**EMS Full Stack App-Requirement .**

**WriteUp:**

Introduction:

Simplona Tech. Solutions requires the development of a Full Stack Employee Management System (EMS) to effectively manage employee profile details. This project incorporates fundamental technologies such as C#, MS-SQL Server, ASP.Net MVC, and Web API. The application follows a multi-layered architecture, ensuring efficient data handling and providing a comprehensive solution for employee management.

System Components:

1. Data Access Layer (DAL):

Entity Model Classes:

Define entity classes such as DeptMaster and EmpProfile to represent the database structure. These classes capture essential details about departments and employees.

Context Class:

Implement a Context class that inherits from DbContext to facilitate communication with the database. This class includes DbSet properties for entities like DeptMaster and EmpProfile.

Database Initialization:

Develop a database initializer class that utilizes the DropCreateDatabaseIfModelChanges strategy. This class seeds default data into the DeptMaster table.

2. Business Logic Layer (BLL):

DAL Library Reference:

Establish a reference to the DAL project within the BLL project to enable communication between these layers.

BLL Class:

Create a dedicated class in the BLL to interact with the DAL. This class invokes various functionalities related to employee details, ensuring a separation of concerns within the application.

3. App Service Layer:

ASP.Net Web Application (Web API):

Web API Controller:

Develop a Web API controller responsible for handling HTTP requests related to employee management. This controller interacts with the BLL to perform CRUD operations on employee data.

Swagger Documentation:

Enable Swagger support to facilitate documentation and testing of the Web API. Swagger enhances the usability of the API by providing a user-friendly interface for testing and understanding available functionalities.

4. App UI Layer:

Angular or ReactJS Application:

The UI layer of the application is designed using Angular or ReactJS. This layer is responsible for issuing requests to the Web API to consume its functionalities. Although specific details about the UI implementation are not provided in this document, it plays a crucial role in providing a user-friendly interface for interacting with the EMS.

Sample Input/Output Scenarios:

Testing Functionalities using Swagger:

Users can access the Swagger index page to test all functionalities provided by the Web API.

Testing Save Functionality using Swagger:

The application allows the addition of employee details through Swagger. Users can save employee profiles, and the system ensures the data is stored accurately in the database.

Testing Get All Functionality using Swagger:

Swagger facilitates the retrieval of all employee details, ensuring that the system effectively gathers and presents this information.

Testing Get by Code Functionality using Swagger:

Users can retrieve specific employee details by code using Swagger, providing a targeted approach to accessing employee information.

Testing Update Functionality using Swagger:

After updating employee details via Swagger, users can confirm the changes by issuing a Get by Code request. This ensures the successful implementation of the update functionality.

Conclusion:

This Full Stack Employee Management System offers a comprehensive solution for Simplona Tech. Solutions to manage employee profiles efficiently. The application's architecture and design principles ensure scalability and maintainability. The inclusion of Swagger enhances the overall user experience by providing a user-friendly interface for testing and documentation, contributing to the success of the EMS. While specific UI details are not provided, the Angular or ReactJS layer is integral to delivering a seamless and intuitive user interface for the EMS.

GitHub Link: <https://github.com/sivareddy3116/phaaseEndProject>