Assignment 1 – 2703ICT Web Application Development

Due Date: 9 September 2024 (Monday Week 8) 9am
Weight: 40%
Individual Assignment

The aim of this assignment is to build a simple web application with the basic building blocks (to a reasonable level). The only support from Laravel framework you can use are routing and templating/view, similar to what we did up to Week 6. The understanding gained from this assignment provides a foundation for students to learn other frameworks or to develop new frameworks. If you use other features such as controller, model, migration/seeder and validation, you will be heavily penalized, and potentially be investigated for misconduct.

In assignment 2, we will let you unleash the full capabilities of Laravel.

Introduction
For this assignment you are required to build the foundation for a review application. This application stores and displays user reviews for an item (e.g. products/services/websites) of your choice. Your website needs to have a theme for a particular type of items, e.g. A review website for mobile phones.

There is NO requirement in Assignment 1 to support user authentication.

Details
The name and design of the website can be to your choosing however it must satisfy the following requirements:

1. All pages must have a **navigation menu**, either across the top of the page or down the left or right column.
2. The **home page** must display a list of all items in the database. Clicking on the item will bring up the review page for that item.
3. The **review page** will display all the details for that item (i.e. name, manufacturer, and any other detail for that item). Furthermore, it will display all reviews for that item. Each review contains the reviewer's name, the rating, date of the review, and the review (text).
4. Users can **create a new item**. New item must have a name and a manufacturer.
5. Users can **create a new review** for an item. To add a review, user enters the username, rating, and the review text (which can be up to a few paragraphs of text). User (with the same name) cannot post multiple reviews for the same item. An item can have multiple reviews.
   Note: when adding a review, the user should not have to enter the date, the date should be taken from the system.
6. **Input validation 1.** A name (user, item, and manufacturer) must have more than 2 characters and cannot have the following symbols: -, _, +, ". Appropriate error message should be displayed on validation error.
7. **Input validation 2.** Any odd number in a username (e.g. 1Smith, Bob21, or Ace1337Nova) will be removed from the name before the name is stored. E.g. 1Smith becomes Smith, Bob21 becomes Bob, Ace1337Nova becomes AceNova. A message should be displayed to inform user that the name they have entered has been changed, and display the altered name.
8. Users can **edit existing reviews**.
9. An **item** can be **deleted**. When a user deletes an item, the reviews for that item should also be deleted.
10. When listing all items in the home page, the **number of reviews** and **the average rating** for that item should also be displayed.
11. The home page has a feature where user can select to **sort the list** of items in the follow ways:

    a. By the number of reviews
    b. By the average rating

   User can select to sort by either ascending or descending order.
12. There is a page that **lists all manufacturer** and displays the average rating for that manufacturer. The average rating for a manufacturer is the average of the average ratings of all items from that manufacturer. Clicking on a manufacturer will list all items and average rating of each item from that manufacturer. Clicking on the item will bring up the review page for that item.
13. After the user submitted a review, the system **remembers that user's name** for the duration of the session. Subsequent reviews made will not require the user to enter his/her name, but instead use the same username as the first post/comment. Note: this feature requires you to learn to use the session feature of PHP/Laravel.
14. There is a feature that identifies or help users **identify fake reviews**. You have to research into methods for identifying or support the identification of fake reviews and implement the method(s). You will be judged on the usefulness and usability, creativity/innovation, and also technical competence of your design and implementation. Only the best few will receive full mark.

Technical requirements

- This assignment must be implemented using Laravel, however, you are only allowed to use the following feature of Laravel: routing, and view. Some specific implications are:
    a. Database access should be implemented via raw **SQL** and executed through Laravel's DB class's methods insert(), select(), update(), delete(), and raw(). E.g. `DB::select("select * from item");`
       You are NOT to use Laravel's or ORM (e.g. `Product::where('price', '>', 800)->get();`). ORM will be used for assignment 2.
       An SQL file should be used to create tables and insert initial data. There should be enough initial data to thoroughly test the retrieval, update, and deletion functionalities you have implemented.
    b. You must **implement your own validation check**, like Week 3 exercise (you must not use Laravel's validation feature – which we will use for Assignment 2). Validation errors message must be displayed within the view. Note: validation must be done on the server side (you should know why). To demonstrate this, you should not have any client-side validation.
- Proper **security measures** must be implemented, e.g. perform HTML and SQL sanitisation etc. You should be able to explain the security measures you have implemented.
- **Template inheritance** must be properly used.
- **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.

Documentation

Provide the following documentation in no more than 1 page:

1. An ER diagram for the database.
2. Reflect on the process you have applied to develop your solution (e.g. how did you get started, did you do any planning, how often do you test your code, how did you solve the problems you come across). What changes would you make for assignment 2 to improve your process?
3. If you have completed the fake review detection requirement, explain how you have performed your detection.

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

Self-assessment
You need to perform a self-assessment by marking your work against the self-assessment rubric. You will be assessed on the accuracy of your self-assessment.

Submission Requirements
Your submission should consist of:

1. A compressed file containing ALL source files in your submission (including all PHP code), but **excludes the vendor directory**.
   Note: Delete the *vendor* directory before you compress the files. (Restore the vendor directory before your demonstration with the command: *composer update*). Use the *zip* command to compress your assignment directory (see Lecture 1-3).
1. A PDF file containing your documentations.
2. A DOC file containing your self-assessment rubric.

These files must be submitted via Learning@Griffith through the assignment 1 link.

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.

Do not update your source code after you have submitted your work. 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).

Assignment Demonstration and Marking
You must demonstrate and explain your work to your teacher in Week 8 lab to have your submission marked by your teacher and receive personalised feedback. If you do not demonstrate and explain your work, then your submission will be regarded as incomplete and will not be marked.

<u>If you do not *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).

<u>Important:</u> On-campus students, you will need to bring a printout of your self-assessment rubric you have submitted to your week 8 lab class.

Reference and the Use of AI
If you have used code snippets from the Internet, you must reference your source. In this case, you do not receive credit for writing the code, but you'll receive some credit for integrating the code (i.e. you'll receive partial marks).

In this course we do not discourage the use of AI/Chat Bot. You are free to use AI to help with your assignment. The referencing requirement for the use of AI is that you copy and paste (or screen shot) your question/prompt and AI's answer into the reference section of your submission. Or provide a link to the chat log (by using the Share Chat feature of Chat GPT).

Warning: We take student academic misconduct very seriously!