Assignment 01

Building a Sales Management Application with Razor Page

Introduction

Imagine you're an employee of a product retailer named **FStore**. Your manager has asked you to develop a Razor Page application for member management, product management, and order management. The application has a default account whose email is "**admin@fstore.com**" and password is "**admin@@**" that stored in the **appsettings.json**.

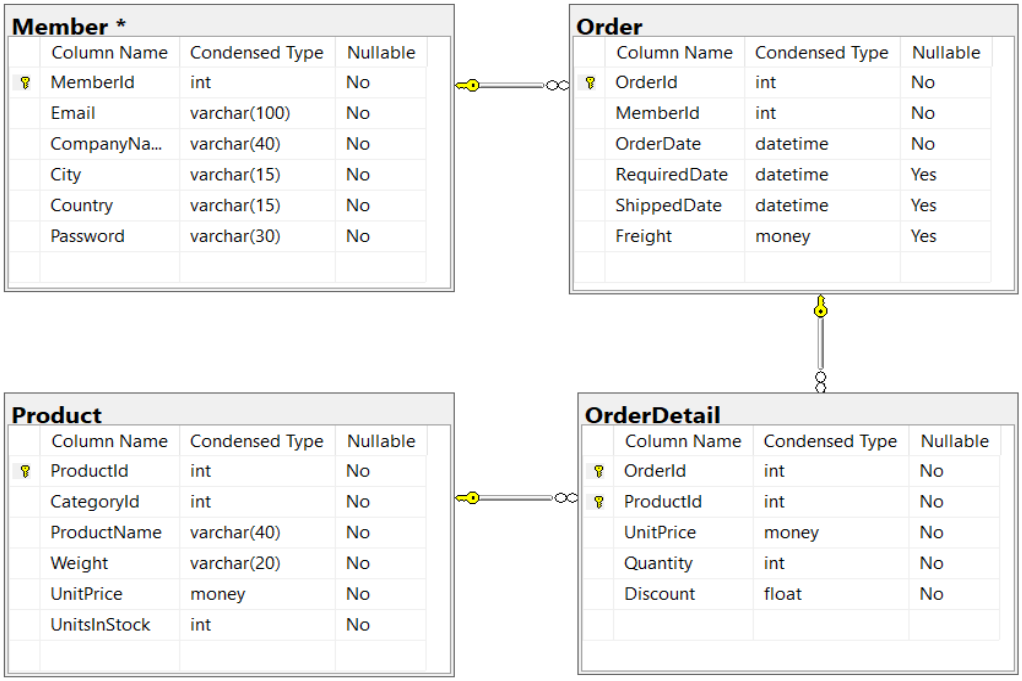
The application has to support adding, viewing, modifying, and removing products—a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD) and Search. This assignment explores creating an application using Razor Page with .NET Core, C#, and ADO.NET / Entity Framework Core. The MS SQL Server database will be created to persist the data and it will be used for reading and managing data.

Assignment Objectives

In this assignment, you will:

- Use the Visual Studio.NET to create Razor Page and Class Library (.dll) projects.
- Perform CRUD actions using ADO.NET or Entity Framework Core.
- Use LINQ to query and sort data.
- Apply passing data in the WPF application.
- Apply 3-Layers architecture to develop the application.
- Apply MVVM (Model-View ViewModel) pattern in the application.
- Apply Dependency injection (DI) in the application.
- Apply Repository pattern and Singleton pattern in a project.
- Add CRUD and searching actions to the application.
- Apply to validate data type for all fields.
- Run the project and test the Razor Page application actions.

Database Design



Member *

	Column Name	Condensed Type	Nullable
🔑	MemberId	int	No
	Email	varchar(100)	No
	CompanyNa...	varchar(40)	No
	City	varchar(15)	No
	Country	varchar(15)	No
	Password	varchar(30)	No

Order

	Column Name	Condensed Type	Nullable
🔑	OrderId	int	No
	MemberId	int	No
	OrderDate	datetime	No
	RequiredDate	datetime	Yes
	ShippedDate	datetime	Yes
	Freight	money	Yes

Product

	Column Name	Condensed Type	Nullable
🔑	ProductId	int	No
	CategoryId	int	No
	ProductName	varchar(40)	No
	Weight	varchar(20)	No
	UnitPrice	money	No
	UnitsInStock	int	No

OrderDetail

	Column Name	Condensed Type	Nullable
🔑	OrderId	int	No
🔑	ProductId	int	No
	UnitPrice	money	No
	Quantity	int	No
	Discount	float	No

Main Functions

- Member management, Product management, and Order management: Read, Create, Update and Delete actions. Creating and Updating actions must be performed by popup dialog
- Search Product by ID , ProductName (by keyword of ProductName), UnitPrice, and UnitInStock
- Create a report statistics sales by the period from StartDate to EndDate, and sort sales in descending order
- Member authentication by Email and Password. If the user is "**Admin**" then allows to perform all actions, otherwise, the normal user is allowed to view/create/update the profile and view their orders history.

Guidelines

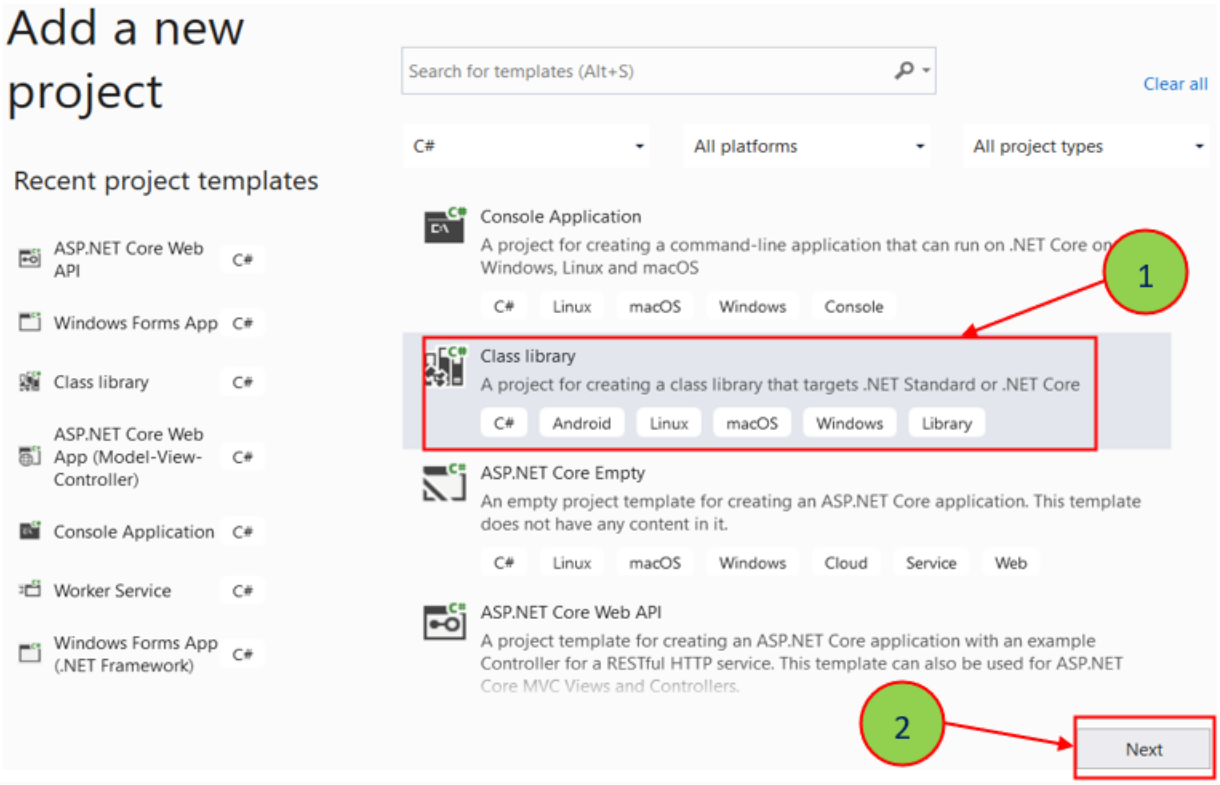
Activity 01: Build a solution [01 mark]

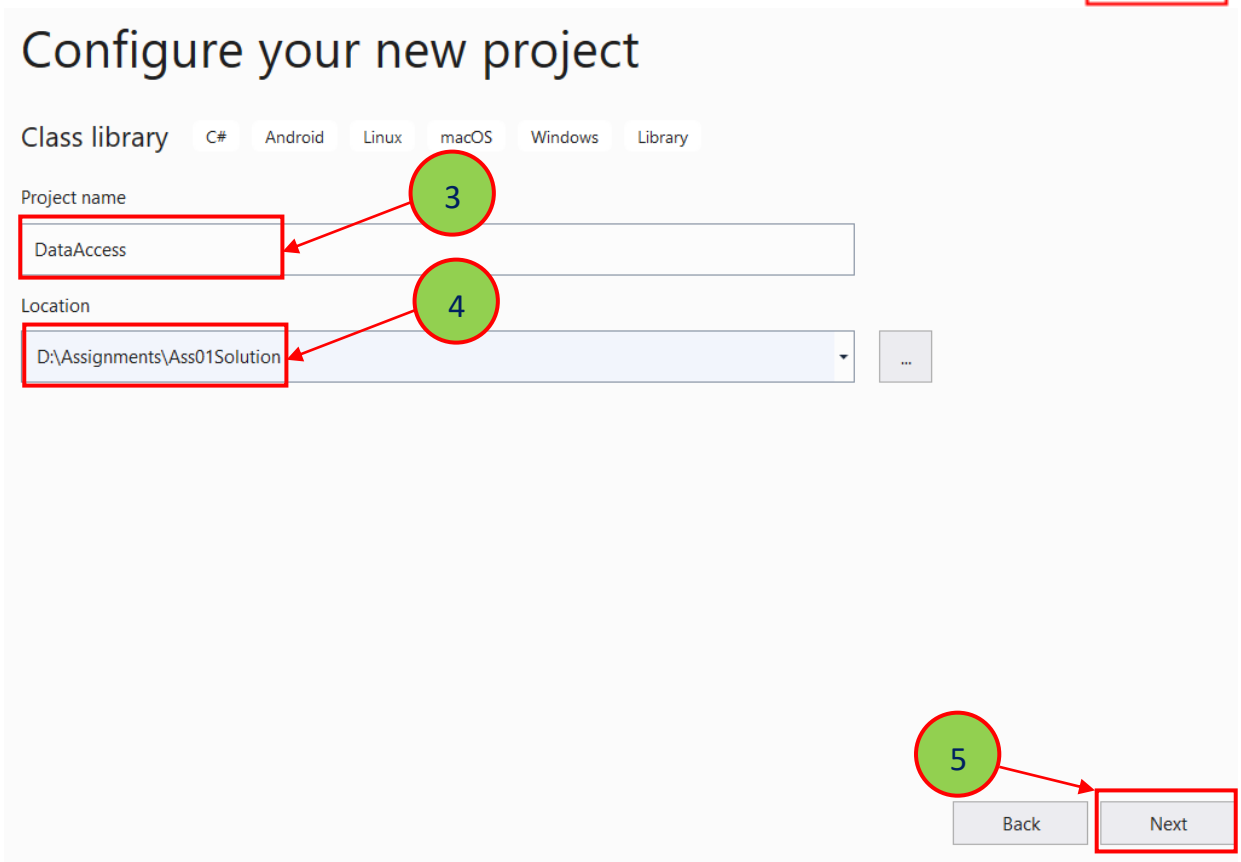
Create a Blank Solution named **Ass01Solution** that includes Class Library Project: **DataAccess, BusinessObject,** and a Windows Presentation Foundation (WPF) project named **SalesRazorPageApp**

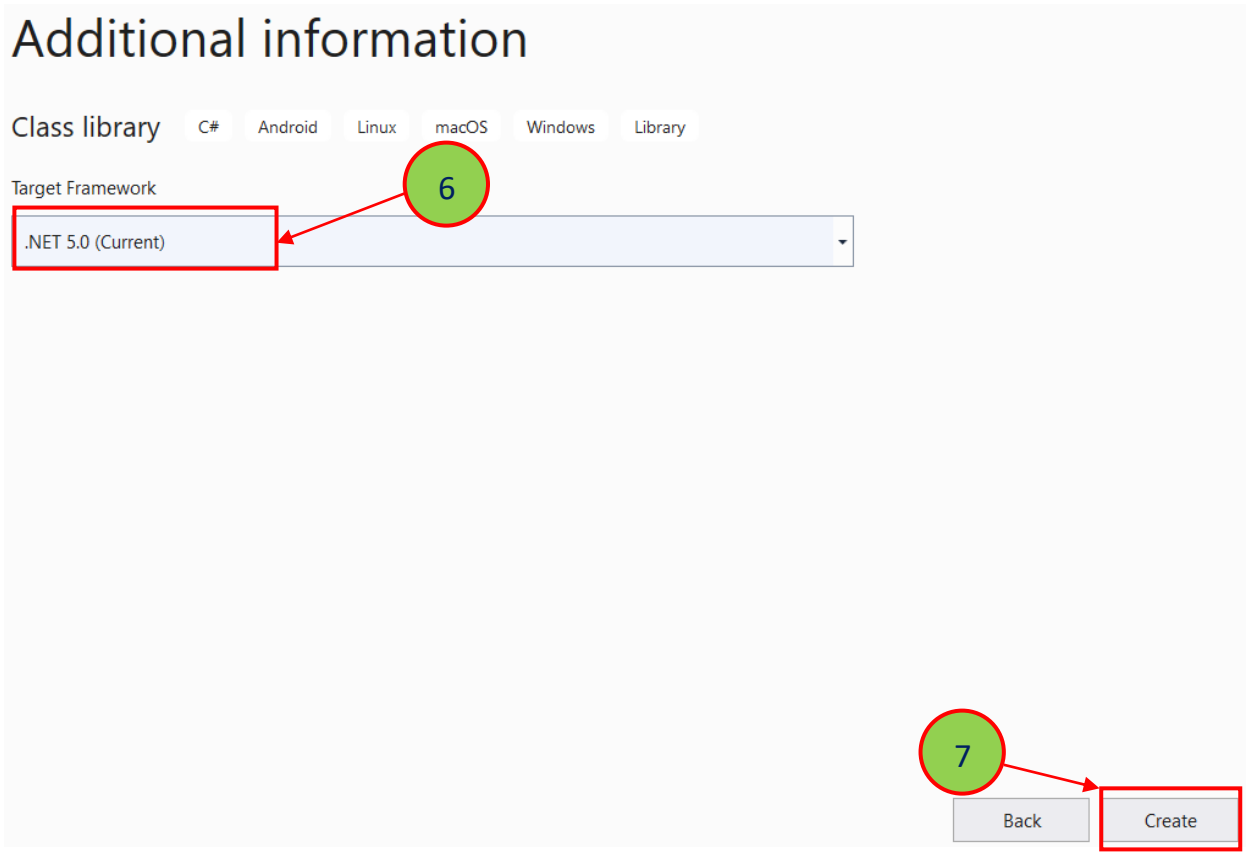
Step 01. Open the Visual Studio .NET application and create a Blank solution named **Asm01Solution**

Step 02. Create a Class Library project named **DataAccess**

From the File menu | Add | New Project, on the Add New Project dialog, select "Class Library" and performs steps as follows:


Add a new project
Search for templates (Alt+S)
Clear all
C#
All platforms
All project types
Recent project templates
ASP.NET Core Web API C#
Windows Forms App C#
Class library C#
ASP.NET Core Web App (Model-View-Controller) C#
Console Application C#
Worker Service C#
Windows Forms App (.NET Framework) C#
Console Application
A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS
C# Linux macOS Windows Console
Class library
A project for creating a class library that targets .NET Standard or .NET Core
C# Android Linux macOS Windows Library
ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
C# Linux macOS Windows Cloud Service Web
ASP.NET Core Web API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
1
2
Next


Configure your new project
Class library C# Android Linux macOS Windows Library
Project name
DataAccess
Location
D:\Assignments\Ass01Solution
...
3
4
5
Back
Next

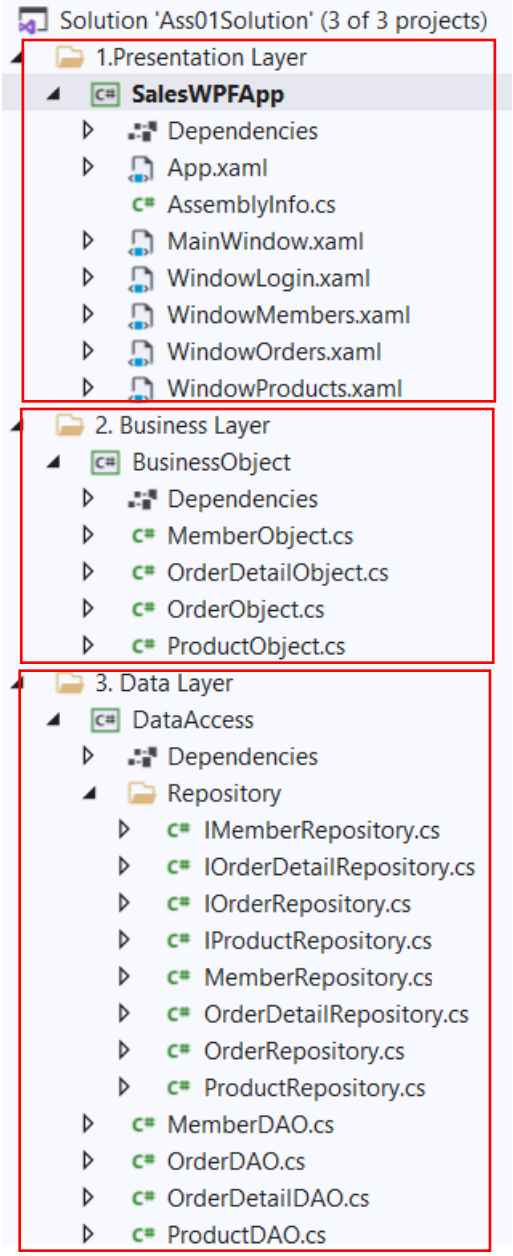


Step 03. Repeat **Step 02** to create a **BusinessObject** project.

Step 04. Create a Razor Page project named **SalesRazorPageApp**

- From the File menu | Add | New Project, on the Add New Project dialog, select "Razor Application" and performs steps as follows:

Step 05. Create folders and add classes to the projects as follows:



Activity 02: Develop BusinessObject project

[02 marks]

Step 01. Write codes to create classes and definition all data members

Step 02. Write codes to perform business rules for data members

Activity 03: Develop DataAccess project [03 marks]

Step 01. Add the project reference to the **BusinessObject** project

Step 02. Write codes for **MemberDAO.cs, IMemberRepository.cs** and **MemberRepository.cs**

Step 03. Write codes for **ProductDAO.cs, IProductRepository.cs** and **ProductRepository.cs**

Step 04. Write codes for **OrderDAO.cs, IOrderRepository.cs** and **OrderRepository.cs**

Step 05. Write codes for **OrderDetailDAO.cs, IOrderDetailRepository.cs** and **OrderDetailRepository.cs**

***Hints*: If using Entity Framework Core, you can install the AutoMapper package from Nuget to map Entity with Busines Object**

Activity 04: Develop SalesWPFApp project [03 marks]

Step 01. Add the reference to **BusinessObject** and **DataAccess** projects

Step 02. Design UI (XAML code) for windows and write codes to perform functions

Activity 05: Run the Razor Page project and test all actions [01 mark]