CODE EXPLANATION :

System Overview:

A. Data Access Layer (DAL):

This layer is responsible for handling database operations.

It uses Entity Framework Code First Approach to create and manage the database.

Entity Model classes Department and Employee are defined to represent the database tables.

1. Department Entity:

Properties: DeptCode (int), DeptName (string) Represents a department with a unique code, name, and a collection of employee profiles associated with it.

2. Employee Entity:

Properties: EmpCode (int), DateOfBirth (DateTime), EmpName (string), Email (string), Department_Code (int), Department (Department navigation property)

Represents an employee profile with a unique code, date of birth, name, email, department code, and a reference to the department it belongs to.

3. ContextClass:

Inherits from Phase2EndProjectContext class.

Manages the database context and includes DbSet properties for Department and Employee.

Handles database migrations and initialization.

4. Data Seeding:

A class inherits DropCreateDatabaseIfModelChanges<ContextClass> and overrides the Seed method to populate initial data for the Department table.

5. Data Repository Pattern:

Implement CRUD (Create, Read, Update, Delete) operations for Employee entity using the Data Repository Pattern.

6. Business Logic Layer (BLL):

This layer references the DAL library and invokes its functionalities.

It acts as an intermediary between the DAL and the App Service Layer.

7. App Service Layer:

This layer is an ASP.NET Web Application (Web API) responsible for exposing RESTful services to interact with employee data.

It includes a Web API controller with action methods to handle HTTP requests for functionalities such as creating, reading, updating, and deleting employee records.

Attribute-based routing is used to define the routes for these action methods.

Swagger support is enabled for API documentation and testing.

8. App UI Layer:

This layer is responsible for the user interface and is intended to be developed using Angular or ReactJS.

It communicates with the App Service Layer through RESTful API calls to perform CRUD operations on employee data.

9. Sample Input/Output:

The sample input/output provided illustrates how to test various functionalities using Swagger, a tool for API documentation and testing.

You can use Swagger to perform actions like saving an employee's name, retrieving all employee details, getting an employee by code, updating employee details, and deleting an employee record.

Each action is accompanied by a description of what it does and, in some cases, a "before" and "after" state to confirm the functionality's success.

GIT LINK : git@github.com:sundar2568223/Phase2EndProject-1.git