**1.Web Api using custom model class**

Employee.cs :

using System;

using System.Collections.Generic;

namespace EmpDemo.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; }

    }

*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; }

    }

}

EmployeeController.cs :

using System;

using System.Collections.Generic;

using System.Linq;

using EmpDemo.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace EmpDemo.Controllers

{

    [Authorize(Roles = "Admin,POC")]

    [ApiController]

    [Route("api/[controller]")]

*public* class EmployeeController : ControllerBase

    {

*private* *static* List<Employee> employees;

*static* EmployeeController()

        {

            employees = GetStandardEmployeeList();

        }

        [HttpGet]

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

        [ProducesResponseType(500)]

*public* ActionResult<List<Employee>> Get()

        {

*// throw new Exception("Test exception from GET method");*

            return Ok(employees);

        }

        [HttpGet("standard")]

        [ProducesResponseType(typeof(Employee), 200)]

*public* ActionResult<Employee> GetStandard()

        {

            return Ok(employees.First());

        }

        [HttpPost]

        [ProducesResponseType(typeof(Employee), 201)]

*public* ActionResult<Employee> Post([FromBody] Employee emp)

        {

            emp.Id = employees.Max(e => e.Id) + 1;

            employees.Add(emp);

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

        }

        [HttpPut("{id}")]

        [ProducesResponseType(typeof(Employee), 200)]

        [ProducesResponseType(400)]

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

        {

            if (id <= 0)

            {

                return BadRequest("Invalid employee id");

            }

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

            if (existingEmp == null)

            {

                return BadRequest("Invalid employee id");

            }

            existingEmp.Name = updatedEmp.Name;

            existingEmp.Salary = updatedEmp.Salary;

            existingEmp.Permanent = updatedEmp.Permanent;

            existingEmp.DateOfBirth = updatedEmp.DateOfBirth;

            existingEmp.Department = updatedEmp.Department;

            existingEmp.Skills = updatedEmp.Skills;

            return Ok(existingEmp);

        }

*private* *static* List<Employee> GetStandardEmployeeList()

        {

            return new List<Employee>

            {

                new Employee

                {

                    Id = 1,

                    Name = "John Doe",

                    Salary = 60000,

                    Permanent = true,

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

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

                    Skills = new List<Skill>

                    {

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

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

                    }

                },

                new Employee

                {

                    Id = 2,

                    Name = "Jane Smith",

                    Salary = 70000,

                    Permanent = false,

                    DateOfBirth = new DateTime(1992, 5, 15),

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

                    Skills = new List<Skill>

                    {

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

                        new Skill { Id = 4, Name = "Communication" }

                    }

                }

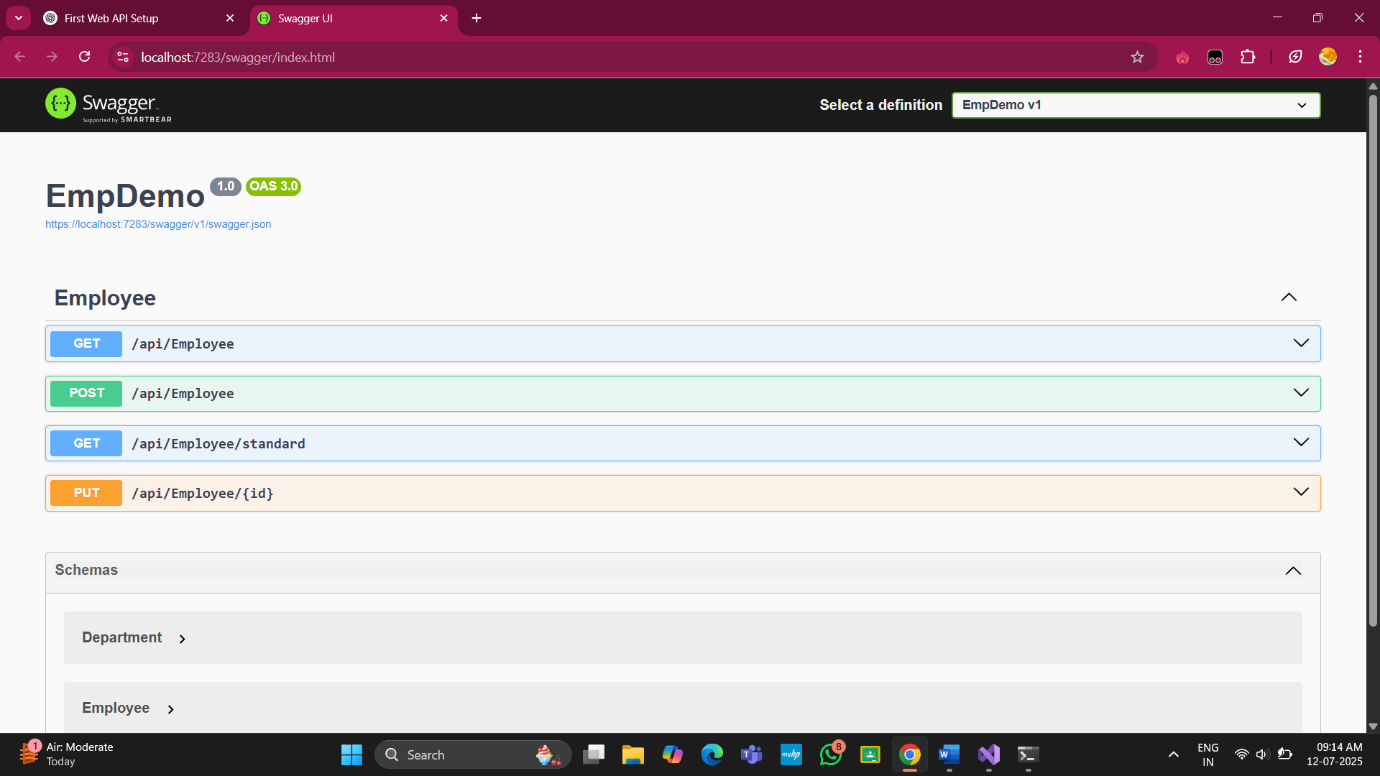
            };

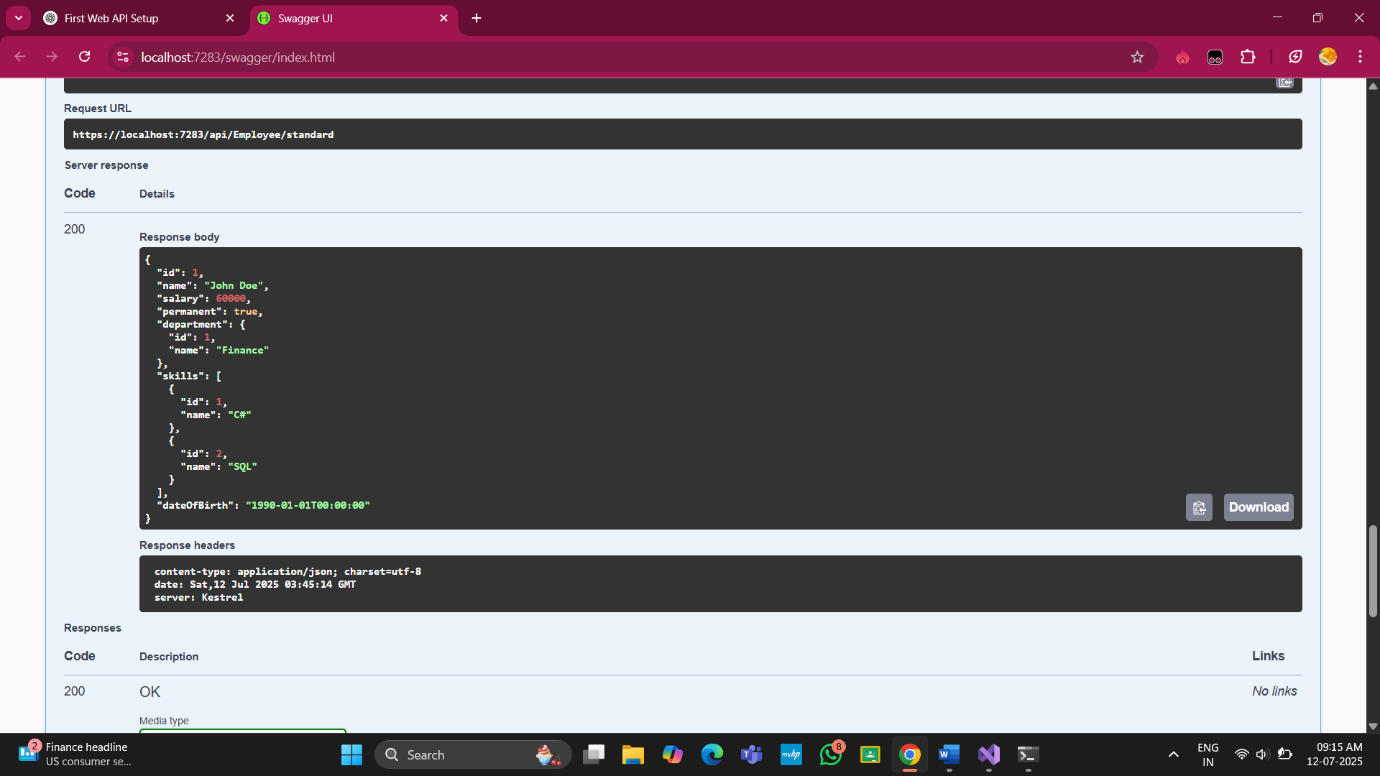
        }

    }

}

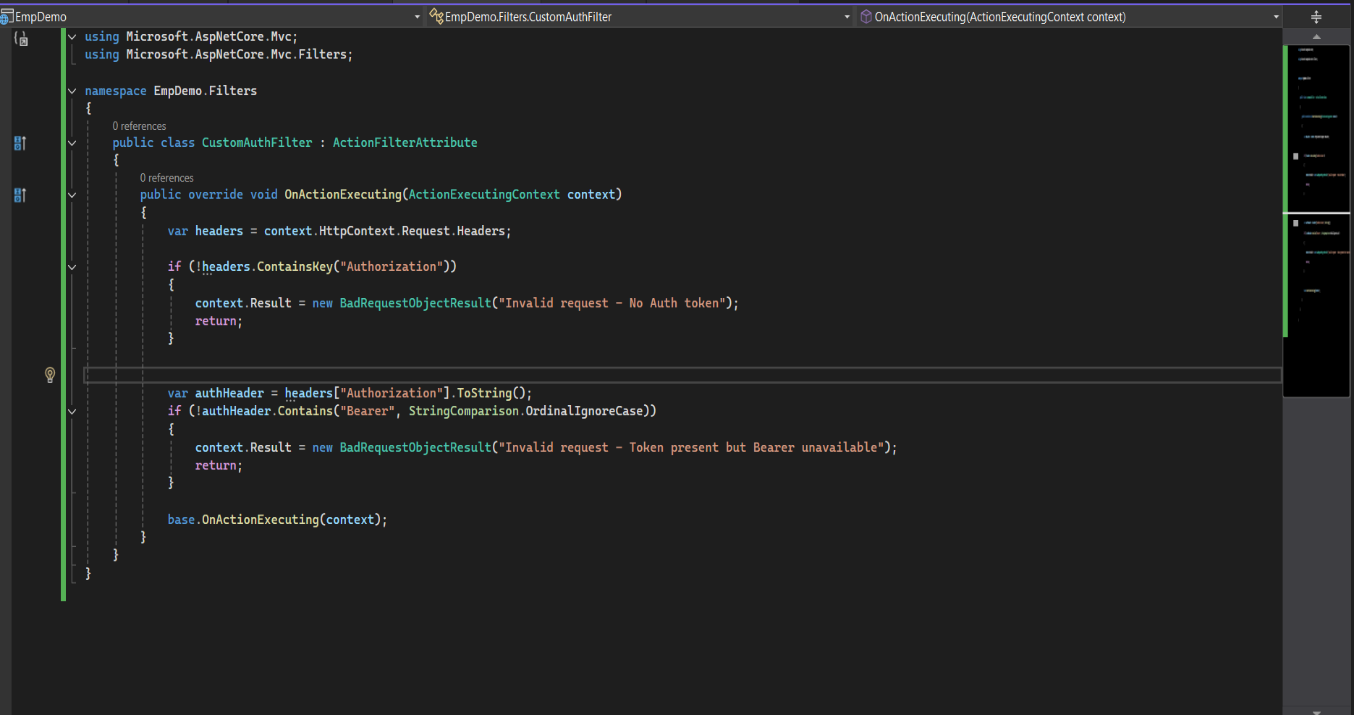
Output:





**2. Create a Custom action filter for Authorization**

CutomAuthFilter.cs :



EmployeeController.cs :

using System;

using System.Collections.Generic;

using System.Linq;

using EmpDemo.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace EmpDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [CustomAuthFilter]

*public* class EmployeeController : ControllerBase

    {

*private* *static* List<Employee> employees;

*static* EmployeeController()

        {

            employees = GetStandardEmployeeList();

        }

        [HttpGet]

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

        [ProducesResponseType(500)]

*public* ActionResult<List<Employee>> Get()

        {

*// throw new Exception("Test exception from GET method");*

            return Ok(employees);

        }

        [HttpGet("standard")]

        [ProducesResponseType(typeof(Employee), 200)]

*public* ActionResult<Employee> GetStandard()

        {

            return Ok(employees.First());

        }

        [HttpPost]

        [ProducesResponseType(typeof(Employee), 201)]

*public* ActionResult<Employee> Post([FromBody] Employee emp)

        {

            emp.Id = employees.Max(e => e.Id) + 1;

            employees.Add(emp);

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

        }

        [HttpPut("{id}")]

        [ProducesResponseType(typeof(Employee), 200)]

        [ProducesResponseType(400)]

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

        {

            if (id <= 0)

            {

                return BadRequest("Invalid employee id");

            }

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

            if (existingEmp == null)

            {

                return BadRequest("Invalid employee id");

            }

            existingEmp.Name = updatedEmp.Name;

            existingEmp.Salary = updatedEmp.Salary;

            existingEmp.Permanent = updatedEmp.Permanent;

            existingEmp.DateOfBirth = updatedEmp.DateOfBirth;

            existingEmp.Department = updatedEmp.Department;

            existingEmp.Skills = updatedEmp.Skills;

            return Ok(existingEmp);

        }

*private* *static* List<Employee> GetStandardEmployeeList()

        {

            return new List<Employee>

            {

                new Employee

                {

                    Id = 1,

                    Name = "John Doe",

                    Salary = 60000,

                    Permanent = true,

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

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

                    Skills = new List<Skill>

                    {

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

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

                    }

                },

                new Employee

                {

                    Id = 2,

                    Name = "Jane Smith",

                    Salary = 70000,

                    Permanent = false,

                    DateOfBirth = new DateTime(1992, 5, 15),

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

                    Skills = new List<Skill>

                    {

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

                        new Skill { Id = 4, Name = "Communication" }

                    }

                }

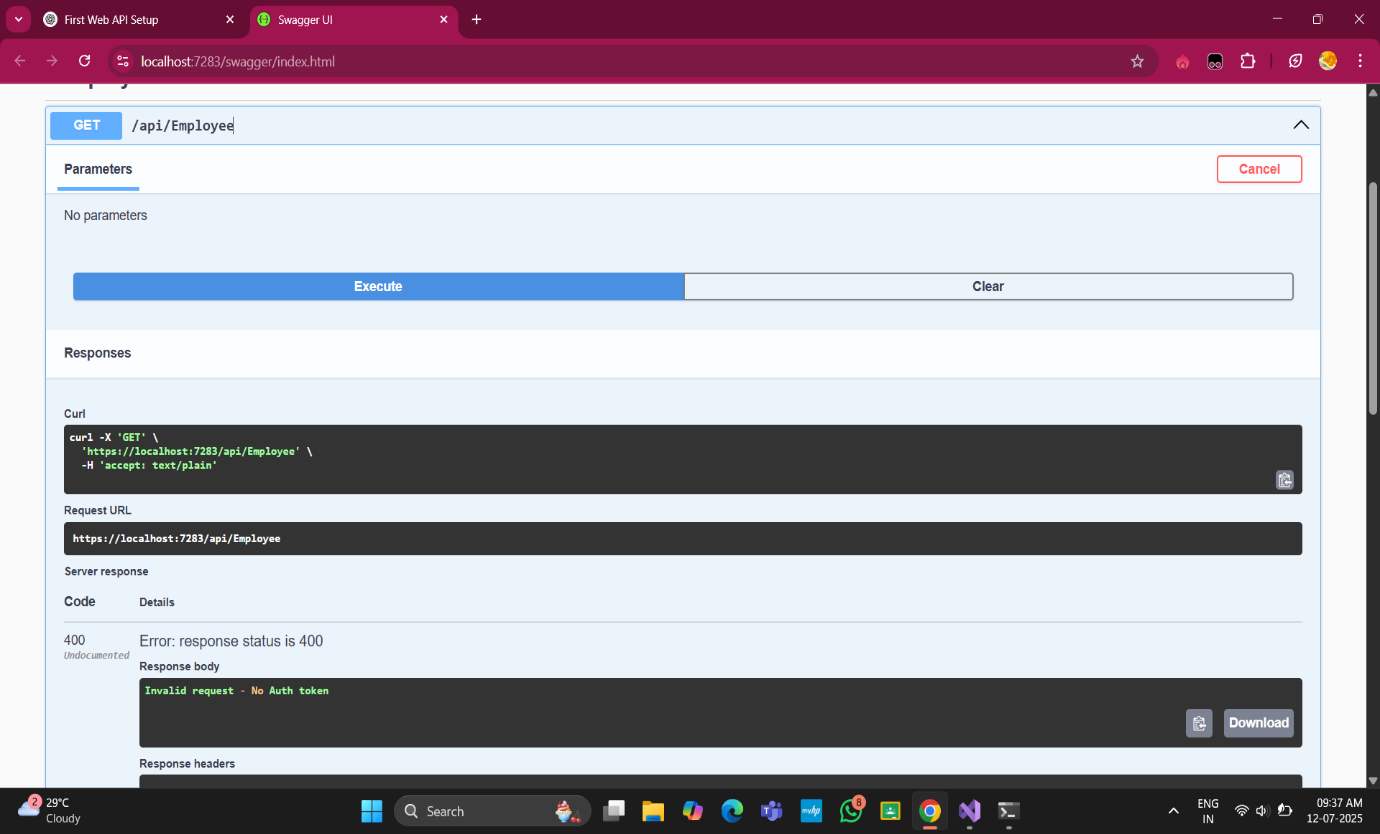
            };

        }

    }

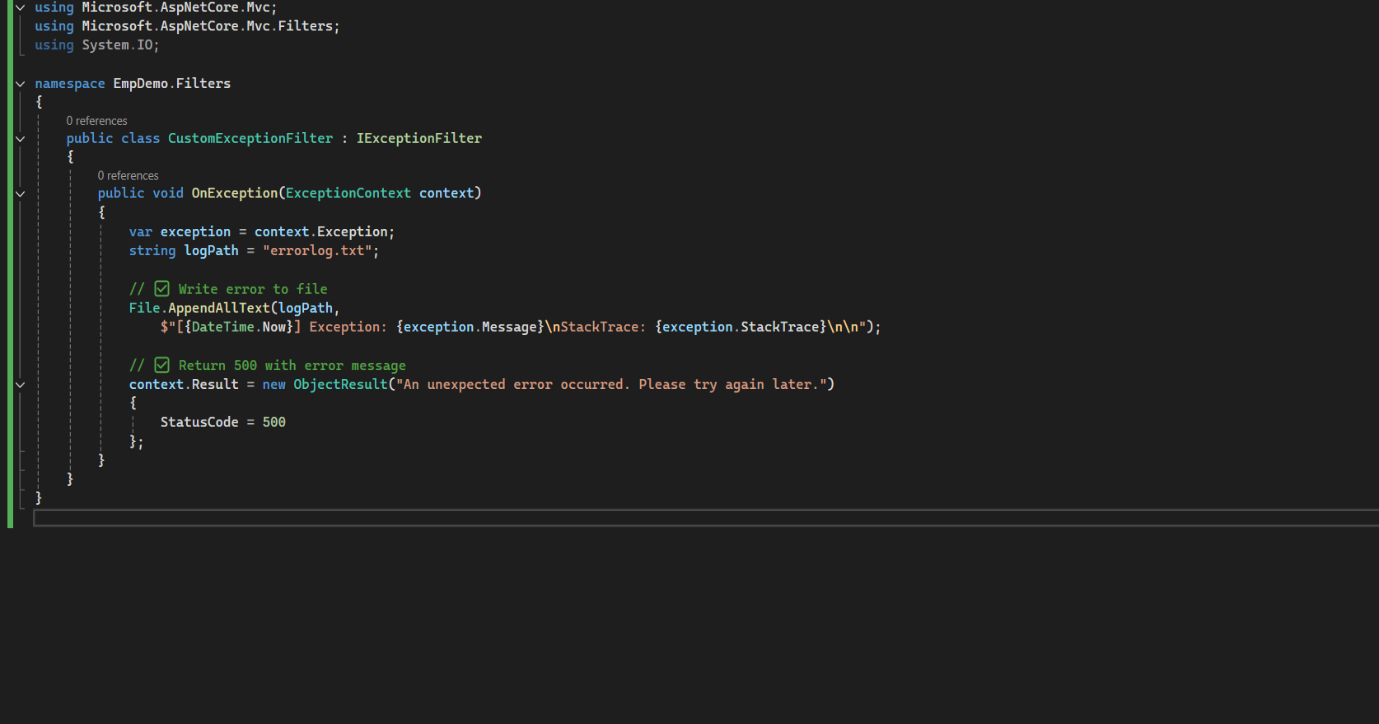
}

Output :



**3.** **Custom Exception filter**

CustomExceptionFilter.cs :



Program.cs :

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers(options =>

{

options.Filters.Add<EmpDemo.Filters.CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

EmployeeController.cs :

using Microsoft.AspNetCore.Mvc;

using EmpDemo.Models;

using System;

using System.Collections.Generic;

using System.Linq;

namespace EmpDemo.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> employees;

static EmployeeController()

{

employees = GetStandardEmployeeList();

}

[HttpGet]

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

[ProducesResponseType(500)]

public ActionResult<List<Employee>> Get()

{

throw new Exception("Test exception from GET method");

}

[HttpGet("standard")]

[ProducesResponseType(typeof(Employee), 200)]

public ActionResult<Employee> GetStandard()

{

return Ok(employees.First());

}

[HttpPost]

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

{

emp.Id = employees.Max(e => e.Id) + 1;

employees.Add(emp);

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

}

[HttpPut("{id}")]

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

{

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

if (emp == null) return NotFound();

emp.Name = updatedEmp.Name;

emp.Salary = updatedEmp.Salary;

emp.Permanent = updatedEmp.Permanent;

emp.DateOfBirth = updatedEmp.DateOfBirth;

emp.Department = updatedEmp.Department;

emp.Skills = updatedEmp.Skills;

return NoContent();

}

private static List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John Doe",

Salary = 60000,

Permanent = true,

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

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

Skills = new List<Skill>

{

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

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

}

},

new Employee

{

Id = 2,

Name = "Jane Smith",

Salary = 70000,

Permanent = false,

DateOfBirth = new DateTime(1992, 5, 15),

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

Skills = new List<Skill>

{

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

new Skill { Id = 4, Name = "Communication" }

}

}

};

}

}

}

Output :

