Assignment 1: PHP

Due No due date **Points** 100

COMP284 Scripting Languages (2020-21) -- Assignment 1: PHP

Your task for this assignment consists of two parts:

1. Develop a web-based application using PHP and MySQL that provides the functionality stated in the **Requirements section** below.
2. Make the application that you have created accessible and usable via the URL `https://student.csc.liv.ac.uk/~<your user name>/tutorials.php` taking care of the requirements set out in **Submission and Setup section** below.

Requirements

It is likely that at some point in the future it will again be possible to have on-campus tutorial sessions, but that social distancing rules would still need to observed. This means tutorial capacity will be much more limited. It might also be the case that students would still be allowed to take part in such tutorial sessions remotely and when doing so would not need to be allocated a place in a tutorial session. Furthermore, a student's situation may suddenly change, so it makes little sense to allocate students to particular tutorial sessions for a whole semester.

Our department would like to put a web-based application in place that allows each student to choose their own tutorial sessions for each module for the coming week. The application will be open only on Saturdays and Sundays for bookings for the next week and all data will be wiped on Friday night before the application opens again for the next week. Neither the restriction to Saturdays and Sundays nor the wiping of data is something that you should implement. These restrictions / operations happen outside your part of the application.

Obviously, there will still be a fixed number of tutorial sessions with each tutorial session taking place at a particular time during the week in a particular room, both determined by the Timetabling Team. Each tutorial session will also only be able to accommodate a certain number of students, in the following we will call this the *capacity* of the tutorial session. Initially, the *number of (free) places* on a tutorial session at a specific time is equal to its *capacity* and is then reduced by one for each student who books that tutorial session. Once the number of free places reaches zero, the tutorial session is full at that time and no further student can book it. We keep the range and capacity of tutorial sessions small. To further simplify things, we assume that students do not have to authenticate themselves to the application but instead enter a few personal details when they book a tutorial session.

In more detail, the application deals with the following tutorial sessions at the indicated times and all with the indicated capacity independent of the time at which the tutorial session is offered:

Module	Times and Location	Capacity
COMP315	Thursday, 11:00, REN-LT2; Thursday, 16:00, REN-LT4	4
COMP318	Thursday, 11:00, REN-LT3; Friday, 11:00, REN-LT3	3
COMP326	Tuesday, 9:00, REN-LT1; Thursday, 13:00, REN-LT1	2

That is, in total there are 6 tutorial sessions associated with three different modules, each with a capacity for only two to four students. For instance, four students can take the COMP315 tutorial session on Thursday, 11:00, in REN-LT2, and another four can take the COMP315 tutorial session on Thursday, 16:00, in REN-LT4.

The web-based application should allow a user

1. select a module via a drop-down menu;
2. select a time at which a tutorial session for that module takes place via a **separate** drop-down menu;
3. enter their name via a text field;
4. enter their e-mail address via a second text field (to uniquely identify students and to be able to contact them in case a tutorial session is cancelled);
5. submit a booking request by pressing a `Submit' button, after selecting/entering the data above.

The menus should be populated with data from a database.

On submission of a booking request, a student should be shown a confirmation whether the booking request has been successful or unsuccessful. This confirmation should include all the details of the booking that was attempted. This is subject to the following conditions:

- The application should ensure that the string entered as a name satisfies the following constraints: A name only consist of letters (a-z and A-Z), hyphens, apostrophes and spaces; contains no sequence of two or more of the characters hyphen and apostrophe; starts with a letter or an apostrophe; and does not end in a hyphen. If these constraints are satisfied, then we call the name valid. If these constraints are not satisfied, then the application should display an error message and the booking request must be unsuccessful. This must be realised using PHP (not HTML5 nor JavaScript).
- The application should ensure that the string entered by the user as an e-mail address has exactly one occurrence of @ that is preceded and followed by a non-empty sequence of the characters a-z, A-Z, 0-9, dot, where neither sequence ends in a dot. An e-mail address that satisfies these constraints is called *valid* and the check that the user has entered a valid e-mail address must be performed using PHP only. If the input of the user is not a valid e-mail address, then an error message should be shown and the booking request is unsuccessful.
- If name and e-mail address are valid, then a booking request must be successful if the selected tutorial session still has at least one place left. On success, the number of places on the selected tutorial session is reduced by one and a record of the booking will be kept in the database, including the module, time, name, and e-mail address.
- A booking request must be unsuccessful if there are no places left on the selected tutorial session.

Underlying the application must be a MySQL database. Initially, the database must contain information on the modules, times at which tutorial sessions for each module are offered and the capacity of each tutorial sessions. The database should then keep track of the number of free places after each successful booking requests and also keep records of each successful booking, including all the data that the user has provided. The **PHP Data Objects** (PDO) extension of PHP must be used to implement the interactions between your application and the database.

Additional requirements and comments:

- Ideally, the whole application is realised by a single PHP script. The application does not need to follow the Post/Redirect/Get pattern.
- The description of the application above suggests that a user goes through a sequence of five steps in order to submit a booking request. However, if you use a single page design for your application, then there is little that prevents a user from skipping a step or skipping all steps before clicking on the `Submit' button. Your application should produce appropriate error messages if the booking request does not contain all the expected data. But no error messages should be shown before activation of the `Submit' button.
- The application should help the user by only listing in the first menu the modules that have tutorial sessions with free places, and once a student has selected a particular module in that menu, should in the second menu only list the times at which the tutorial sessions with free places take place.

- Entries in the two menus should be listed in a `sensible' order, that is, modules should be ordered alphabetically and the times chronologically. Also, each module should only occur at most once in the first menu and each time at most once in the second menu.
- If a booking request is unsuccessful, then it should then be possible for a student to make another attempt with as little effort as possible. For example, if name and e-mail address were valid, but there were no places left on the selected tutorial session, then the application should pre-populate the text fields for name and email address with valid information the user has previously enter. (invalid names and email addresses should not be carried forward).
- You should expect that all `inputs' to the application, even those that are intended to come from a menu, will be used by malicious users to inject code that causes your application to malfunction or reveal the contents of the database. You should program the application in a way that safeguards against such attacks.
- It is possible that two student nearly simultaneously try to book the last remaining place on a tutorial session. Depending on how you implement the interaction with the database, this could lead to a situation in which your application determines for both students that there is still a free place left and then records for both students that he/she has successfully booked that place. This is obviously an incorrect behaviour by the application and needs to be avoided. The booking request of exactly one of the two students must be successful and the booking request of the other has to be unsuccessful.
- There could be more students than the combined capacity of all the tutorials. If someone tries to use the application once all places on the tutorial sessions have already been booked, then the application should produce a message indicating that all tutorials are full instead of showing the various menus and text fields. The same should also happen if this situation occurs when a student re-enters the application after an unsuccessful booking request.
- As this is an assignment on PHP, the use of other scripting languages should be kept to a minimum. In particular, the use of JavaScript must be restricted to the JavaScript `submit()` function. User-defined JavaScript functions are not allowed.
- Use of the PHP superglobal `$_GLOBALS` is poor programming practice and its use will results in a lower mark.
- Your code should follow the **COMP284 Coding Standard (https://cgi.csc.liv.ac.uk/~ullrich/COMP284/notes/COMP284CodingStandard.pdf)** . This includes pointing out which parts of your code have been developed with the help of on-line sources or textbooks and references for these sources.

Each of these requirements is linked to one of more of the assessment criteria for this assignment. Therefore, the more requirements an application satisfies, the higher the mark.

Submission and Setup

Submit your HTML/PHP file and your MySQL database dump via the departmental submission application at **https://sam.csc.liv.ac.uk/COMP/Submissions.pl?module=comp284 (https://sam.csc.liv.ac.uk/COMP/Submissions.pl?module=comp284)** (COMP284-1: PHP).

Do not forget to also set up the database on the departmental MySQL server, to make `tutorials.php` accessible via the departmental web server, and to correctly connect the two.

The files submitted must be identical to those set up on the departmental web server. Furthermore, no alterations are allowed to the latter after files have been submitted. If a submitted file and the corresponding file on the departmental web server have different timestamps, then the later timestamp will be used to determine lateness. This applies even if the earlier file is used for marking.

Permissions of the files in your filestore must be such that no other user can view their contents in the filestore.

Deadline

The deadline for this assignment is

To be confirmed

Earlier submission is possible, but any submission after the deadline attracts the standard lateness penalties. Please remember that a strict interpretation of `lateness' is applied by the Department, that is, a submission a minute after the deadline is considered to be a day late (analogously for submissions that are delayed further).

Assessment

This assignment addresses the following learning outcome of the module:

- Develop server-side web-based applications using an appropriate scripting language, with emphasis on concurrent use of such applications.

This assignment contributes **50%** to the overall mark of COMP284. Failure on this assignment may be compensated by higher marks on other assignments for this module.

Marks will be awarded according to the following scheme:

- Submission, Setup, Error-freeness: 10
- Input/Output handling: 45
- Database and database transactions: 24
- Correctness of processing booking requests: 9
- Code layout, commenting, and quality of code: 12

In more detail, the requirements above translate into about 35 criteria that your application and its underlying code / database must satisfy. Marks are given according to the extent to which the application is observed to behave in the expected way and produces correct results, and, to a lesser extent, how well the code is written. Code that has no observable effect will typically receive no marks.

The mark for a submission that is not set up correctly on the departmental web server and MySQL server will be capped at 22. That is, only the first and the last item of the marking scheme will be assessed.

As stated above, the University policy on late submissions applies to this assignment, as do the University policy on coursework submission (available at **https://www.liverpool.ac.uk/media/livacuk/tqsd/code-of-practice-on-assessment/appendix_Q_cop_assess.pdf (https://www.liverpool.ac.uk/media/livacuk/tqsd/code-of-practice-on-assessment/appendix_Q_cop_assess.pdf)**) and the University policy on academic integrity (available at **http://www.liv.ac.uk/student-administration/student-administration-centre/policies-procedures/academic-integrity/ (http://www.liv.ac.uk/student-administration/student-administration-centre/policies-procedures/academic-integrity/)**). You should follow the **COMP284 Lab Rules (https://cgi.csc.liv.ac.uk/~ullrich/COMP284/notes/COMP284LabRules.pdf)** to ensure that you do not breach the latter policy.

Feedback

You can expect individual feedback for this assignment about three weeks after the deadline. Generic feedback will be provided 7 days after the deadline. You should take that to be the date referred to in Section 5.1.4.(i) of the UG Computer Science Student Handbook.