Assignment 03

Apply security, Ajax request with RESTful API and Web Application

Introduction

Imagine you're an employee of a product retailer named **eStore**. Your manager has asked you to develop a 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**.

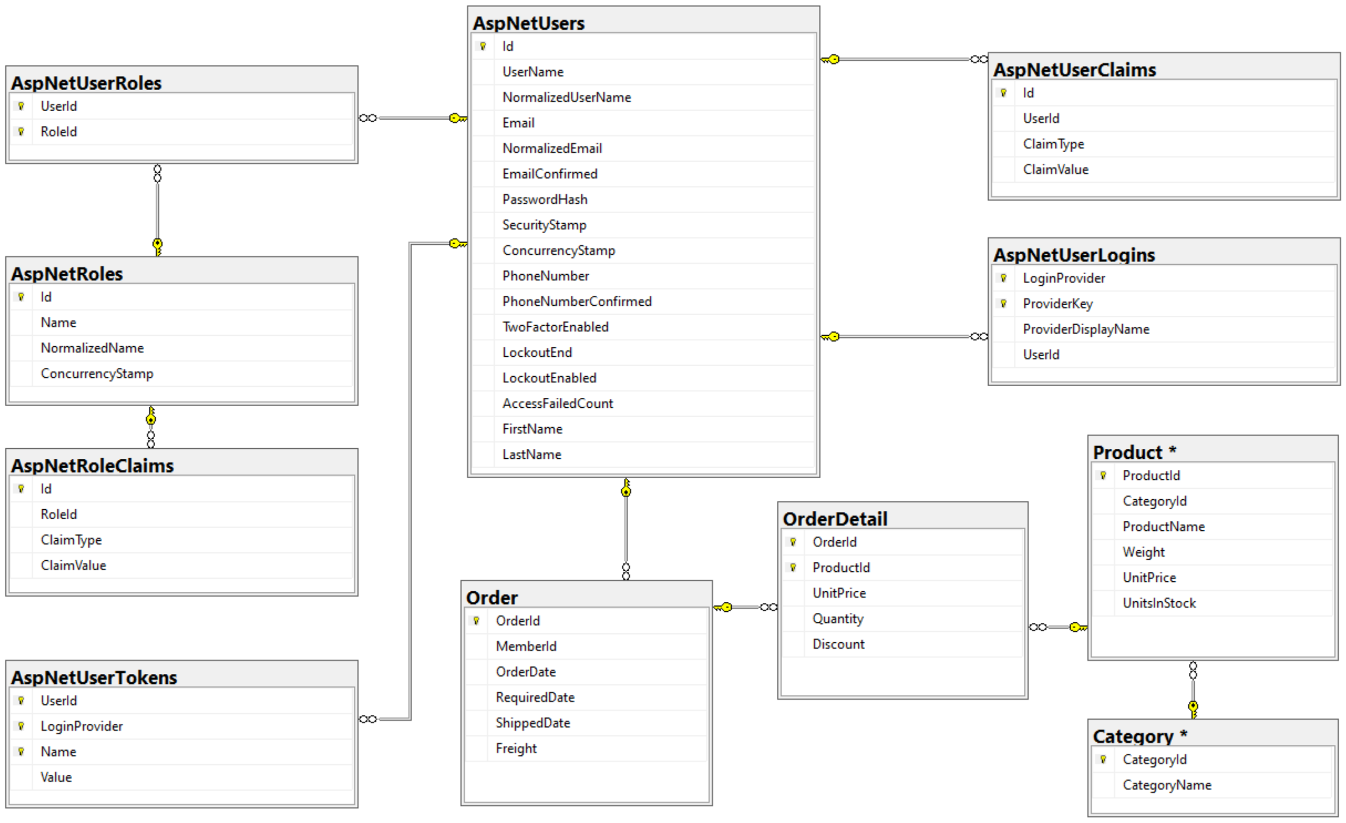
The application has to support adding, viewing, modifying, and removing information - a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD) and Search. This assignment explores creating an ASP.NET Core Web API with C#, Identity or JWT and Entity Framework Core, the client application can be used as Web Application (ASP.NET Core Web MVC or Razor Pages) combination with AJAX request. 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 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 Web application combination with AJAX request.
- Apply to validate data type for all fields.
- Run the project and test the actions of the Web application.

Sample Database Design



Main Functions

- Use Identity to manage User, manage login authentication process, build application with ASP.NET's account registration, login/logout function, configure Identity lockout to lock user if login fails many times.
- Create Web API with Product management, and Order management: Read, Create, Update and Delete actions.
- Create Client application (Web application) interactive with WebAPI using AJAX techniques 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
  - Manage product
  - Add product to cart, manage cart and make 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/update the profile, add product to cart, manage cart and view their orders history.

Guidelines

Activity 01: Build a solution

Step 01. Create a Blank Solution named **Assigment02Solution_StudentCode**

Step 02. The Solution includes includes Class Library Project: **DataAccess, BusinessObject,** and an ASP.NET Core Web API project named **eStoreAPI**

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

Step 01. Add NuGet package related to Entity Framework Core.

Step 02. Using Entity Framework Core Backward/Forward Engineering approach, create business object classes and DB context class with BusinessObject project.

Step 03. Migrate database with SQL Server if using Forward Engineering approach. Create Object Models from Database if using Backward Engineering.

Step 04. Write codes for **DAO.cs, IRepository.cs** and **Repository.cs** (Repeat this step for each business object class in the Project)

Activity 04: Develop eStoreAPI project

Step 01. Create ASP.NET Core Web API project named **eStoreAPI**.

Step 02. Add a reference to **BusinessObject** and **DataAccess** project.

Step 03. Write codes for controllers to perform function requirements .

Step 04. Test Project with Swagger or Postman.

Activity 05: Develop eStore project

Step 01. Create ASP.NET Core MVC project named **eStore**.

Step 02. Use Identity to manage User, manage login authentication process, build application with ASP.NET's account registration, login/logout function, configure Identity lockout to lock user if login fails many times.

Step 03. Ajax request with eStoreAPI.

Step 04. Test Project with Assignment 03 requirements.

Activity 05: Run the Client project and test all actions