**4.Web Api CRUD operation**

Update Employee data as per the input thru Web API PUT action method call

Employee information has to be updated based on the user input. Use Swagger tool to invoke the action method mapped with Http PUT action verb to update an employee data.

Modify the action method to return Employee data thru ActionResult.

Check if the id value is lesser than or equal to 0. If true, throw BadRequest action result with the message ‘Invalid employee id’

If the value is greater than 0 but not available in the list of employee ids that is there in the hardcoded list of employees, throw BadRequest action result with the same message as stated above.

If the id value is valid, use the JSON data from the input body and update the hardcoded list. Filter the employee list data for the input id and return that as the output.

Employee.cs:

namespace FirstWebApi.Models

{

    public class Employee

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public string Department { get; set; }

        public double Salary { get; set; }

    }

}

EmployeeController.cs:

using Microsoft.AspNetCore.Mvc;

using FirstWebApi.Models;

using System.Collections.Generic;

using System.Linq;

namespace FirstWebApi.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class EmployeesController : ControllerBase

    {

        // Hardcoded list of employees (static to persist in memory)

        private static List<Employee> employees = new List<Employee>

        {

            new Employee { Id = 1, Name = "Alice", Department = "HR", Salary = 50000 },

            new Employee { Id = 2, Name = "Bob", Department = "IT", Salary = 60000 },

            new Employee { Id = 3, Name = "Charlie", Department = "Finance", Salary = 55000 }

        };

        // CREATE: POST /api/employees

        [HttpPost]

        public ActionResult<Employee> CreateEmployee([FromBody] Employee newEmployee)

        {

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

            employees.Add(newEmployee);

            return CreatedAtAction(nameof(GetEmployee), new { id = newEmployee.Id }, newEmployee);

        }

        // READ: GET /api/employees/{id}

        [HttpGet("{id}")]

        public ActionResult<Employee> GetEmployee(int id)

        {

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

            if (emp == null)

                return NotFound("Employee not found");

            return Ok(emp);

        }

        // UPDATE: PUT /api/employees/{id}

        [HttpPut("{id}")]

        public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updatedEmployee)

        {

            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 = updatedEmployee.Name;

            existing.Department = updatedEmployee.Department;

            existing.Salary = updatedEmployee.Salary;

           return Ok(existing);

        }

  // DELETE: DELETE /api/employees/{id}

        [HttpDelete("{id}")]

        public ActionResult DeleteEmployee(int id)

        {

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

            if (emp == null)

                return NotFound("Employee not found");

  employees.Remove(emp);

            return Ok("Employee deleted successfully");

        }

    }

}

Program.cs:

using FirstWebApi.Models;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add controllers and Swagger

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

    c.SwaggerDoc("v1", new OpenApiInfo

    {

        Title = "Employee API",

        Version = "v1",

        Description = "Simple Web API for CRUD operations on employees"

    });

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseAuthorization();

app.MapControllers();

app.Run();

Output:







