2703ICT - Assignment 2

Due Date: 5pm Friday 4 October 2024 (Week 11)
Weight: 50%
Individual Assignment

Introduction
In assignment 2, you will create a web application for students to share and submit their peer reviews for a course. Note: the peer review process you are required to implement for this assignment is not the same as the peer review process used for this course, it's simpler.

Terminology
Review submitted: review submitted by the reviewer, written for their peer/reviewee.
Review received: review received by the reviewee, written by their reviewer.

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

1. There are two types of users of this system:
    a. Teachers, and
    b. Students.
   Students need to **register** before they can login. Students need to supply their name, email, and s-number when they register. Teachers are simply seeded into the database. For testing purpose, there should be sufficient seeded teachers and students.
2. All users need to **login** before they can access the functionalities of this system. Users login with their s-number and password. Once a user logs in, their name and user type (teacher or student) will be displayed at the top of every page.
3. A logged in user should be able to **log out**.
4. **Home page**: once logged in the user can see a list of courses they are enrolled in or teaching. As per real-world, a student can enrol into multiple courses, and a teacher can teach multiple courses. A course has a course code and name. Clicking on a course brings the user to course details page.
5. **Details page for a course** displays the teachers and the list of (peer review) assessments for this course (e.g. Week 1 Peer Review). Next to each assessment, it should display its due date. User can click on an assessment to bring them to the details page for that assessment (see below).
6. Teacher can **manually enrol** a registered student into a course.
7. From the course details page, a teacher can **add a peer review assessment** to this course. A peer review assessment should contain an assessment title (up to 20 characters), instruction (free text), the number of reviews required to be submitted (a number 1 or above), maximum score (between 1 and 100 inclusive), a due date and time, and a type. There are two types of peer review: student-select and teacher-assign (reviewee).
8. Teachers are allowed to **update a peer review assessment**, unless there has already been a submission. When updating, the old value should be shown.
9. The **details page for an assessment for student** has the following functionalities:
    a. Allows students to **submit** their peer **review**. This page displays the assessment title, instruction, number of required reviews, submitted review, and due date. For peer reviews that are of "student-select" type, students select their reviewees (from a drop-down menu of all students in this course) and enter their review text (free text with at least 5 words). Students need to be able to submit the required number of reviews for this assessment. The system needs to ensure each review submitted must be for a different reviewee.

b. **Displays the peer reviews** received by this student for this assessment, including the name of the reviewer.

10. The **details page for an assessment for teacher** can only be accessed by teachers of this course for marking. This page will list all students in the course, and for each student it shows how many reviews this student has submitted and received, and the score for this assessment. Clicking on a student will show a page containing all the reviews submitted and received by this student. Teacher can then assign a score to this student for this assessment.
11. There is **pagination** in the marking page that lists all students. A page is limited to 10 students. You need to have sufficient initial data to show your pagination works.
12. A teacher can **upload** a text file containing a course information, which includes the course name, teachers, assessments, and the student enrolled in this course. Uploading this file will result in a new course a new course being created in the system with the supplied assessment, teachers, and students/enrolments. You are free to define the file format (keep it simple) and what data should be in this file to achieve this requirement. The system should check that the course, with the same course code, does not already exist. It is possible for students in the list have yet to register with this system.
    For your demonstration, you should have a file prepared to show this feature works.
13. Design a feature to **encourage reviewers to submit useful reviews** for their reviewee without requiring reviews be marked. A base solution would be for reviewees to rate the reviewer (say out of 5), then there is a page all users can access that lists the 5 reviewers with the highest average rating. To achieve full mark for this requirement, you need to come up with a more innovative working solution and convincing argument why your solution is better at encouraging good review.
14. There are two types of peer review, "student-select" as described in 9a, and "teacher-assign". In this requirement you need to design and implement **teacher-assign**. With Teacher-assign the teacher (randomly) assigns students to peer review group for each peer review. Students only review other students in their assigned group. This web application should support the following:
    - Be able to obtain the correct information regarding which student has been assigned to which group.
    - Students can only submit reviews for students in the assigned group.

    This requirement is modelled after this course's peer review process. You can take into account that the peer review happens in a workshop, we know who's enrolled in which workshop, but not all students turn up to the workshop (they are enrolled in).

    We are looking for a practical solution that we can adopt for real-world use. A main focus should be on the ease of use for the teacher and the student. The teacher would not like to manually enter each student/group into the system. But if this is the only solution, then it has to be as efficient as possible. The design needs to work for large number of students, e.g. is a dropdown menu usable when there are 150 students to select from? There are online and on-campus classes. Also, we cannot trust students to always enter (type in) the correct name of their reviewee.

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 all the functionalities you have implemented. There should be at least 5 teacher, 5 courses, 5 assessments, and 50 students (with course enrolments).
- Use Laravel's **ORM/Eloquent** to perform database operations. Only partial mark will be awarded for implementations using SQL or query builder.
- Proper **input validation** for all inputs. You must NOT implement any client side (html/Javascript) validation, so your server-side validation can be tested.
- Proper **security measures** must be implemented.

- For requirements 13 and 14, you are allowed to use client-side scripts (i.e. Javascript) if it helps with your solution.
- **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.

There will be deductions for not meeting the technical requirements.

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

Documentation

Provide the following documentation in no more than 1 page (excluding appendix):

1. An **ER diagram** for the database. Note: many-to-many relationships must be broken down into one-to-many.
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).
3. Describe, explain and justify the design of your solutions for **requirements 13 and 14**.
4. In the appendix section, include the **screen shot** of your home page, and any screen shots to help you explain your requirements 13 and 14.

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

You must submit the following items for the assignment:

- A compressed file containing ALL source files in your submission (including all PHP code), but **excludes the vendor and node_modules directories**.
  - This file must be submitted via the LMS through the assignment 2 link.
    Note: Delete the *vendor* and *node_modules* directories before you compress the files. (Restore the vendor directory before your demonstration with the command: *composer update*. The node_modules directory can be restored by the command: *npm install*). Use the *zip* command to compress your assignment directory (see Lecture 1-3).
- A PDF file containing your documentations.
- A DOC file containing your self-assessment rubric.

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.

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 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).

Reference and the Use of AI
If you have used code snippets from the Internet, you need to reference your source.

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!