**Employee Management Application**

This documentation provides a concise guide on setting up and running an \*\*Employee Management Application\*\* built with \*\*ASP.NET MVC\*\* and \*\*Web API\*\*. The app allows users to perform CRUD (Create, Read, Update, Delete) operations on employee records. Below is a brief summary:

**Project Structure**

**1. \*\*Controllers\*\*:**

- `EmployeeController`: Handles API endpoints for employee operations (list, add, update, delete).

**2. \*\*Models\*\*:**

- `Employee`: Defines the properties of an employee like `EmployeeID`, `Name`, `Age`, `State`, and `Country`.

**3. \*\*Database Interaction\*\*:**

- `EmployeeDB`: Provides methods to interact with the database (using \*\*Entity Framework\*\*), such as `ListAll()`, `AddOrUpdate()`, and `Delete()`.

**4. \*\*Views\*\*:**

- `Home/Index.cshtml`: Displays the employee list and a modal for adding/editing employees.

**5. \*\*Scripts\*\*:**

- `Scripts.js`: Manages AJAX calls for CRUD operations and dynamically updates the employee table without reloading the page.

**Technologies Used**

- \*\*ASP.NET MVC\*\* for the front-end UI.

- \*\*ASP.NET Web API\*\* for the back-end RESTful API.

- \*\*JavaScript, jQuery, and AJAX\*\* for dynamic content updates.

- \*\*Entity Framework\*\* for database interactions.

- \*\*Bootstrap\*\* for responsive design and \*\*SweetAlert2\*\* for notifications.

**Steps to Run Locally:**

1. \*\*Download the Project\*\*: Download the repository.

2. \*\*Set Up Database\*\*:

- Create a SQL Server database (e.g., `EmployeeDB`).

- Update the connection string in `appsettings.json`.

- Run the database script from the folder(Folder Name:-Db Script)

-Install the necessary NuGet packages for Entity Framework Core

**Install-Package Microsoft.EntityFrameworkCore**

**Install-Package Microsoft.EntityFrameworkCore.SqlServer**

3. \*\*Build and Run\*\*:

- Open the project in \*\*Visual Studio\*\* and run the application.

**API Testing**

- Use \*\*Postman\*\* to test API endpoints for adding, updating, retrieving, and deleting employee records.

**Code Breakdown**

- \*\*Controller\*\*: Handles the API CRUD operations.

- \*\*Database\*\*: Methods in `EmployeeDB` interact with the database to fetch, add, update, and delete employee records.

- \*\*AJAX\*\*: Handles asynchronous interactions for adding/updating employees, dynamically updating the employee list in the UI.

**Integration and Testing**

- \*\*Test the API\*\* using tools like swagger.

**Conclusion**

This documentation explains how to set up the \*\*Employee Management Application\*\* locally, how to test the functionality, and provides a structure of the code. The app combines \*\*ASP.NET MVC\*\* for the front-end and \*\*ASP.NET Web API\*\* for the back-end to allow seamless CRUD operations with real-time updates using \*\*AJAX\*\*.