**1. Create a new ASP.NET Core Web API Project**

dotnet new webapi -n EmployeeManagementAPI

cd EmployeeManagementAPI

**2. Define the Employee Model**

Create a new file **Models/Employee.cs**:

namespace EmployeeManagementAPI.Models

{

public class Employee

{

public int EmployeeId { get; set; }

public string FirstName { get; set; } // Corrected from Name to FirstName/LastName

public string LastName { get; set; }

public DateTime DateOfBirth { get; set; }

public string Position { get; set; }

public decimal Salary { get; set; }

}

}

**3. Create the Employee Service**

Create a new file **Services/EmployeeService.cs**:

using EmployeeManagementAPI.Models;

using System.Collections.Generic;

using System.Linq;

namespace EmployeeManagementAPI.Services

{

public class EmployeeService

{

private readonly List<Employee> \_employees;

public EmployeeService()

{

\_employees = new List<Employee>

{

new Employee { EmployeeId = 1, FirstName = "John", LastName = "Doe", DateOfBirth = new DateTime(1985,5,15), Position = "Manager", Salary = 75000m },

new Employee { EmployeeId = 2, FirstName = "Jane", LastName = "Smith", DateOfBirth = new DateTime(1990,7,20), Position = "Developer", Salary = 65000m },

new Employee { EmployeeId = 3, FirstName = "Alice", LastName = "Johnson", DateOfBirth = new DateTime(1980,12,10), Position = "Designer", Salary = 70000m }

};

}

public List<Employee> GetAllEmployees() => \_employees;

public Employee? GetEmployeeById(int id) => \_employees.FirstOrDefault(e => e.EmployeeId == id);

public Employee AddEmployee(Employee newEmployee)

{

int newId = \_employees.Max(e => e.EmployeeId) + 1;

newEmployee.EmployeeId = newId;

\_employees.Add(newEmployee);

return newEmployee;

}

}

}

**4. Register the Service in Program.cs**

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddSingleton<EmployeeService>();

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**5. Create the Employee Controller**

Create **Controllers/EmployeeController.cs**:

using EmployeeManagementAPI.Models;

using EmployeeManagementAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace EmployeeManagementAPI.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private readonly EmployeeService \_employeeService;

public EmployeeController(EmployeeService employeeService)

{

\_employeeService = employeeService;

}

// GET: api/employee

[HttpGet]

public IActionResult GetAllEmployees()

{

var employees = \_employeeService.GetAllEmployees();

return Ok(employees); // HTTP 200

}

// GET: api/employee/{id}

[HttpGet("{id}")]

public IActionResult GetEmployeeById(int id)

{

var employee = \_employeeService.GetEmployeeById(id);

if (employee == null)

return NotFound(); // HTTP 404

return Ok(employee); // HTTP 200

}

// POST: api/employee

[HttpPost]

public IActionResult CreateEmployee([FromBody] Employee newEmployee)

{

if (newEmployee == null || string.IsNullOrEmpty(newEmployee.FirstName) || string.IsNullOrEmpty(newEmployee.LastName))

return BadRequest(); // HTTP 400

var createdEmployee = \_employeeService.AddEmployee(newEmployee);

return CreatedAtAction(nameof(GetEmployeeById), new { id = createdEmployee.EmployeeId }, createdEmployee); // HTTP 201

}

}

}

**6. Enable Swagger Documentation**

Swagger is already enabled in Program.cs with:

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

Run the app and access Swagger UI at:

https://localhost:8080/swagger/index.html

**7. Run the Project**

dotnet restore

dotnet build

dotnet run

* Use **Swagger** to test endpoints:
  + **GET** /api/employee → Get all employees
  + **GET** /api/employee/{id} → Get employee by ID
  + **POST** /api/employee → Add a new employee

**8. Status Codes Implementation**

| **Operation** | **Status Code** | **Notes** |
| --- | --- | --- |
| Get all employees | 200 OK | Returns list of employees |
| Get employee by ID (found) | 200 OK | Returns employee details |
| Get employee by ID (not found) | 404 NotFound | Employee does not exist |
| Create new employee (valid) | 201 Created | Returns created employee with URL |
| Create new employee (invalid) | 400 BadRequest | Missing required data |

This setup follows **coding standards, proper HTTP methods, and status codes**, uses **in-memory static data**, and includes **Swagger for testing**