.NET Core Azure Sprint

SALES APPLICATION

Document Revision History

Date	Revision No.	Author	Summary of Changes
22-06-2023	1.0	M .Monisha Devi	Case Study Created

Table of Contents

INTRODUCTION

This document outlines a project for the .NET Line of Technology (LOT). The project is todevelop Attendance and Leave Management System. This document contains the requirements, workflow of the system and gives guidelines on how to build the functionality gradually in each of the course modules of the .NET LOT.

SETUP CHECKLIST

Minimum System Requirements

- Intel Pentium 4 and above Windows 7
- Memory 8 GB
- Edge / Chrome browser
- SQL Server 2019 and SQL Server Management Studio
- Visual Studio 2022
- Visual Studio Code
- Git

INSTRUCTIONS

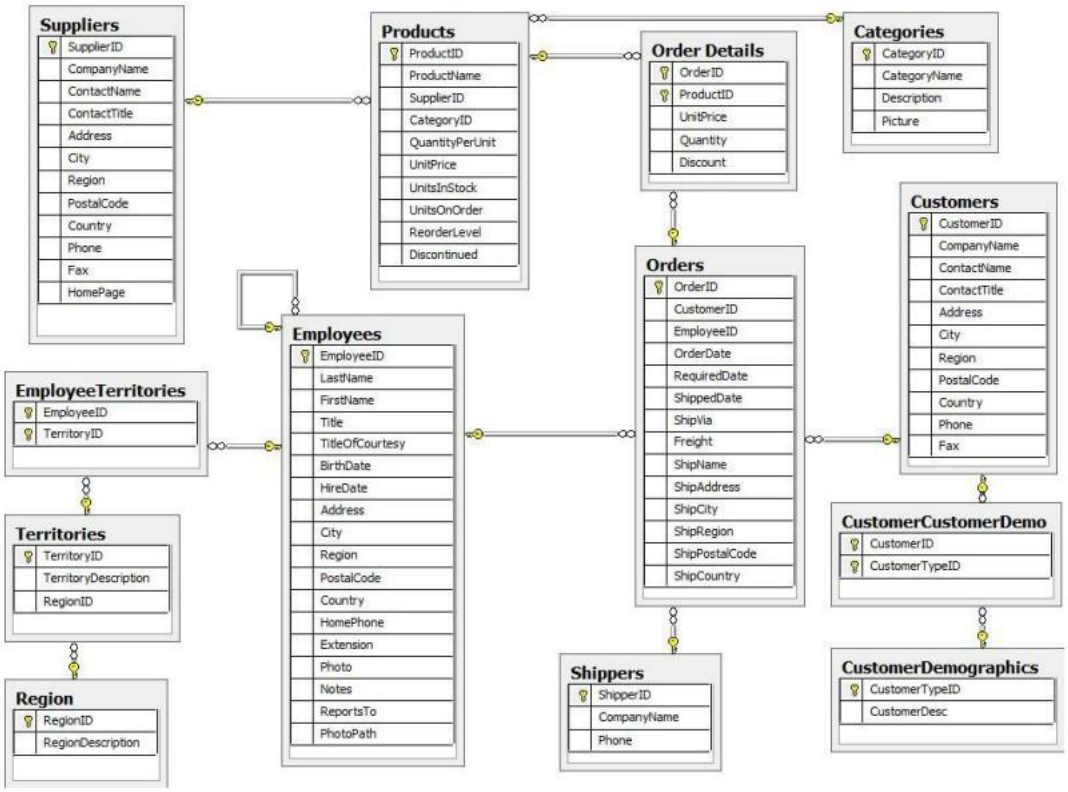
- The code modules in the Sprint should follow all the coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory Sprint. Store your Project here.
- High Level Design document should be created (HLD).
- You can refer to your course material.
- You may also look up the help provided in the MSDN
- Since this project work will span over week, you will need to take care of maintaining the code.

PROBLEM STATEMENT

OBJECTIVE

Development of Sales Application - This application is greatly used by the Admin and Employees to view and manage the sales .

Database Design:



Instruction:

- Add password nvarchar(50) column in Employees table in database
- Store the Admin details in server side code
- Add email varchar(30) and password nvarchar(50) column in Shippers table

1) Administrator

   a) View /Approve Shipper Information
   b) View / Approve Employee Information
   c) Add/Edit/View Territory
   d) Search Territories under each Region
   e) View number of Territories under each Region
   f) Search Employee by City
   g) Search Employee by Region
   h) View Employee by Title
   i) View the Employee by HireDate
   j) View Employee who placed Highest Sale in a date/year
   k) View Employee who placed Lowest Sale in a date/year
   l) View the number of shipment made by each shipper
   m) View the total amount earned by each shipper
   n) View the amount earned by shipper in a date/year
   o) View total number of orders made by each employee
   p) View Order placed by each Employee
   q) View Ship details for an order
   r) View Ship details between given dates
   s) View all Ship details
   t) View all OrderDetails
   u) View bill amount of all order, sales made by an employee

2) Shipper

   a) Add/Edit/View his details

3) Employees

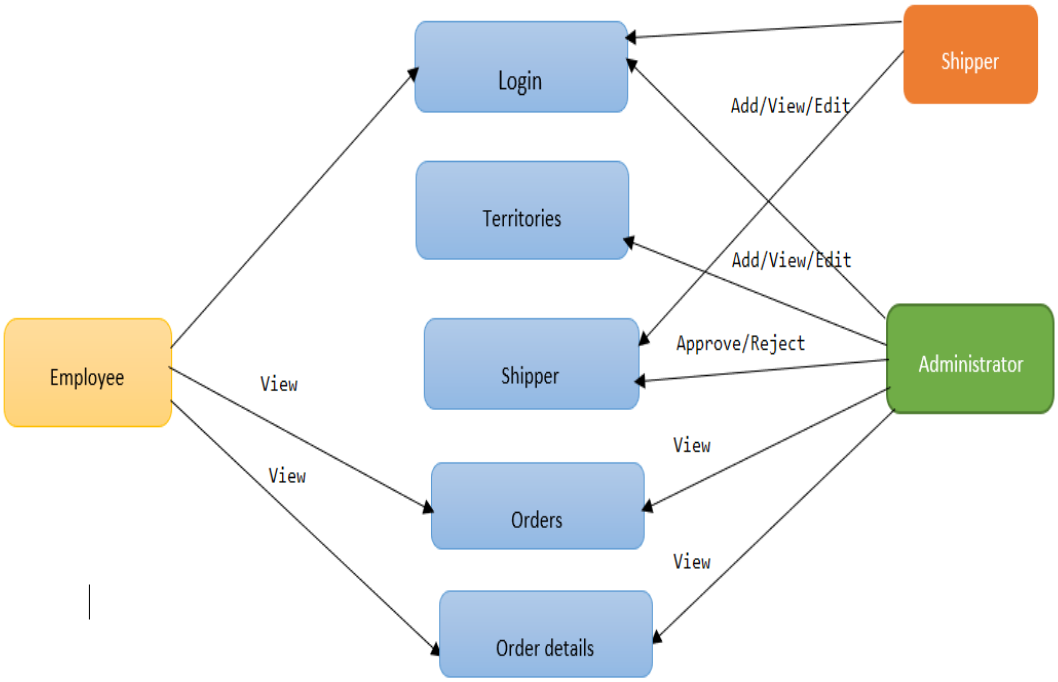
   a) Add/Update Employee details
   b) View Sales made by him in a day/month/year
   c) View Sales made by him between dates
   d) View his highest sale
   e) View all the Company Name sold by an Employee
   f) View all the Order placed by an Employee

Create Client Application in Angular and Services using ASP.NET Core Web API

MODULE LIST AND MODULE DETAILS

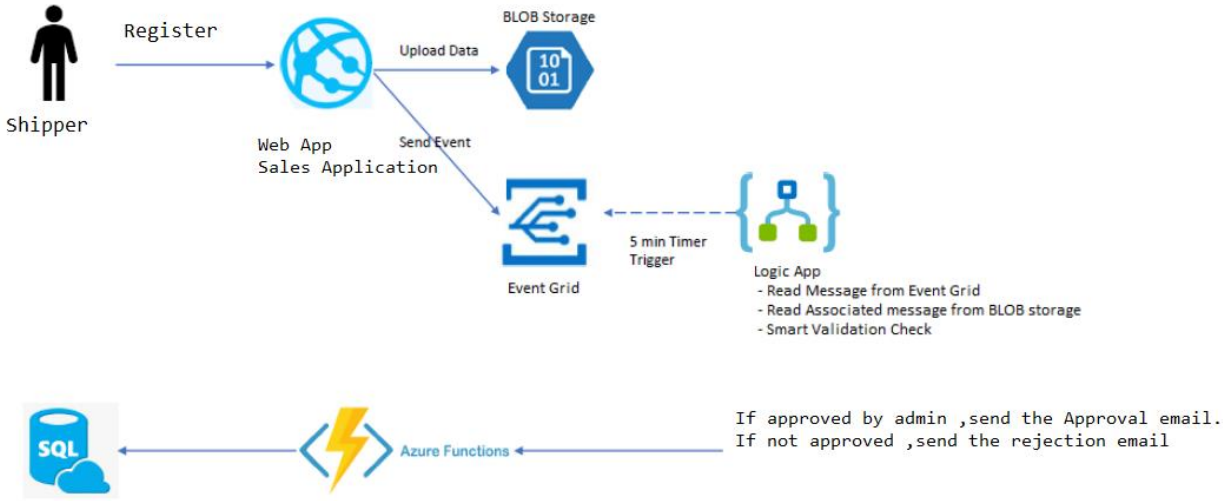
Modules

- Admin
- Shipper
- Employee
- Orders
- Login



PROJECT STRUCTURE

The project should have minimum target framework – .NET 6.0 for Core WEB API Services andAngular 10 for UI Design



API EXPECTATION

Create API using

- Visual Studio 2022 IDE
- .NET 6.0 / C# 5.
- Angular
- SQL Server 2019
- Entity Framework Core

REQUIREMENT

In this sprint, you must use:

- Angular
- Use Entity Framework Core
- Create Web API and host separately
    - Functionality will be like services logic
- Web App
- Azure BLOB Storage
- Azure Function
- Azure Logic App

Create Client Application in Angular and Services using Web API Core and host them on Azure Cloud.

Design guidelines

- All the exceptions/errors to be captured and user-friendly message to be displayed on the Common Error page.

- .NET Web API must be created using Database First Approach.

End Points are given below

Note : Add the appropriate error message for each response

Endpoints	HTTP Verb/Method	Description	Success Response	Error Response
/api/shipper/	POST	Add shipper	String: Record Created Successfully	

/api/shipper/	GET	Display all Shipper	Collection of shipper	
/api/shipper/edit/{ShipperID}	PUT	Edit the Shipper Details	No Content 204	
/api/shipper/edit/{ShipperID}	PATCH	Edit the particular Shipper Details	No Content 204	
/api/shipper/{CompanyName}	GET	Search Shipper by CompanyName	Shippers	
/api/shipper/totalshipment	GET	Display the total Shipment made by Each Shipper	Shipper , Total Shipment	
/api/shipper/totalamountearnedbyshipper	GET	Display the sum amount earned by each shipper	Company Name, Total Amount	
/api/shipper/totalamountearnedbyshipper/{date}	GET	Display the sum amount earned by each shipper in a date	Company Name, Total Amount	
/api/shipper/totalamountearnedbyshipper/{year}	GET	Display the sum amount earned by each shipper in a year	Company Name, Total Amount	

Endpoints	HTTP Verb/Method	Description	Success Response	Error Response
/api/territory	POST	Add new Territory	String: Record Created Successfully	
/api/territory	GET	Get all territory	Collection of Territory	

/api/territory/edit/{Territoryid}	PUT	Update the territory	No Content 204	

Endpoints	HTTP Verb/Method	Description	Success Response	Error Response
/api/employee/	POST	Add new employee object	String: Record Created Successfully	{"Timestamp": "2023-06-08", "message": "Validation failed ..."}
/api/employee/	GET	Display all employee	Collection of Employees	
/api/employee/edit/{EmployeeID}	PUT	Edit all the EmployeeDetails	No Content 204	
/api/employee/edit/{EmployeeID}	PATCH	Edit particular EmployeeDetails	No Content 204	
/api/employee/title/{title}	GET	Search Employee by title	Collection Employees	
/api/employee/City/{City}	GET	Search employee by city	Collection of Employees	
/api/employee/Region/{RegionDescription}	GET	Search employee by Region	Collection of Employees	
/api/employee/HireDate/{HireDate}	GET	Search employee by HireDate	Collection of Employees	
/api/employee/highestsalebyemployee/{date}	GET	View employees who made the highest sale in a given date	Collection of Employees/Employees	
/api/employee/highestsalebyemployee/{year}	GET	View employees who made the highest sale in a given year	Collection of Employees/Employees	
/api/employee/lowestsalebyemployee/{date}	GET	View employees who made the lowest sale in a given date	Collection of Employees/Employees	

/api/employee/lowestsalebyemployee/{year}	GET	View employees who made the lowest sale in a given year	Collection of Employees/Employees	
/api/employee/salemadebyanemployee /{Employeeid}/{date}	GET	Search the Sales made by an employee on a given date	Collection ofOrderId,Companyname	
/api/employee/ Salemadebyanemployeebetweendates /{EmployeeId}{fromdate}/{todate}	GET	View the Sales made by employee on a between the given dates	Collection of OrderId,Companyname	
/api/employee/companyname/{EmployeeID}	GET	Display the Company Name ,sales made by an Employee	Collection of CompanyName	

Endpoints	HTTP Verb/Method	Description	Success Response	Error Response
/api/orders/	GET	Get All Orders	Collection of Order	
/api/orders/orderbyemployee/{Firstname}	GET	Display all the Order placed by an Employee	Collection of Order	
/api/orders/shipdetails/{OrderId}	GET	Search Shipdetails by OrderId	Collection of ShipName,ShipAddress	
/api/orders/BetweenDate/{FromDate}/{ToDate}	GET	Search Shipdetails by between FromOrderDate and ToOrderDate	Collection of ShipName,ShipAddress,ShipRegion	
/api/orders/allshipdetails	GET	View all ship details	Collection of ShipName,ShipAddress,ShipRegion, ShipRegion,ShipPostalcode	
/api/orders/numberoforderbyeachemployee	GET	View number of orders made by each employee	Collection of Employee Name, TotalOrder	

Endpoints	HTTP Verb/Method	Description	Success Response	Error Response
/api/orderDetails/	GET	Get all order details	Collection of Orderdetails	
/api/orderDetails/{EmployeeID}	GET	View bill amount of all order sales made by an employee	OrderId,Bill amount	

Implementation

SUMMARY OF THE FUNCTIONALITY TO BE BUILT:

The participants need to develop the Attendance and Leave Management System by building the functionality incrementally in each of the course modules of .NET LOT.

Sr. No	Course	Duration (in PDs)	Functionality to be built
1	Angular ASP.NET Core Web API Entity Framework Core SQL Server Azure services	5	Developing Angular App and Web API and hosting them on Web App (Azure App Service) Using Azure BLOB Storage, Using Logic App to validate data, Using Function app to write data to SQL Server

