Diploma in Multimedia & Infocomm Technology

EGL219 Web Application Development Framework

Project Part 2: Admin and Ordering Modules of the Fruits Ordering System

Objectives:

At the end of part 2 of the project, you will have completed the admin and ordering modules of the Fruits Ordering System (FOS).

Marks to be awarded:

20% of total module marks.

Submission:

1. Ensure that your web application can run in Edge, Firefox or Chrome browser.
2. Zip your web application and database files
3. Name the zip file FOS_P2_adminNo.zip (e.g. FOS_P2_12345A.zip).
4. Demonstrate your working project to your module tutor and submit your work on Blackboard by module tutor the working project by **11 Feb 2022, Friday** (Week 17 second lab session).
5. Late submission will result in the deduction of marks based on the following scale:

Late Submission past due date	Marks deducted
1 Day	20%
2 Days	40%
3 Days	60%
4 Days	80%
> 4 Days	100%

   This means that if you submit your file 5 days after the due date, no marks will be awarded for this part of the project.

6. No marks will be awarded if you do not present your project.

Instructions:

ASP.NET Core Identity and Authentication

1. Continue from Project Part 1 of your FOS web application.
2. Configure the project to use ASP.NET Core Identity.
3. Create the **ApplicationUser** class to store the **Name** of the user.
4. Create a class that will seed the following into the user database :
    a. Roles – "Admin", "Customer"
    b. Users:

UserName	Password	Name	Email	PhoneNumber	Role
maryt	Pa$$w0rd	Mary Tan	marytan@fos.com	61153589	Admin
peterc	Pa$$w0rd	Peter Chew	peterchew@test.com	91158935	Customer
davidp	Pa$$w0rd	David Pau	davidpau@test.com	81158798	Customer

5. UserName field should be unique.
6. Create the user database using data migrations.
7. Create the `Register` and `Login` views. Create links to the 2 views in the navigation bar.
    a. Client and Server-side validation should be done for both views.
    b. Users who register via the `Register` view are automatically given the Customer role.
8. Implement the Logout action method. Create a link to logout in the navigation bar.

Admin Module

1. Do the following for Fruit:
    a. Modify the `AddFruit` view so you are able to insert a Fruit record into the database.
    b. Create an `EditFruit` view so you are able to update a Fruit record in the database.
    c. Implement the `Delete` functionality to remove a Fruit record from the database.
    d. Modify the `Index` view for Fruit to link to the `Edit` and `Delete` functionality for Fruit.
    e. Make sure to catch exceptions and display any error messages.

Ordering Module

1. Add a new table called `Order` into the `FruitsOrderDB`. The table contains the following:

Name	Data Type	Allow Nulls	Foreign Key
Id • Primary key • Auto increment	int	False	
Address	nvarchar(MAX)	False	
ContactNumber	nvarchar(8)	False	
Date	datetime7	False	
TotalPrice	decimal(8,2)	False	
UserName	nvarchar(MAX)	False	

2. Add a new table called `OrderItem` into the `FruitsOrderDB`. The table contains the following:

Name	Data Type	Allow Nulls	Foreign Key
Id • Primary key • Auto increment	int	False	
OrderId	int	False	Id, Order Table
FruitId	int	False	Id, Fruits Table
Quantity	int	False	
SubTotal	decimal(8,2)	False	

3. Create an `OrdersController`.
4. Create a view called `Ordering` for users to make an order.
    a. Each order should allow the user to select 1 to 3 different fruits and their respective quantities.
    b. Calculate the Total Price for the user to review before saving the Order into the database.
    c. Each order and its individual item should be stored in the `Order` and `OrderItem` tables respectively in the database.
    d. Catch any exceptions and display appropriate error messages.
5. Create an `Index` view that will list all the orders of the current user in table form. E.g. If Peter Chew is logged in, the `Index` view will list his fruit orders only.
6. Allow the user to sort the table in Pt 5 according to the Order Date or Total Price.
7. Create a `Details` view that will allow the user to see the details of each Order, i.e. the user is able to see the fruit, quantity and subtotal of each item in the Order.
8. Modify the `Index` view in Pt 5 to link to the `Details` view to each Order.
9. Link to the `Ordering` and `Index` views in the Navigation bar.

Authorization

1. Implement the following Access Matrix:

Views / Roles	Admin	Customer	Anonymous
Home/Index	X	X	X
Register	X	X	X
Login	X	X	X
Logout	X	X	
FruitCategory/Index FruitCategory/SearchFruitCategory	X	X	X
FruitCategory /Add FruitCategory /Edit FruitCategory /Delete	X		
Fruit/Index Fruit/SearchFruit	X	X	X
Fruit/Add Fruit/Edit Fruit/Delete	X		
Orders/Index Orders/Ordering	X	X	

2. If a user is anonymous, i.e. has not logged in, and tries to access the views only available to Admin or Customer users, you will redirect the user to the Login view to be authenticated.
3. If an authenticated user who is not an Admin tries to access the views only available to Admin users, you will display the following view:

Unauthorized Access

You are not authorized to access the page.

Figure 1

Additional Functionality (max 15 marks)

1. Implement at least 2 more functionalities.
2. You can implement **any** functionality as long as it makes sense for the project. Marks are awarded for completeness and complexity of the functionalities.
3. Some examples include:
    a. Implement the Add, Edit and Delete functionality for FruitCategory (10m). Take note to take care of exceptions and display appropriate error messages.
    b. Allow logged in user to change password (3m)
    c. Allow logged in customer to cancel own Orders (3m)
    d. Allow Admin user to edit customer order (5m)
    e. Allow Admin user to search for Orders by Customer name (5m)
    f. Allow logged in user to edit his/her own information (5m)
    g. Allow the customer to apply a promo code that will apply a discount to the total price. Note that the promo code should not be hardcoded. (10m)