Assignment of 300582 Technologies for Web Applications

A WEBSITE FOR WESTERN SYDNEY HOTEL

Checkpoint Due: During Practical Class in Week 11

Submission Due: 11pm Wed 01/06/2016

General Requirements

- Read this entire document first. If you are not able to implement all of the functionality required in this document, you should attempt the functionality in an easy-to-hard order based upon your skills. This will help you achieve most marks from this assignment.
- All stylings must be done by external CSS. You can use as many external css files as appropriate. The only exception to using external CSS is that, when you emphasize a phrase, you can use or tag.
- For the HTML forms, you should choose the right input devices for the data required. All input devices should have appropriate labels.
- For client-side input validations, only the enforcement of mandatory fields may be implemented by HTML5 attribute 'required'; all other types of validations must be implemented by javascript. You may use either internal or external javascript code. The external JS file name is of your own choice.
- For server-side scripting, if user input is used to compose a database query, it should be sanitized either by calling the mysqli_real_escape_string() method or by binding to a prepared statement.
- It is your responsibility to ensure no inappropriate images or content are used in your assignment.

1. Assignment Overview

This assignment is to build a simple website for the Western Sydney Hotel (WSH). The main functionality of the website includes: allowing customers to register, login, search and book rooms, and allowing hotel administrators to browse any table in the database, and change room prices.

2. Database description

It is important to understand the design of the database for the website before creating individual html or php pages. Take some time to review the details in this section before proceeding with website page implementation.

2.1 Database tables

The WSH uses a MySQL database named 'westernhotel' to store the information about it's rooms, customers, administrators, and room bookings. A pre-populated copy of the westernhotel database is available to each TWA student. Subsection 2.2 of this document describes how to connect to your database, and also provides a PHP page allowing you to examine the data initially in your database. Note that details on data types supported in MySQL can be obtained from http://www.w3schools.com/sql/sql_datatypes.asp.

Specifically, the 'westernhotel' database consists of four tables: **rooms**, **customers**, **administrators** and **bookings**.

rooms

Table description: records all the rooms in the WSH. The WSH has only one 4-story building. On each floor, there are a certain number of 1-bed rooms, 2-bed rooms, and 3-bed rooms. Each room is uniquely

identified by a Room ID. The Room ID has the following format: FNN, where F indicates the floor and can be 'G', '1', '2', or '3', and NN consists of two digits that indicate the room number on a floor. For example, the room ID '121' means a room with number 21 on the first floor.

Table notes: This table is already populated in the 'westernhotel' database. Only the price field can be updated.

Field Name	Data Type	Description	Key?
rid	CHAR(3)	The room ID	primary
level	CHAR(1)	The level of this room, 'G', '1', '2', or '3'	
beds	TINYINT	the number of beds in the room, 1, 2, or 3	
orientation	CHAR(1)	'N', 'S', 'W', 'E' means facing north, south, west and east respectively	
price	FLOAT(6,2)	the price per night	

customers

Table description: records all customers registered at the website.

Table notes: There is initially one customer with username 'test' and password '123456' in this table. You can use it to test your login page. You are expected to insert more records into this table through the registration page you implement for this website.

Field Name	Data Type	Description	Key?
username	VARCHAR(20)	The user name of this customer	primary
password	VARCHAR(20)	The password	
gname	VARCHAR(20)	the given name	
sname	VARCHAR(20)	the surname	
address	VARCHAR(40)	the street address	
state	VARCHAR(20)	should be one of the states or territories in Australia	
postcode	CHAR(4)	4 digits, we assume customers are only from Australia in this assignment	
mobile	CHAR(10)	mobile number; 10 digits	
email	VARCHAR(40)	email address	

administrators

Table description: records all administrators of WSH.

Table notes: This table is already populated with two records: username 'bob' with password 'bobbob' and username 'alice' with password 'alicealice'. You should use one of these records to test your administrators login. You are not expected to change this table in any way in this assignment.

Field Name	Data Type	Description	Key?
username	VARCHAR(20)	The user name of this admin	primary
password	VARCHAR(20)	The password	
gname	VARCHAR(20)	the given name	
sname	VARCHAR(20)	the surname	

bookings

Table description: records all bookings for customers.

Table notes: There is one booking record in this table initially. It is booked by the customer 'test' for the room 'G01' with check-in date '2016-11-1' and check-out date '2016-11-3'. Your web application will write more records to this table for each booking made by a customer. Note that each booking can only consist of one room. If multiple rooms are needed by a customer then the customer will need to make multiple bookings.

Field Name	Data Type	Description	Key?
bid	INT	The booking id, with AUTO_INCREMENT set	primary
rid	CHAR(3)	the ID of the room booked	foreign
username	VARCHAR(20)	The username of the customer who booked this room	foreign
checkin	DATE	the day the customer starts to occupy the room	
checkout	DATE	the day the customer leaves; the customer won't occupy the room on this day.	
cost	FLOAT(8,2)	the total cost of this booking	
btime	TIMPSTAMP	the time of the booking, with ON UPDATE CURRENT_TIMESTAMP set; In an INSERT query, this field will automatically set itself to the current date and time. So you can omit this field in your INSERT query.	

As a final note, all the fields in the above four tables are set with the NOT NULL property, which means when you insert a record into one of these tables, the values for all the fields of that record must be present in your 'insert' query to make it successful. The only exception to the above is that, you can omit the value of a field, if that field can be automatically initialized by the database (e.g., a field with the AUTO_INCREMENT property or the ON_UPDATE_CURRENT_TIMESTAMP property as aforementioned).

2.2 Connecting to the Database

Each student has their own copy of the westernhotel database. The name of the database, and authentication details are derived from your TWA website credentials as follows:

- The **name** of your database is westernhotel### where ### is your TWA website number
- The database **username** is your TWA website ID
- The database **password** is twa###XX, where the ### is your TWA website number and XX are the first two characters of your TWA website password.

For example for website twa999:

TWA website:	twa999	**Database Name:**	westernhotel999
TWA username:	twa999	**Database username:**	twa999
TWA password:	abcd7890	**Database password:**	twa999ab

And hence the code to connect to **twa999**'s database would be:

```
<?php
$connection = mysqli_connect ('localhost', 'twa999', 'twa999ab', 'westernhotel999');
if ( !$connection ) {
   die("Connection failed: " . mysqli_connect_error());
```

```
}
?>
```

Moreover, to examine the data in your database initially, we provide you with a PHP file named **listdata.php**, which lists the data of all your database tables in one web page. To run **listdata.php**, you need to replace the database username, database password and database name in this file with your own. Your copy of the database and the listdata.php file will be made available by the end of Week 5 in vUWS.

3. Site-wide requirements

3.1 Website design/layout

The layout of your website is open to your creativity but you may use freely available layout templates if you wish. Hint: you can look at http://www.w3schools.com/html/html_layout.asp to find a simple layout template; or you can use frameworks such as W3.CSS or Bootstrap to give your site a responsive design, but this is not required. You may use as many external css files as appropriate.

3.2 Navigation

Each page should have a navigation section, with the same 'look and feel', although the content will vary. This navigation section on each page can be either horizontal at the top of the page, or vertical on the left.

The links contained in the navigation section should be dynamic. The detailed requirements are as follows:

- The links for unlogged-in customers/administrators should include the following: Home, Customer Registration, Customer Login, and Administrator Login.
- The links for logged-in customers should include the following: Home, Search Rooms, Book Rooms, and Logout.
- The links for logged-in administrators should include the following: Home, Change Pricing, Browse Database Tables, and Logout.

Hint: Since the links in the navigation section are dynamic, you should use PHP code to generate the links in this section. One way to do this is: code a PHP file (say, named **generate-nav.php**) for this purpose, and then use the PHP 'include' function to include it in every webpage you created for this website. The **generate-nav.php** would need to: check the session variables which are initialized when customers or administrators login, and then decide which links are generated in the navigation section.

If the above hint is too difficult for you, you can implement static links (meaning all links are present in the navigation section) to make your website at least functioning, but you'll lose marks for this. For the details of marking, please refer to the rubric for this assignment.

3.3 Protecting pages only available to logged-in customers/administrators

Some PHP pages detailed later should only be available to logged-in customers or administrators. To guarantee this, session variables for remembering the identity and the role of the logged-in users should be introduced when customers or administrators login to this website. Then in the PHP pages that are only available to logged-in customers/administrators, these session variables should be checked to prevent unauthorized access.

3.4 Homepage — index.html or index.php

The homepage should contain the aforementioned navigation section, and also contain a main section. If you can achieve dynamic links in the navigation section, implement this page as a PHP file and name it **index.php**; otherwise, you can use the static link approach and implement this page as an html page and name it **index.html**.

In the main section of this homepage, you should welcome the customers, briefly introduce the hotel, and briefly explain the usage of this website.

4. Pages for Customers

As every page will contain the navigation section, we will no longer mention the navigation section below, but focus on the actual functionality of each page.

4.1 Registration page — register.php

This page allows customers to register at this website. This page should display a form with input devices for entering the following information:

1. Username (at most 20 characters long, should be unique in the database and hence your php code needs to check if it already exists in the database before creating the customer record)
2. Password (between 6 and 20 characters long)
3. Retype Password (must be same as the Password)
4. Given name (at most 20 characters long, can only contain alphabetical letters, hyphen, apostrophe, and space)
5. Family name (at most 20 characters long, can only contain alphabetical letters, hyphen, apostrophe, and space)
6. Address (at most 40 characters long)
7. State (a dropdown list including all 8 states and territories of Australia should be used)
8. Postcode (a 4-digit string)
9. Mobile number (a string in the following format: 04ddddddddd, where 'd' is a digit.)
10. Email (at most 40-character long, contains only one '@', and at least one '.' after '@')

All of the above input fields are mandatory, and other validation requirements are included inside the parentheses after each input field. You should implement these validations **both** client-side **and** server-side. The only exception is, for the uniqueness of the username, only implement this check in the server-side to ensure that the chosen username does not already exist in the database. **Note:** To reduce duplicate testing of your skills, this is the only form we require both client-side and server-side validations. For all later forms in this assignment, we only require server-side validations. However, you should bear in mind that, in real-world applications, both client-side and server-side validations must be present.

This form should submit to itself using POSTBACK. If there are validation errors, meaningful error messages should be displayed to the customer. Upon successful validation of the entire form, a new record should be inserted into the **customers** table in the 'westernhotel' database, and the customer should be redirected to the login page.

4.2 Login page — customerlogin.php

In this page, a form with the following input devices should be presented:

1. Text box for username
2. Password box for password
3. Submit and reset buttons

The page should submit to itself using POSTBACK. Server-side validations should be implemented to guarantee fields 1 and 2 are not empty. Upon successful validation, the page should look up the **customers** table in the database to compare the username and password supplied by the customer. If both username and password are correct, redirect customers to the **search.php** page; otherwise, display the login form again and report the error to the customer.

4.3 Search rooms — search.php

This page allows logged-in customers to search available hotel rooms by presenting a form with the following input devices:

1. The number of beds required (should be implemented as a dropdown list or radio buttons, consisting of static options for 1 bed, 2 beds and 3 beds)
2. The orientation of the room (should be implemented as a dropdown list or radio buttons, consisting of static options for north, south, west and east)
3. Check-in date (please use <input type="text" …>, since <input type="date" …> is not supported by all browsers yet)
4. Check-out date (the same as field 3)
5. Submit button

Input fields 1, 3 and 4 on this form are mandatory, and field 2 is optional. If field 2 is not selected, it means all orientations are acceptable. Furthermore, for fields 3 and 4, you should validate that:

- They are in the format of 'YYYY-MM-DD', and a valid date in the calendar (e.g., April has 30 days not 31 days, etc.)
- Field 3 is not before the current system date
- Field 4 is after the date of field 3.

All aforementioned validations only need to be done in the server-side.

Once the form submission is successfully validated, you should construct an SQL query to search the available rooms. While this query is a little hard to compose, we provide the following hints for you, in which we suppose the PHP variable `$checkin` stores the check-in date and `$checkout` stores the check-out date retrieved from the customer's input:

- A room can have multiple booking records in the **bookings** table, and if one of its records overlaps with the period `[$checkin, $checkout)`, it becomes unavailable. Note here '`[`' means 'including `$checkin`', and '`)`' means 'excluding `$checkout`', since we allow a customer to check in on another customer's check-out date.
- To test whether the search period overlaps with a booking record, you can use the following observation: "the search period overlaps with a booking record" is equivalent to "`$checkin` is before the check-out date of this booking record and meanwhile `$checkout` is after the check-in date of this booking record".

After the SQL query returns, you should display the results in an HTML table underneath the 'Submit' button. This HTML table should contain the following columns from the **rooms** database table: rid, beds, orientation and price.

For simplicity, you are not asked to implement the functionality that a customer can select a room from the above list and book it. Instead, the customers should record their preferred room somehow (e.g., write it down) and then click the 'book rooms' link in this website to book it.

4.4 Book rooms — book.php

This page allows logged-in customers to book a hotel room by entering the following information:

1. The room ID (should be implemented as a dropdown list, and the room IDs in this dropdown list should come from an SQL query to the **rooms** table).
2. Check-in date
3. Check-out date
4. Submit button

All of the above fields are mandatory. Also, the validation rules for the fields 2 and 3 are the same as those in the previous subsection. All these validations only need to be done in the server-side.

After the form submission is successfully validated, you should still need to check whether the submitted booking is available. Please construct an SQL query to do this. Note that some hints for this are already given in the previous subsection.

If this booking is unavailable, you should inform the customer about this and display the above form

again. If available, you should insert a record about this booking into the **bookings** table. Note that, we have set the 'bid' field with the 'AUTO_INCREMENT' property and the 'btime' field with the 'ON UPDATE CURRENT_TIMESTAMP' property when we set up the database, so you should omit these two columns when constructing your 'insert' query. An example 'insert' query should roughly look like below:

```
INSERT INTO bookings (rid, username, checkin, checkout, cost)
VALUES ($rid, $username, $checkin, $checkout, $cost);
```

If the 'insert' query is successful, you should inform the customer with the following information: the room booked, the check-in and check-out dates, and the cost of this booking.

4.5 Logout page — logout.php

This page is shared by both customer logout and administrator logout. This page should not display anything (i.e., it will not have any HMTL elements). The session should be destroyed, and then the customer or administrator should be redirected to their respective login page.

Hint: To determine if the logout is for a customer or an administrator, a session variable that was created in the customerlogin.php or adminlogin.php can be investigated.

5. Pages for Administrators

You will not be asked to implement a registration page for administrators, since their records should be entered into the database via an internal channel. In the 'westernhotel' database, we have created two administrators for you: 'bob', whose password is 'bobbob', and 'alice', whose password is 'alicealice'. You should use these two admin accounts to login to access the pages for the administrators, and should neither modify these two accounts nor add new accounts.

5.1 Login page — adminlogin.php

This page allows administrators to login by presenting a form with the following input devices:

1. Text box for username
2. Password box for password
3. Submit and reset buttons

The form should submit to itself using POSTBACK. Server-side validations should be implemented to guarantee fields 1 and 2 are not empty. Upon successful validation, the page should look up the **administrators** table in the database to compare username and password. If both username and password are correct, redirect the administrator to the **browse.php** page; otherwise, display the login form again and report the error to the administrator.

5.2 Browse the westernhotel database — browse.php and showtable.php

These two pages allow logged-in administrators to browse and show database tables. The browse.php page should display a form with the following input devices:

- A group of radio buttons which includes the name of every table in the 'westernhotel' database. The tables in the group should come from an SQL query (i.e., "show tables") to the database.
- A Submit button

The method of this form should be 'post' and the action of this form should be showtable.php.

The showtable.php page should extract the table name from the _POST array and then construct an SQL query to retrieve all rows and all columns from this database table. Then, it will first display the name of this table in a heading, and then display the entire table in an HTML table. Note that the first row of this HTML table should contain the column names of the database table.

5.3 Change room price — pricing.php

This page allows logged-in administrators to change the nightly price of a room. This page should display a form with the following input devices:

- A dropdown list which includes each room ID from the **rooms** table. The room IDs in this list should be retrieved by an SQL query to the **rooms** table.
- A text box for entering a new price.
- A submit button and a reset button.

Of the above, the first and the second fields are mandatory, and the second field must contain a price within the range of $10.0 -- $999.99. All validations only need to be done in server side.

This page should submit to itself using POSTBACK. Upon successful validation, an SQL query should be composed to update the price of the room in the database. If the update succeeds, the administrator should be informed about this with a message underneath the above form, and the above form is to be displayed again to allow the administrator to select another room to update the price.

Checkpoint (4 marks)

At the checkpoint, you should upload your relevant work to the TWA server and show from there the following:

- The homepage has an appropriate layout with all required links and contents. (1 mark)
- Customer registration page contains correct client-side validations for all input fields. Note that server-side validations are not required in this checkpoint. (2 mark, marks will be awarded based on the percentage of the validations you have implemented correctly)
- Show all the tables and their records in the database by the **listdata.php** provided to you. You need to change the username, password, etc. in it to yours. (1 mark)

Submission Details

1. Prepare a **Readme.txt** file that lists some customers' usernames and passwords for markers to test your website. If you have some special notes for the markers, you can also include them in this file.
2. Upload all your files to the **assignment1** folder in your TWA web site on the TL28serv server.
3. Run the submission script located at http://tl28serv.westernsydney.edu.au/twainfo/submit.asp. As part of the submission you will be prompted for your TWA website username and password. You will then be asked to read the UWS policy on plagiarism and certify that work submitted by you is your own work. This action will be logged in a database for future reference and is deemed to be evidence that you claim that your work is original. Next, you will need to select from a drop down list the work you are submitting, eg. Assignment 1, and click the "Submit Assessment" button. The web page will then display a listing of the files you have submitted along with a receipt number. You should print or save as PDF this page for proof of submission.

Notes:

- The submission script may be run more than once if needed. A record of all submissions will be automatically created by the submission script in a set of web log files and in a database for future verification purposes. This submission script copies the files from your website to a duplicate version on a marking web server. You will not be able to access the marking web server.
- For the detailed marking criteria for your final submission of the assignment, please see the rubric at My Grades in vUWS, which will be available within the next one or two weeks.