Lab No: Project Part 5

Lab Title: Program and deploy a Database-Driven Web Application

Objective: At the end of the lesson, students will be able to develop and deploy a database-driven web application to implement the functional requirements in accordance with the application requirements specification

Resources:

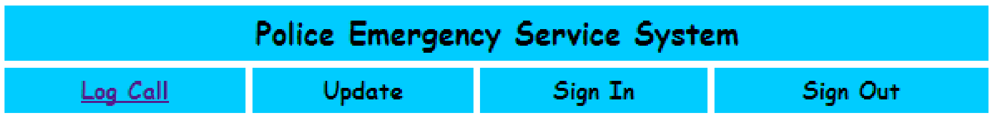
1. Development Computer.
2. Apache server, MySQL, and phpmyadmin
3. Code editor.

Procedures:

4. You will develop a database-driven web application to implement the Police Emergency Service System (PESS). As the idiom goes, "Rome was not built in a day." It will take XXX labs to complete this web application, provided you come and do your labs punctually and diligently.
5. You would have already created the database for the PESS in previous labs. You will now write the HTML code to create the web interfaces and PHP codes to implement the functional requirements for the PESS. You will have to refer to the function requirements and screen designs that you have completed in earlier labs regularly.
6. First, make sure the Apache server and MySQL are running. Next, launch eclipse, set the workspace as **C:\xampp\htdocs**. In eclipse, create a new project and named it as **pess_yourname** on the Apache server. How to do it? Re-visit the previous labs.

header.php

7. The first page should be the **header.php** page that will be displayed on the top of every other page. This header.php will include a title, logo, and list of menu link or any other stuffs that you want them to appear on every page. It should look like this:



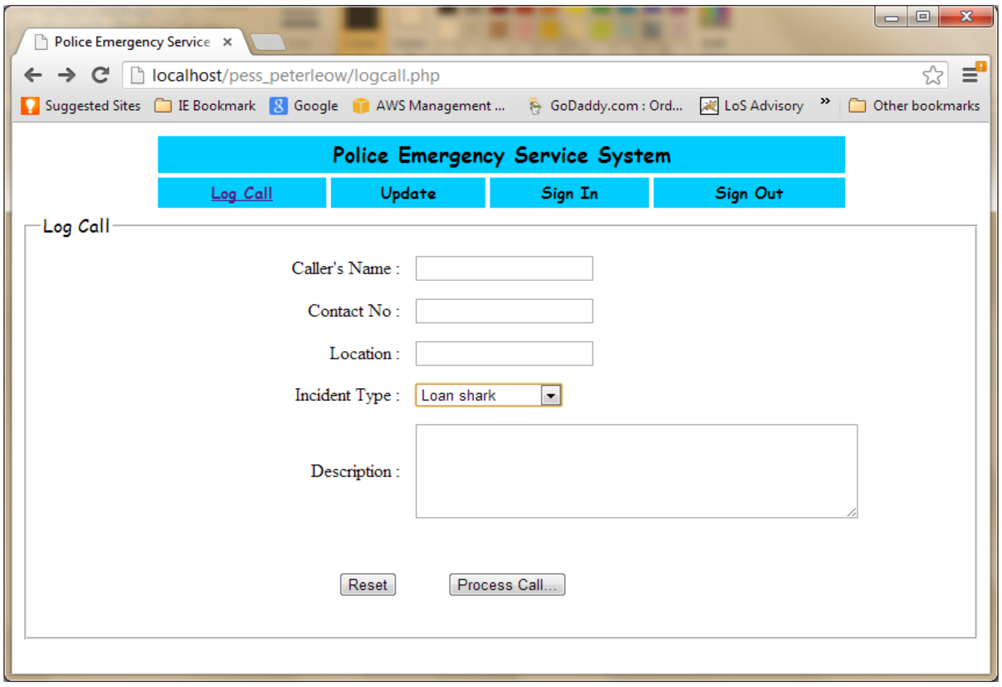
8. This will be created as an include file and its CSS should be inline type. Why? This page should not have the usual html, head and body tags. Why?

9. Next, you will create the web page to log calls received. We name the page as logcall.php. Re-look at the log call task, screen design and draw the activity diagram for this task in the following box. Activity diagram will help you understand the task by providing a visualization of its process flow, and **you shall do this at the start of every lab from now**.

10. You should use one external CSS to style all your web pages that are created from now on.

logcall.php

11. The logcall.php will allow the operator to enter details of each call received and save it to the database. It should look similar to this:



12. Upon receiving a call, the operator will enter the above details on the **Log Call** screen, click the **Process Call…** button to submit the form to **logcall.php** which will then insert the form data into the database.

13. Wait a minute, the Incident Type is empty. You have to populate this drop down box with the incident type data from the database. This has to done before this page is displayed. Now, let's do it.

Connect to MySQL Database

14. Before you can access data in a database, you must create a connection to the database. Re-visit lab 21, note down the database name, username, and password. Also note down the field names for incidenttype table in your database.

15. Enter the following code in the logcall.php. But, where to put it?

```
<?php
$con = mysql_connect("localhost","peterleow","password");
if (!$con)
  {
  die('Cannot connect to database : ' . mysql_error());
  }

mysql_select_db("pessdb_peterleow", $con);

$result = mysql_query("SELECT * FROM incidenttype");

$incidentType;

while($row = mysql_fetch_array($result))
{
// incidentTypeId, incidentTypeDesc

$incidentType[$row['incidentTypeId']] = $row['incidentTypeDesc'];

}

mysql_close($con);
?>
```

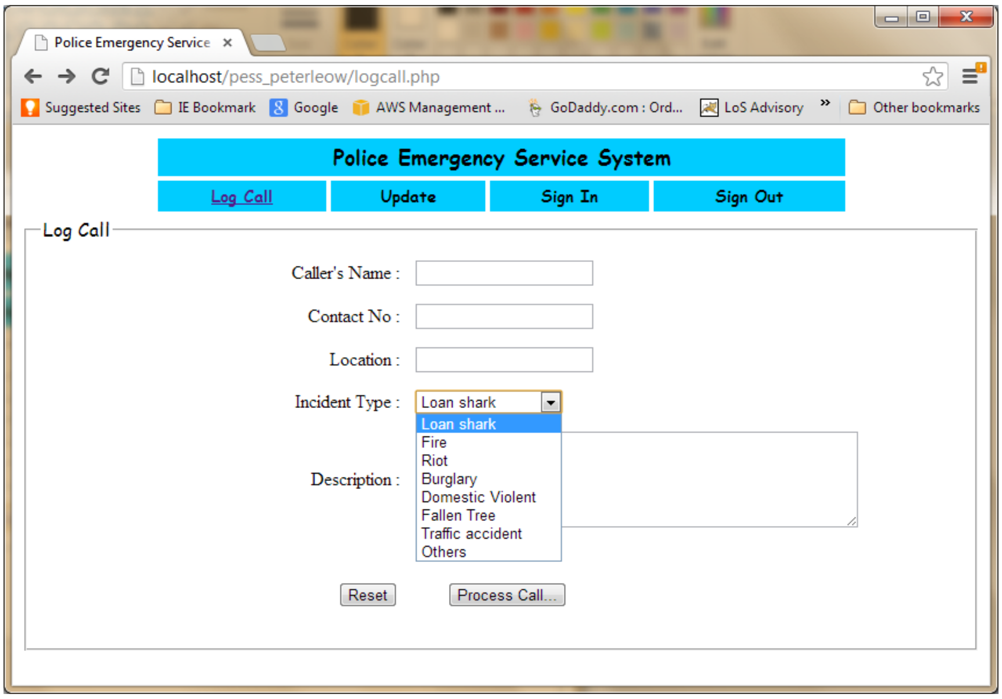
16. Explain the above code in the box below:

17. Next, we have to populate the Incident Type drop down box with the incident type data that we have retrieved from the incidenttype table. Enter the following code in the logcall.php. But, where to put it?

```
<tr>
  <td class="td_label">Incident Type : </td>
  <td class="td_Date">

  <select name="incidentType" id="incidentType">

  <?php foreach( $incidentType as $key => $value){ ?>

  <option value="<?php echo $key ?>"><?php echo $value ?></option>

   <?php } ?>

  </select>

  </td>
</tr>
```

18. Here, you have to weave in and out of PHP code and HTML code. Contrast this with the plain <select> syntax in HTML will help you understand. Explain the above code in the box below:

19. Test run your work in a browser, you should see the following screen:



20. Now, you will have to save the call log to the incident table in the database when the operator click the Process Call… button.

21. When the operator click the Process Call… button, the form data will be submitted to a script named in the action attribute of the form. (Recall your HTML lessons! If you have been using Dreamweaver dreamily, then you will have to re-learn your lessons.). Here, we will make it submit to itself, that is the **logcall.php**.

22. Rewrite the form element as follows:

```
<form name="frmLogCall" method="post" action="<?php echo
htmlentities($_SERVER['PHP_SELF']); ?>">
```

23. Again, you see PHP code being embedded in HTML. Google the explanation for the following statement:

htmlentities($_SERVER['PHP_SELF'])

24. Next, we put in the following code for inserting the form into the incident table of the database. But, where to put it.

```
<?php

$con = mysql_connect("localhost","peterleow","password");
if (!$con)
{
    die('Cannot connect to database : ' . mysql_error());
}

mysql_select_db("pessdb_peterleow", $con);

$sql = "INSERT INTO incident (callerName, phoneNumber, incidentTypeId,
incidentLocation, incidentDesc, incidentStatusId)  VALUES ('$_POST[callerName]',
'$_POST[contactNo]', '$_POST[incidentType]','$_POST[location]',
'$_POST[incidentDesc]', '1')";

if (!mysql_query($sql,$con))
{
        die('Error: ' . mysql_error());
}

mysql_close($con);

?>
```

25. Explain the above code in the box below:

26. Notice that we do not have to insert data into incidentId and timeCalled fields? Why?

27. Launch the page in a browse. Then, check out the incident table in the database, you should notice some undesirable outcome. What is it?

28. The problem is whenever this page is loaded or refreshed, new null incident record is inserted into the incident table! Why and how to overcome the problem?

29. Correct the mistake.

30. Have we done? Not so fast. What if the operator forgets to enter some fields, do you still allow the form to be submitted. NO!. You have to put in codes to check the validity of the user inputs before allowing the form data to be submitted to the server. Remember javascript? Do it now!

31. Test the logcall.php page in a browser and make sure it works correctly!

32. Version control your project and the database file to an online version control system.