300583 Web Systems Development

PRACTICAL SET 3 DUE: 9PM 16 OCT 2020

In this Practical Set, you will implement a nearly full-fledged website for the "Gourmet Pizza" shop with features such as authentication, database access, etc.

Steps for completing this practical set

1. Since we are going to include authentication in this project, you need to create a brand-new ASP.NET Core project called "GourmetPizzaPrac3" with Visual Studio. You should follow the steps in Lecture 9 to:
    a. Include the Identity package in your project;
    b. Install the Sqlite package into your project;
    c. change the database provider to Sqlite;
    d. apply the initial migration;
    e. scaffold the source code of the Identity package into your project.
2. Modify the Home page of your project to introduce the "Gourmet Pizza" shop. To achieve this, you should repeat the Steps 4, 7, and 12-14 from Prac1.
3. Create the following three Model classes: Pizza, Customer and Purchase, and establish the one-to-many relationship between Pizza and Purchase, and the one-to-many relationship between Customer and Purchase with navigation properties.
    a. Pizza: the definition of this class should include those properties in Prac1 with the addition of necessary navigation properties.
    b. Customer: the definition of this class should be similar to that in Prac2 with the following two exceptions:
        i. There is no 'ID' property in this class; instead, the 'Email' property is used as the primary key for this class.
        ii. It should include necessary navigation properties.
    c. Purchase: the definition of this class should include the following properties and their associated requirements:

Property	Data Type	Requirement
ID	int	Primary key;
PizzaID	int	Foreign Key (NB: since the name follows convention, this will be automatically figured out by EF Core)
CustomerEmail	string	Same as above;
PizzaCount	int	The number of pizzas purchased. Should range from 1 to 80.
TotalCost	decimal	The total cost of this purchase. Should range from 1.00 to 1000.00
ThePizza	Pizza	This is a navigation property
TheCustomer	Customer	Same as above

4. Use scaffolding to create the Razor pages for these three Model classes respectively.
5. Use migration to create the database tables for these three Model classes together.
6. Follow Lecture 9 slides to implement a 'My Details' page to allow a logged-in customer to enter and update his/her details. Note that the details should include all properties defined in the Customer model such as given name, family name, data of birth, mobile number, etc.; but the 'Email' property should be excluded as it should be obtained by source code.
7. Follow Lecture 9 slides to redirect a newly-registered customer to the 'My Details' link. In running the project, when a customer is redirected to this page, please enter the details in this page, otherwise the database record for this customer won't be created.
8. Modify the 'Create' Razor page in the 'Purchases' folder to implement a customized page for a logged-in customer to purchase pizzas. Specifically, you should implement the following:
    a. The web form for purchasing pizzas should include the following input devices:
        i. A dropdown list for selecting pizzas. The options in this dropdown list should be read from the database, and each option should use a pizza's ID as its value and a pizza's Name as its text.
        ii. An <input> field for entering PizzaCount. The PizzaCount should range from 1 to 80.

    Note: You can refer to the sample project for Lecture 8 on how to implement this step, but there is a difference: the dropdown list for customers is absent in the web form required by this step, since we know that the customer should be the logged-in user. Thus, you should retrieve the customer's email by calling a method from the Identity package. (See hint in the Step 8.b below.)

    b. If the customer inputs in this web form are valid, insert this purchase into the database. Note that you should do the following before the insertion:
        i. Get the CustomerEmail for this purchase by calling a method from the Identity package. (Hint: see the 'My Details' implementation in the sample project for Lecture 9).
        ii. Calculate the TotalCost for this purchase.
    c. Display a confirmation message for this purchase if it is successful. The message should include the following pieces of info: the name and the number of the pizza ordered, and the total cost. (**Hint:** Refer to the sample project for Lecture 8)

Note: The step 8 allows you to insert purchases into db. Without it, you cannot test the implementation of later steps. To work around this, you are allowed to manually enter purchases records into the Purchase table by Sqlite DB Browser if you have to skip this step.

9. Modify the Index page in the 'Purchases' folder to implement the following:
    a. Only display the purchases made by the logged-in customer. (Hint: this can be done by calling the Where() method with a lambda expression on CustomerEmail).
    b. Remove the 'Edit', 'Details' and 'Delete' links in each row.
    c. There should be a column displaying the name of the pizza purchased.
    d. There should be a column displaying the full name of the customer making this purchase.

e. The three column headings ('Pizza name', 'PizzaCount' and 'TotalCost') should be links that a logged-in customer can click to sort all records by that column. (Note: it is OK to sort 'TotalCost' alphabetically).

f. Clicking a column heading toggles the display order between ascending and descending.

10. Add a Razor page named 'CalcPurchaseStats' under the 'Purchases' folder to calculate and display the following distribution: the number of purchases against each PizzaCount (i.e., how many purchases records contain PizzaCount of 1, 2, 3, …, respectively?). All purchases should be considered. That is, this page is for a customer to learn statistics over all customers; you should not restrict the purchases to those made by the logged-in customer.

    (Hint: refer to Lecture 11)

11. Make the links in the navigation bar dynamic:

    a. When a customer is not logged-in, he/she should see the following links: Home, Pizzas (pointing to /Pizzas/Index), Privacy, Register, Login.

    b. When a customer is logged-in, he/she should see the following links: Home, Pizzas Index (pointing to /Pizzas/Index), My Details, Purchase Pizzas (pointing to /Purchases/Create), My Purchases (pointing to /Purchases/Index), Statistics (pointing to /Purchases/CalcPurchaseStats), Logout. (Note the missing of the last three links from the above.)

    Note: This step has a difference from the sample project for Lecture 9: the link dynamic display is based on whether a user is logged in, not on the user's role.

    Hint: This dynamic display can be implemented by following the idea of Pages/Shared/_LoginPartial.cshtml in the sample project for Lecture 9. You should use this idea to modify the _Layout.cshtml.

12. Use the [Authorize] attribute class to make the following links only accessible to logged-in customers: My Details, Purchase Pizzas, My Purchases, and Statistics.

Checkpoint (2 marks)

The Prac3 Checkpoint will occur in your registered practical class in week 10. At this checkpoint you should complete steps 1-2.

Marking criteria:

- Step 1: 1 mark.
- Step 2: 1 mark.
- Any mistake in a step will results in a zero mark for that step. No decimal mark is available.

Final Submission (16 marks)

Please follow the steps below to submit your completed Prac3 to vUWS:

- Zip up the entire GourmetPizza folder and its sub-folders, which contain all your files for your project.
- Name the ZIP file *Surname_studentid*_Prac3.zip
- Submit the zip file:
    - In VUWS, go to 'Practicals'
    - Click on 'Prac3 Submission'; this link will be made available closer to the deadline.
    - Attach your zip file and submit

Marking Criteria: Please see the rubric for Prac3 in the **My Grades** page on vUWS.

Notes:

- You can submit the zip file multiple times until the due time.
- Work submitted after the due time will incur a 10% per day late penalty as per university policy **https://policies.westernsydney.edu.au/document/view.current.php?id=227**
- **Based on the policy, the 10% above means the 10% of the worth of the assessment, not your final score. So if the worth of the assessment is 10 marks and you scored 5 marks with a 5-day lateness, then you'll receive 0 mark.**