**Employee Management System (ASP.NET Core Web API)**

**Project Overview**

The Employee Management System is an ASP.NET Core Web API application designed to manage employee profiles and their associated departments. This system allows users to perform CRUD (Create, Read, Update, Delete) operations on employee profiles while ensuring that department information is accurately maintained.

**Key Components**

**1. Data Access Layer (DAL)**

- Entity Framework Code First Approach: The DAL layer utilizes Entity Framework to create and manage the database. It follows the Code First approach to define the data model using C# classes and then generates the database schema accordingly.

- Entity Model Classes: The application defines two primary entity classes:

- `EmpProfile`: Represents employee profiles with properties such as `EmpCode`, `EmpName`, `Email`, `DateOfBirth`, and `DeptCode`. It also has a navigation property `DeptMaster` to represent the associated department.

- `DeptMaster`: Represents department information with properties like `DeptCode` and `DeptName`.

- Context Class: A context class inherits from DbContext and serves as the bridge between the application and the database. It includes DbSet properties for both `EmpProfile` and `DeptMaster` entities.

- Data Repository Pattern: The DAL layer implements a Data Repository Pattern to encapsulate database operations. It provides functionalities such as saving employee details, retrieving all employee details, fetching an employee by code, updating employee details, and deleting employee records.

**2. Business Logic Layer (BLL)**

In this layer, the controller performs operations for fetching all employee details, retrieving employee details by their unique id, deleting employee records by their id, updating employee information by their id, and inserting new employee details.

**3. App Service Layer (Web API**

**- ASP.NET Core Web Application:** This layer is developed as an ASP.NET Core Web API application that exposes RESTful services to interact with the system.

**- API Controllers:** The Web API includes controllers for handling employee-related operations, such as creating, reading, updating, and deleting employee profiles. It also integrates department-related actions.

**- Attribute-Based Routing:** Attribute-based routing is used to define routes for API actions, making it easy to access specific endpoints.

**- Swagger Documentation:** The application is equipped with Swagger support, allowing developers to document and test the API endpoints interactively.

**Project Workflow**

**1. Create Employee Profile:** Users can create a new employee profile by sending a POST request to the appropriate API endpoint. The employee is associated with a specific department identified by `DeptCode`.

**2. Read Employee Profile:** Users can retrieve a list of all employee profiles or fetch the details of a specific employee by providing their `EmpCode`.

**3. Update Employee Profile:** Updating an employee profile involves sending a PUT request to the API with the modified data. If the `DeptCode` is changed, the associated `DeptMaster` entity is updated accordingly.

**4. Delete Employee Profile:** To remove an employee profile, users can send a DELETE request to the API. If the department has no more employees after deletion, the associated department is also deleted.

**Conclusion**

The Employee Management System is a robust ASP.NET Core Web API project that offers a flexible and efficient way to manage employee profiles and their departments. By following the principles of good software design and adhering to the Data Repository Pattern, this application ensures data integrity and maintainability.

**GitHub Link:**

https://github.com/Vasanth30e/Practice\_Project\_Phase3/tree/master/Phase%20End%20Project