**EMS Full Stack App-Requirement.**

Source Code:

* Department Model Class

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace P3EmployeeMain.Models

{

[Table("DeptMaster")]

public class Department

{

[Key]

public int DId { get; set; }

[StringLength(100)]

public string DName { get; set; }

public string PropertyName { get; set; }

public virtual ICollection<Employee> Employee { get; set; }

}

}

* Employee Model Class

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Diagnostics.Contracts;

namespace P3EmployeeMain.Models

{

[Table("Employee")]

public class Employee

{

[Key]

public int EId { get; set; }

[StringLength(100)]

public string Ename { get; set; }

public int DId { get; set; }

public DateTime DateOfBirth { get; set; }

public string Email { get; set; }

// public virtualDeptMaster DeptMaster { get; set; }

}

}

* Employee Controller Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using P3EmployeeMain.Data;

using P3EmployeeMain.Models;

namespace P3EmployeeMain.Controllers

{

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

[ApiController]

public class EmployeesController : ControllerBase

{

private readonly EmployeeDbContext \_context;

public EmployeesController(EmployeeDbContext context)

{

\_context = context;

}

// GET: api/Employees

[HttpGet]

public async Task<ActionResult<IEnumerable<Employee>>> GetEmployee()

{

if (\_context.Employee == null)

{

return NotFound();

}

return await \_context.Employee.ToListAsync();

}

// GET: api/Employees/5

[HttpGet("{id}")]

public async Task<ActionResult<Employee>> GetEmployee(int id)

{

if (\_context.Employee == null)

{

return NotFound();

}

var employee = await \_context.Employee.FindAsync(id);

if (employee == null)

{

return NotFound();

}

return employee;

}

// PUT: api/Employees/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutEmployee(int id, Employee employee)

{

if (id != employee.EId)

{

return BadRequest();

}

\_context.Entry(employee).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!EmployeeExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Employees

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Employee>> PostEmployee(Employee employee)

{

if (\_context.Employee == null)

{

return Problem("Entity set 'EmployeeDbContext.Employee' is null.");

}

\_context.Employee.Add(employee);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetEmployee", new { id = employee.EId }, employee);

}

// DELETE: api/Employees/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteEmployee(int id)

{

if (\_context.Employee == null)

{

return NotFound();

}

var employee = await \_context.Employee.FindAsync(id);

if (employee == null)

{

return NotFound();

}

\_context.Employee.Remove(employee);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool EmployeeExists(int id)

{

return (\_context.Employee?.Any(e => e.EId == id)).GetValueOrDefault();

}

}

}

* appSettings.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"EmployeeDbContext": "Server=RADHIKAGOWDA;Database=P3Main1Employee;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

}