2703ICT - Assignment 2

Due Date: 11:59pm Sunday 29 September 2019 (Week 11)
Weight: 45%
Individual Assignment

Introduction
In assignment 2, we will create a food ordering portal (e.g. Menulog, Deliveroo, Uber Eats). With this portal, restaurants can list the dishes they want to sell. Customers can order selected dishes.

Details
Your implementation must use Laravel's migrations, seeders, models, ORM/Eloquent, route, controllers, validator, and view/blade. You have some freedom in designing your website, however, it must satisfy the following requirements:

1. Users can **register** with this website. When registering, users must provide their name, email, password, and address. Furthermore, users must register as either a:
    a. Restaurant, or
    b. Consumer.
2. Registered users can **login**. Users should be able to login (or get to the login page) from any page. A logged in user should be able to log out.
3. Only the restaurant users can **add dishes** to the list of dishes sold by his/her restaurant. They can also **update and delete** existing dishes. A dish must have a name and a price. A dish name must be unique. A price must be a positive value.
4. All users (including guests) can see a list of registered restaurants. They can click into any restaurant to see the dishes this restaurant sells.
5. The list of dishes should be **paginated** with at most 5 dishes per page.
6. (Single purchase) Only consumers can **purchase** a dish. Since we do not deal with payment gateways in this course, when user clicks on purchase, we simply assume the payment is successful, and save the purchase order in the database. Then it will display the dish purchased, the price, and the delivery address (which is the consumer's address) to let the user know that the purchase is successful.
7. A restaurant (user) can see a **list of orders** customers have placed on his/her restaurant. An order should consist of the name of the consumer, that dish (name) that was ordered, and the date that the order was placed.

The above are the basic requirements, which if implemented well should allow a student to obtain a pass level mark. The following are the more advanced requirements. We recommend students first complete the basic requirements (make a backup copy of your assignment) before attempting the more advanced requirements:

8. After user registration, login, or logout, **appropriate redirections** should be provided. E.g. if user logs in from the details page, then after user logs in, s/he should be redirected back to that page.
9. When restaurant users add a new dish, the dish name must be **unique for that restaurant**, not across restaurants. This is an extension of requirement 3.
10. When restaurant users add a dish, s/he can upload a **photo** for that dish. This photo will be displayed when this dish displayed.
11. In addition to requirement 6 (single purchase), consumers can add multiple dishes to a **cart**, see the contents in the cart, the cost of this cart (the sum of all dishes), remove any unwanted dishes, before purchasing these dishes. Once purchased, the cart will be emptied.
12. There is a page which lists the **top 5 most popular** (most ordered) dishes in the last 30 days.
13. Restaurants can view a **statistic page** which shows the sales statics for that restaurant. This page shows:

   a. The total amount of sales (in dollar value) made by this restaurant.
   b. The weekly sales total (in dollar value) for the last 12 weeks, i.e. there should be a sales total for each of the last 12 weeks.
14. There is another user type called **administrator**. There is only 1 administrator which is created through seeder. The purpose of administrator is to approve new restaurant (users). After a new restaurant user (account) is registered, s/he cannot add/remove dishes from his/her restaurant until this account is approved by the administrator. There is a page where the administrator can go to see a list of new restaurant accounts that require approval, and to approve these accounts.

Server-side input validation must be implemented. For the purpose of this assignment, client-side input validation should NOT be implemented so we can test your server-side validation.

Hint: If you are not able to properly implement user registration of different user types, you can still seed the users, restaurants, and dishes so you can implement other functionalities.

Technical requirements

- Use Laravel's migration for database table creation.
- Use Laravel's seeder to insert default test data into the database. There should be enough initial data to thoroughly test the retrieval, update, and deletion functionalities you have implemented.
- Use Laravel's ORM/Eloquent to perform database operations. Only partial mark will be awarded for implementations using SQL or query builder.
- Proper security measures must be implemented.
- Good coding practice is expected. This includes:
  - Naming: using consistent, readable, and descriptive names for files, functions, variables etc.
  - Readability: correct indenting/spacing of code.
  - Commenting: there should at least be a short description for each function.
  - View: proper use of template and template inheritance.

For further details of the requirements, refer to the marking rubric. **All requirements from both the assignment specification and marking rubric must be satisfied.**

Submission Requirements

You must submit the following items for the assignment:

- Documentations:
  - A **peer review reflection document** which describes your participation in peer review, what you have learnt being the reviewer or reviewee, and what steps have you taken to improve/optimise your peer review experience. Write at most 500 words.
  - An **ER diagram** for the database.
  - A short document describing what you were able to complete, what you were not able to complete, any interesting approaches you took, and any extra that was implemented.

  All above documentations should be provided as a page (or pages) in the website and linked to from the navigation menu. Furthermore, you need to copy and paste your peer review reflection document into the SafeAssign text area.
- A compressed file containing ALL the files in your submission (including all PHP code, SQL to create the database, and documentations as described above).
  - This file must be submitted via Learning@Griffith through the assignment 2 link.

  To reduce the file size, you can delete the *vendor* directory before you compress the files. (The vendor directory can be restored by the command: composer update). You can use zip command to compress your assignment directory (see Lecture 1-3). You can download your zip file through the browser, e.g. if your place your assignment2.zip in the html directory, then simply point your browser at https://s1234567.elf.ict.griffith.edu.au/assignment 2.zip.

Note: You are responsible for regularly backing up your work. Hence, if you lose your file due to not backing up, then expect to be heavily penalised.

Peer Review

Assignments will be marked by your tutor in the lab following the due date of the assignment. Before or at the start of your laboratory class for assignment marking, you must complete Peer Review for your submission.

The peer review activity requires you to demonstrate and explain your work to a fellow student (peer) that have already submitted their assignment (you should check their L@G submission page to make sure of this). Your peer will review your assignment using a print out of the rubric. To indicate the completion of peer review, your peer needs to record the date and time of peer review and sign the rubric (form).

Assignment Demonstration and Marking

After you have completed your peer review, you must demonstrate and explain your work to your tutor in Week 12 lab to have your submission marked by your tutor.

<u>If you don't do *Peer Review* and then *demonstrate your assignment* to your tutor, your submission will be regarded as incomplete, hence you will not receive a mark for this assessment item!</u>

During the demonstration, you need to show the last modified date of your file (on Elf, run the command: *ls -la* in your *routes* directory).