Assignment 01

Building ASP.NET Web API with Desktop Application/Web Application

Introduction

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

The application has to support adding, viewing, modifying, and removing information - a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD) and Search. This assignment explores creating an ASP.NET Core Web API with C#, and ADO.NET or Entity Framework Core, the client application can be used as Desktop Application (Windows Forms, WPF) or Web Application (ASP.NET Core Web MVC or Razor Pages). An MS SQL Server database will be created to persist the data and it will be used for reading and managing data.

Assignment Objectives

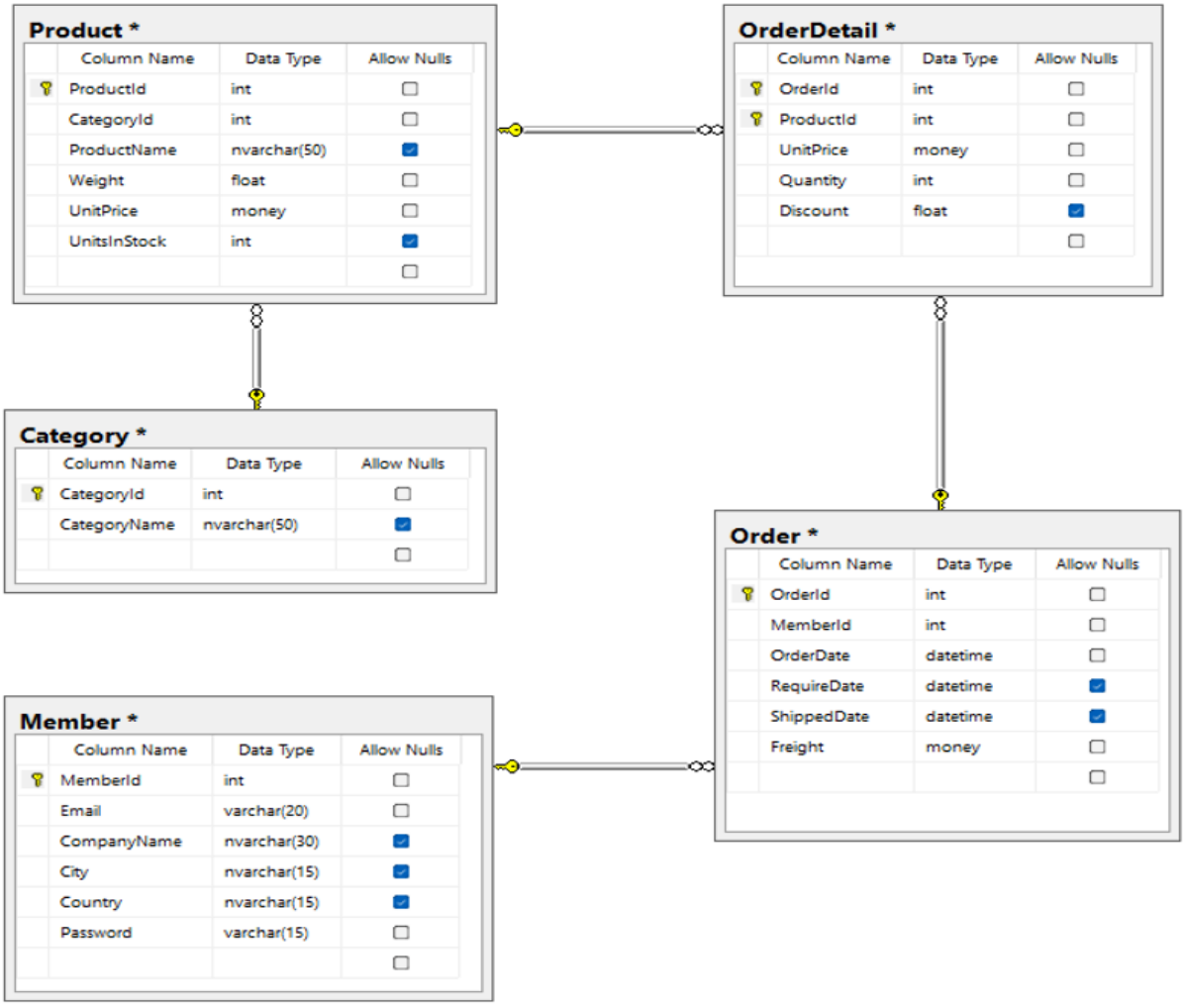
In this assignment, you will:

- Use the Visual Studio.NET to create a Desktop/Web application and ASP.NET Core Web API project.

- Perform CRUD actions using ADO.NET or Entity Framework Core.
- Use LINQ to query and sort data.
- Apply 3-layers architecture to develop the application.
- Apply Repository pattern and Singleton pattern in a project.
- Add CRUD and searching actions to the Desktop/Web application with ASP.NET Core Web API.
- Apply to validate data type for all fields.
- Run the project and test the actions of the Web application.

Database Design

Database name: PRN231_AS1



Main Functions

- Create Web API with Member management, Product management, and Order management: Read, Create, Update and Delete actions.
- Create Client application (with Desktop/Web application) interactive with WebAPI to perform these functions:
  - Search ProductName (keywork of ProductName) and UnitPrice
  - 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**" (presented above) then allows to perform all actions, otherwise, the normal user (stored in DB) is allowed to view/update the profile and view their orders history.

Guidelines

Activity 01: Build a solution

Create a Blank Solution named **Assignment01Solution_StudentCode** that includes Class Library Project: **DataAccess, BusinessObject,** an ASP.NET Core WebAPI project named **eStoreAPI,** and **eStoreClient** (can be used Windows Forms, WPF or Web Application)

Step 01. Open the Visual Studio .NET application and create a Blank solution named **Assignment01Solution_StudentCode**

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

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

Step 04. Create an ASP.NET Core MVC project (or ASP.NET Core Razor Pages) named **eStore**

Step 05. Create folders and add class to the projects.

Activity 02: Develop BusinessObject project

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

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

Step 01. Add a 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**

Activity 04: Develop eStoreAPI project

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

Step 02. Write codes for **MemberAPI.cs,**

Step 03. Write codes for **ProductAPI.cs**

Step 04. Write codes for **OrderAPI.cs**

Step 05. Write codes for **OrderDetailAPI.cs**

Activity 05: Develop eStoreClient project

Step 01. You can choose Desktop Application (Windows Forms/WPF) or Web Application (ASP.NET Core MVC/ASP.NET Core Razor Pages) for eStoreClient project

Step 02. Add a project reference to the **BusinessObject** project

Step 03. Write codes for **Client Project** (It depends on the option you choose in the Step 01.)

Activity 06: Run the Web project and test all actions