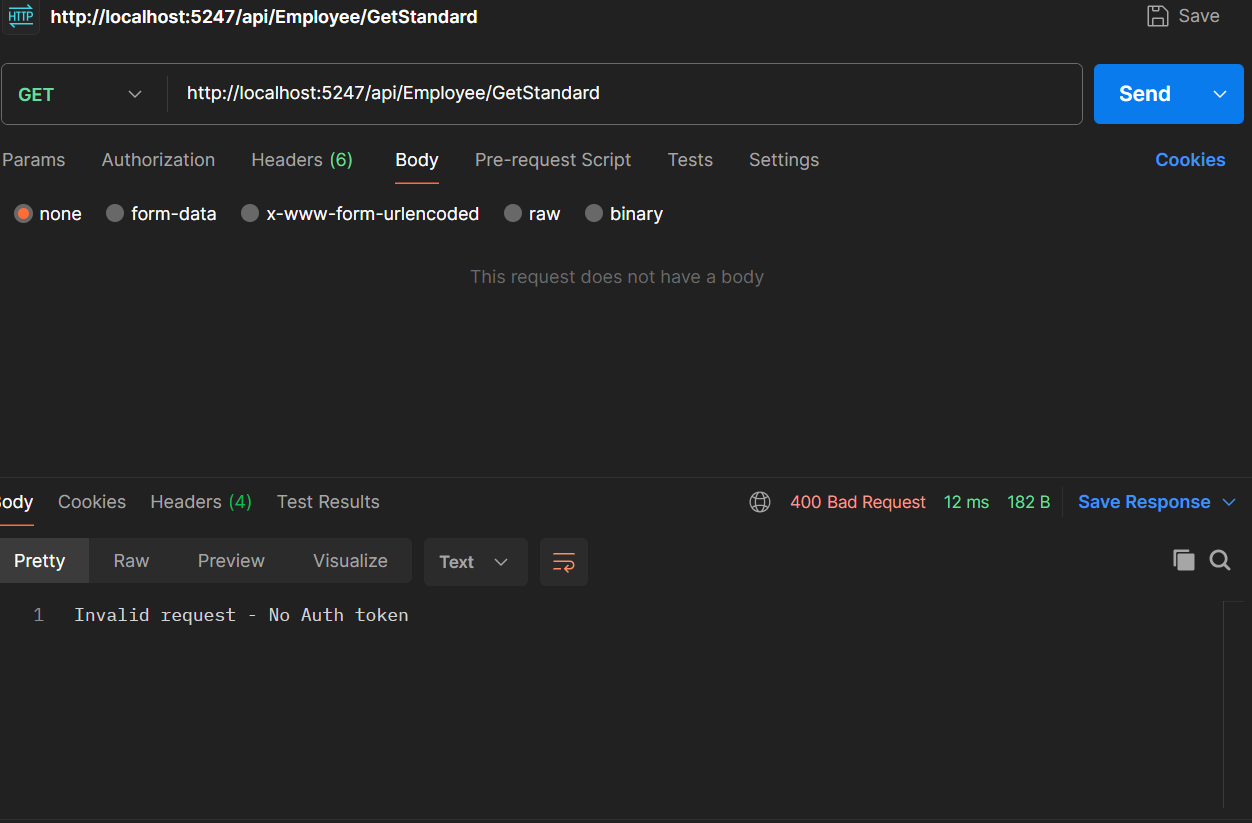
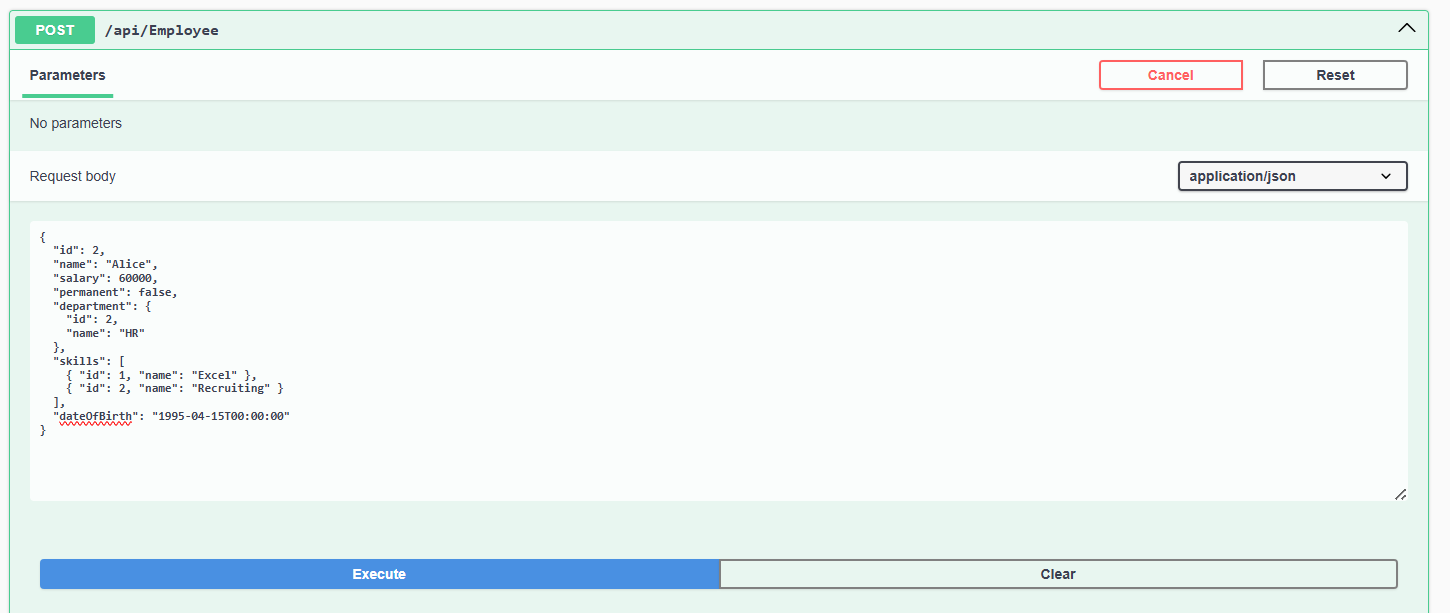
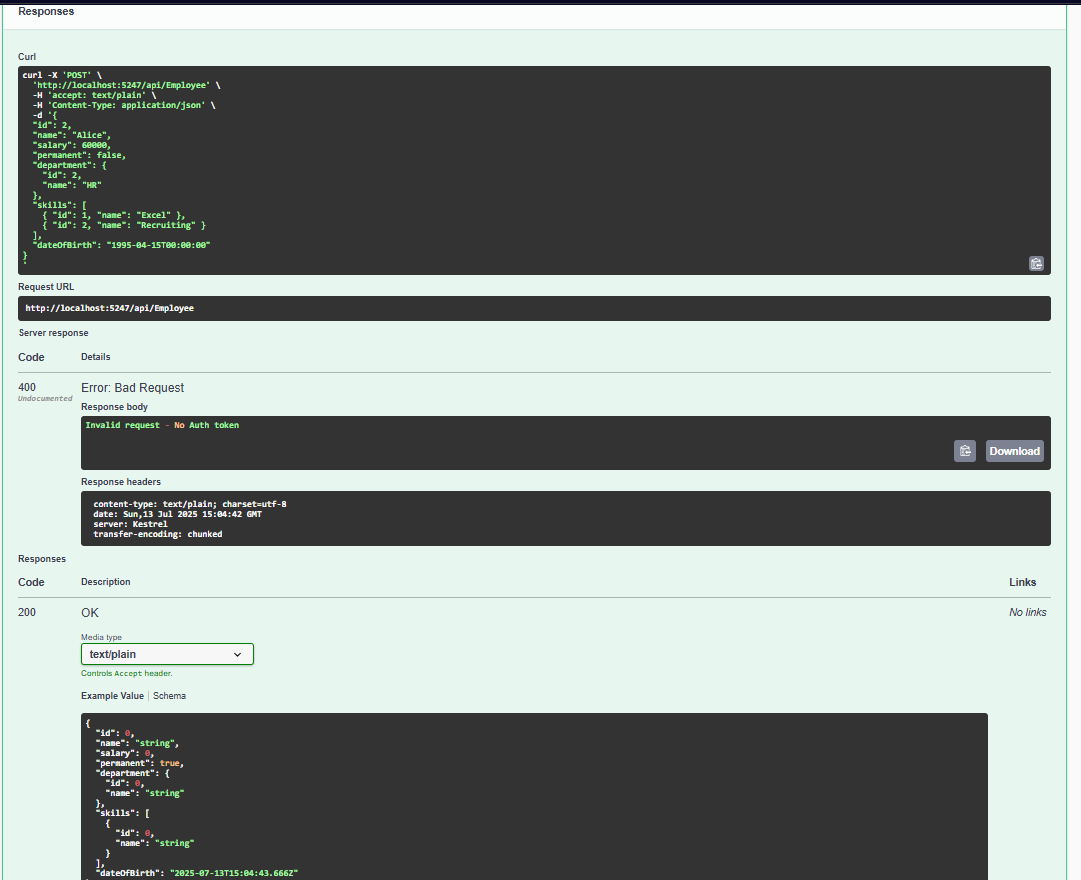
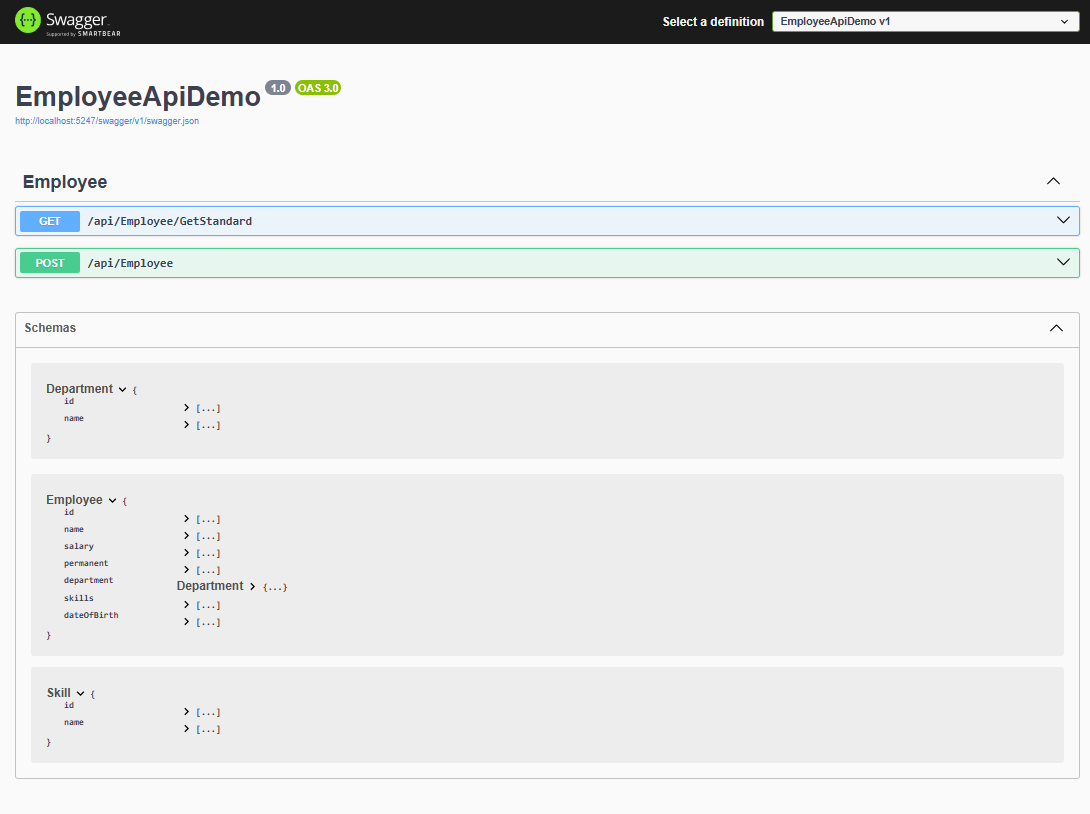
    {

        public int Id { get; set; }

        public string? Name { get; set; }

    }

}



**EX4**

namespace EmployeeCrudApi.Models

{

    public class Skill

    {

        public int Id { get; set; }

        public string Name { get; set; } = string.Empty;

    }

}

using System;

using System.Collections.Generic;

namespace EmployeeCrudApi.Models

{

    public class Employee

    {

        public int Id { get; set; }

        public string Name { get; set; } = string.Empty;

        public int Salary { get; set; }

        public bool Permanent { get; set; }

        public Department Department { get; set; } = new();

        public List<Skill> Skills { get; set; } = new();

        public DateTime DateOfBirth { get; set; }

    }

}

namespace EmployeeCrudApi.Models

{

    public class Department

    {

        public int Id { get; set; }

        public string Name { get; set; } = string.Empty;

    }

}

using Microsoft.AspNetCore.Mvc;

namespace EmployeeCrudApi.Controllers

{

    [ApiController]

    [Route("[controller]")]

    public class WeatherForecastController : ControllerBase

    {

        private static readonly string[] Summaries = new[]

        {

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

        };

        private readonly ILogger<WeatherForecastController> \_logger;

        public WeatherForecastController(ILogger<WeatherForecastController> logger)

        {

            \_logger = logger;

        }

        [HttpGet(Name = "GetWeatherForecast")]

        public IEnumerable<WeatherForecast> Get()

        {

            return Enumerable.Range(1, 5).Select(index => new WeatherForecast

            {

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

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

                Summary = Summaries[Random.Shared.Next(Summaries.Length)]

            })

            .ToArray();

        }

    }

}

using Microsoft.AspNetCore.Mvc;

using EmployeeCrudApi.Models;

namespace EmployeeCrudApi.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class EmployeeController : ControllerBase

    {

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

        {

            new Employee

            {

                Id = 1,

                Name = "Alice",

                Salary = 70000,

                Permanent = true,

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

                Skills = new List<Skill>

                {

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

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

                },

                DateOfBirth = new DateTime(1990, 1, 1)

            }

        };

        // GET all

        [HttpGet]

        public ActionResult<List<Employee>> GetAll()

        {

            return Ok(\_employees);

        }

        // GET by id

        [HttpGet("{id}")]

        public ActionResult<Employee> GetById(int id)

        {

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

            if (emp == null)

                return NotFound();

            return Ok(emp);

        }

        // POST - Create

        [HttpPost]

        public ActionResult<Employee> Create([FromBody] Employee emp)

        {

            if (emp.Id <= 0)

                return BadRequest("Invalid employee id");

            if (\_employees.Any(e => e.Id == emp.Id))

                return BadRequest("Employee already exists");

            \_employees.Add(emp);

            return Ok(emp);

        }

        // PUT - Update

        [HttpPut("{id}")]

        public ActionResult<Employee> Update(int id, [FromBody] Employee updatedEmp)

        {

            if (id <= 0)

                return BadRequest("Invalid employee id");

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

            if (existing == null)

                return BadRequest("Invalid employee id");

            // Update values

            existing.Name = updatedEmp.Name;

            existing.Salary = updatedEmp.Salary;

            existing.Permanent = updatedEmp.Permanent;

            existing.Department = updatedEmp.Department;

            existing.Skills = updatedEmp.Skills;

            existing.DateOfBirth = updatedEmp.DateOfBirth;

            return Ok(existing);

        }

        // DELETE

        [HttpDelete("{id}")]

        public ActionResult Delete(int id)

        {

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

            if (emp == null)

                return NotFound("Employee not found");

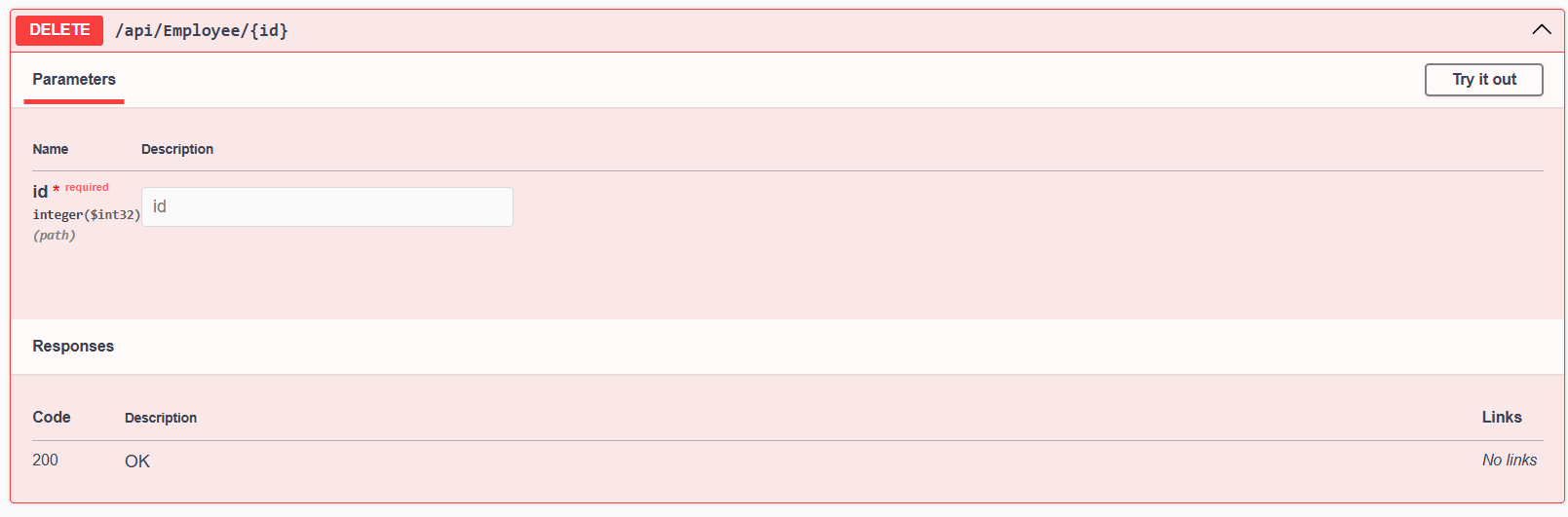
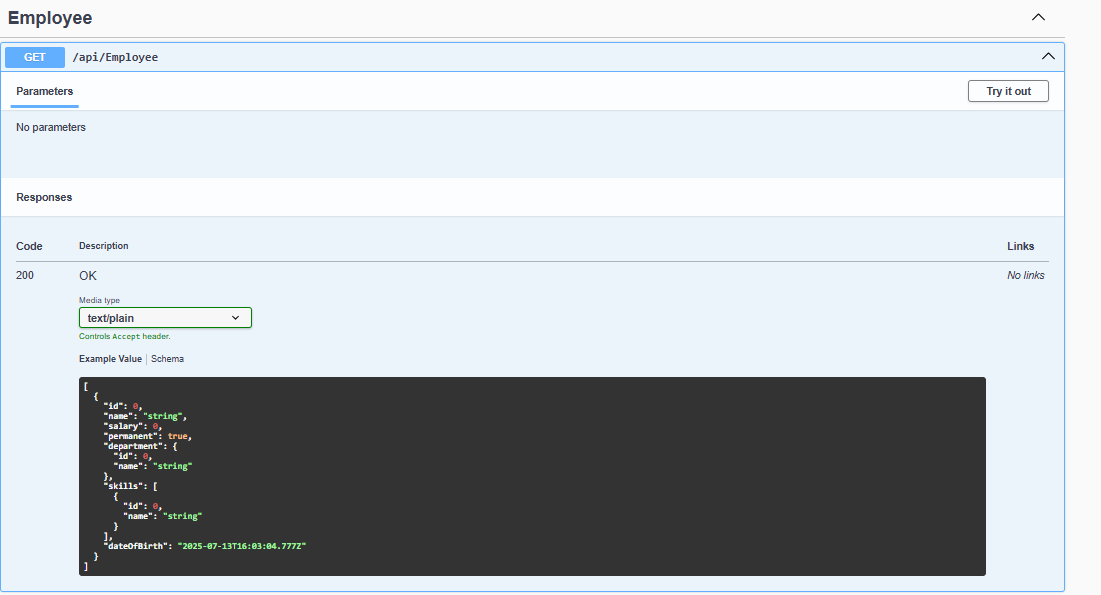
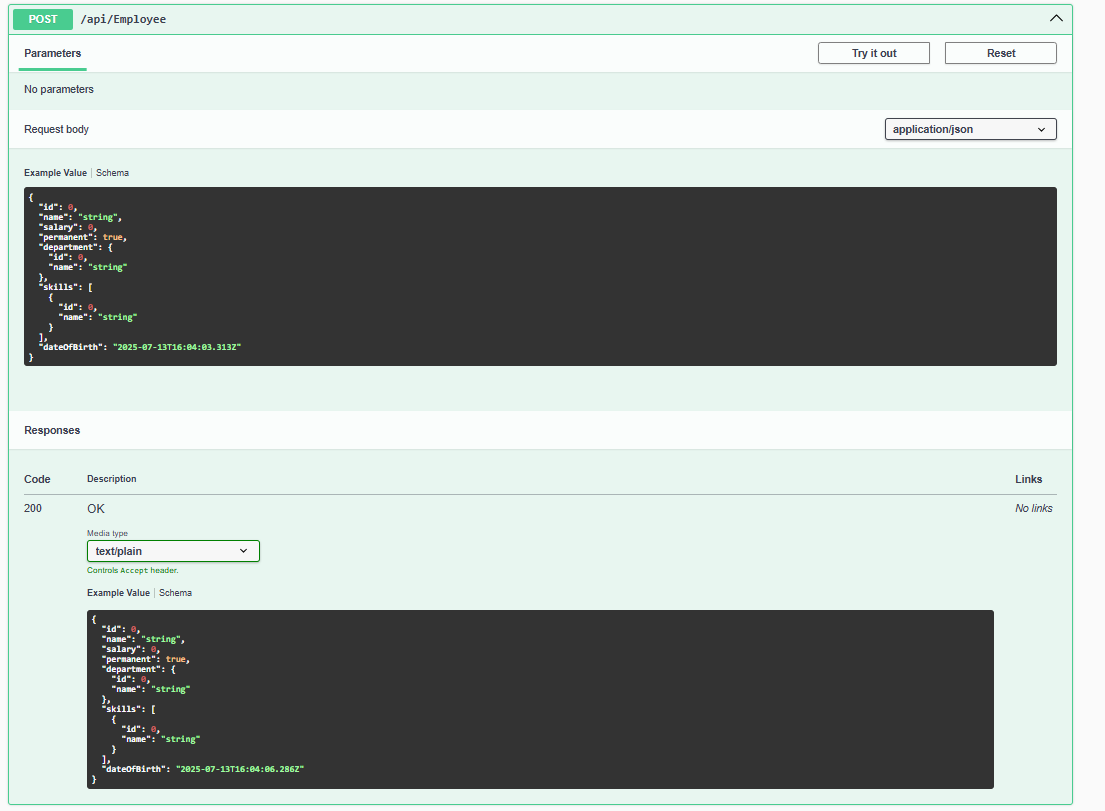
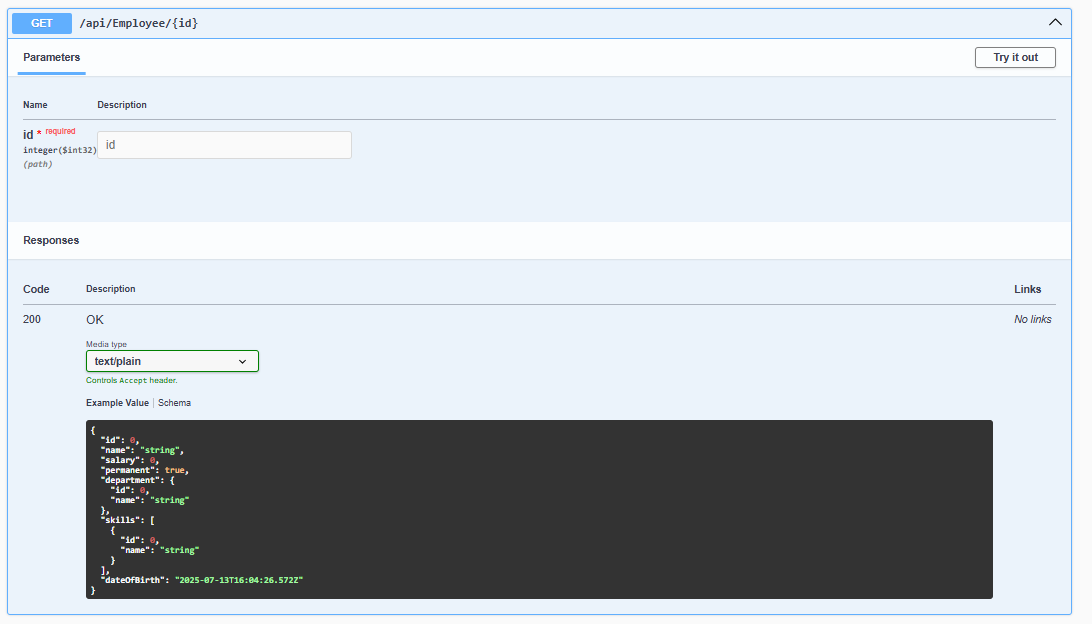
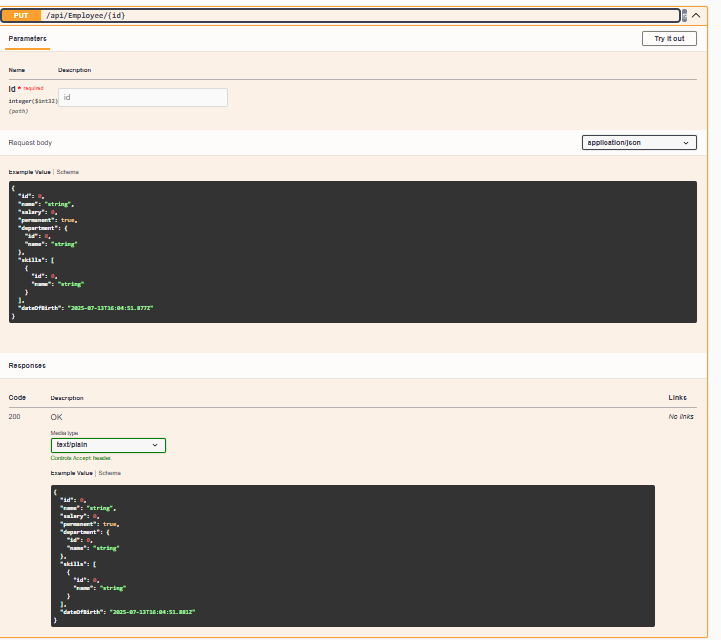
            \_employees.Remove(emp);

            return Ok($"Employee with ID {id} deleted.");

        }

    }

}



**EX-5**

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 EmployeeApiJwtDemo.Controllers

{

    [Route("api/[controller]")]

    [ApiController]

    [AllowAnonymous]

    public class AuthController : ControllerBase

    {

        private string GenerateJSONWebToken(int userId, string userRole)

        {

            var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeythatistotallysecure!123"));

            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),

                signingCredentials: credentials

            );

            return new JwtSecurityTokenHandler().WriteToken(token);

        }

        [HttpGet("token")]

        public IActionResult GetToken()

        {

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

            return Ok(new { token });

        }

    }

}

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace EmployeeApiJwtDemo.Controllers

{

    [Route("api/[controller]")]

    [ApiController]

    [Authorize(Roles = "Admin,POC")]

    public class EmployeeController : ControllerBase

    {

        [HttpGet]

        public IActionResult GetEmployees()

        {

            return Ok("Authorized GET response from EmployeeController");

        }

    }

}

using Microsoft.AspNetCore.Mvc;

namespace EmployeeAuthApi.Controllers

{

    [ApiController]

    [Route("[controller]")]

    public class WeatherForecastController : ControllerBase

    {

        private static readonly string[] Summaries = new[]

        {

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

        };

        private readonly ILogger<WeatherForecastController> \_logger;

        public WeatherForecastController(ILogger<WeatherForecastController> logger)

        {

            \_logger = logger;

        }

        [HttpGet(Name = "GetWeatherForecast")]

        public IEnumerable<WeatherForecast> Get()

        {

            return Enumerable.Range(1, 5).Select(index => new WeatherForecast

            {

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

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

                Summary = Summaries[Random.Shared.Next(Summaries.Length)]

            })

            .ToArray();

        }

    }

}

namespace EmployeeAuthApi.Models

{

    public class UserModel

    {

        public int Id { get; set; }

        public string Role { get; set; } = string.Empty;

    }

}

