Exercise 08 – Creating a basic Log In application using sessions

Aim:

- Create an application that allows logged in users to access home pages that correspond to their user type.
- There are two types of user: admin and employee

Setup:

Download Practical08 from Moodle and extract the contents to a folder named P08 in htdocs.

Practical08.zip includes completed files for employee registration (index.php) and logout (logout.php). You should not modify these files.

You are tasked with modifying connection.php, the login page (login.php), admin home page (adminHome.php) and the employee home page (employeeHome.php).

Task 1 : Set up the database and connection

- Using the phpMyAdmin database administration tool, create a database named **k00999999_p08_company** (replacing k00999999 with your knumber)
- In connection.php, set $dbname to the name of your database.

Task 2 : Create the 'user' table and add an admin user

- This practical uses a **user** table to record user information
- Create the user table with the following three columns:
    - **username** : A unique identifier for a user of the application
    - **password** : The users password
    - **usertype** : The users type. This can be either 'employee' or 'admin'

- The **username** is the primary key and therefore unique for each user in the table.
- **username**, **password** and **usertype** are all strings and therefore have the VARCHAR datatype. You may assume that the maximum length of each of the string columns is 40.
- Once the table has been created, insert a record for an admin user through phpMyAdmin. You may use any strings for the username and password, but the usertype must be **admin** (all lowercase letters)
- Example admin record:
    - username: Jack123
    - password: cf8f0c0d32522bc3d2ebe59d1fa46611d3369c96 (Note: This is the string generated when "Password1" is hashed with the ripemd160 algorithm)
    - usertype: admin

Task 3 : Review the Registration page (index.php)

The provided index.php file has been completed and contains a form that allows an employee to register. It also contains a link to the login page (login.php), which must be modified later in the practical. This section outlines the logic of index.php.

Creating / Joining a Session

- The session_start() function creates a new session if one does not already exist.
- Otherwise, it resumes the session that was created in a previous request.
- Variables set in the $_SESSION global array are available to all scripts during the course of a session.

Redirecting logged in users

- If a logged in user attempts to visit the registration page (index.php), they are redirected to another page according to their user type.
- This is accomplished by first checking if the $_SESSION[‘user’] session variable is set.
- This variable is set once a user logs in (login.php).
- $_SESSION[‘user’] is an associative array, containing user information from the **user** table, where the keys are the column names of the user table.
- Example value of the $_SESSION[‘user’] variable, once a user logs in:
    - array(“username” => “Jack123”, “password” => “Password123”, “usertype” => “admin”)
- If $_SESSION[‘user’] is set, the “usertype” value is checked on the array.
- If the usertype is **admin**, the user is redirected to the adminHome.php page.
- If the usertype is **employee**, the user is redirected to the employeeHome.php page.
- header(‘location: url’) is used to redirect to another page, where url is a placeholder for the new page name url.

Registration Form

- The registration form has input fields for “User Name” (username), “Password” (password1) and “Confirm Password” (password2).
- There is also a submit button (registerBtn)
- All text inputs have the “required” attribute. This prevents the form from being submitted if any of these fields are empty.
- The form action attribute determines the destination that requests from the form are made to.
- For this form, the action is the value of the php variable $_SERVER[‘PHP_SELF’]. This variable refers to the page from which the request was initiated from (index.php).
- The form method is “post”.

Handling Registration Requests

- Once the form is submitted, a post request is made to index.php
- The $_POST array is populated with data from the form, as key/value pairs.
- The keys are taken from the “name” attributes on each form input.
- A check is made to see if a request was received from the form, by checking if $_POST[‘registerBtn’] isset. This variable will only be set if a user submitted the form by selecting the register submit button.
- If a request was received from the form, a check is made to see if the “Password” and “Confirm Password” values match. If not, the script is exited.
- If the passwords match, a query is made against the database to check if a user already exists with the username submitted in the form.
- If a user already exists with the username, registration fails and the script is exited. This is because the username is the primary key of the user table and must therefore be unique.
- If the username supplied in the form does not already exist in the user table, an insert query can be made. Before inserting the record, the password is hashed with ripemd160 for security. The user record is then inserted into the user table.
- Note in the insert query that the usertype is hard coded as “employee”. Therefore, any users that are registered using this form are registered as employees.

Task 4 : Modify the Login Page (login.php)

- You have been provided with a partially completed login page (login.php).
- At the top of the file, session_start() resumes the session if one already exists. Otherwise a new session is created.
- If a user who is already logged in tries to access this page, they are redirected (admin users redirected to adminHome.php, employee users to employeeHome.php).
- You must add the code that handles log in requests.
- Use the username and password submitted from the form to retrieve the matching user row from the user table. Before making the query, you will need to hash the password with the ripemd160 algorithm, because the password in the user table is hashed.
- If a user is found, set the $_SESSION[‘user’] variable to the row that is returned:
    - Hint: mysqli_fetch_assoc() accepts a result set as an argument and returns the next row from the result set as an associative array, where the keys are the column names.
    - For example: array(“username” => “Jack123”, “password” => “Password123”, “usertype” => “admin”)

- After a user has been logged in, they should be redirected to an appropriate page according to their user type (either adminHome.php or employeeHome.php)
- If a user was not found, output an appropriate message.

Task 5 : Modify the Admin Home Page (adminHome.php)

- You have been provided with a partially completed admin home page (adminHome.php).
- This page should only be accessible to admin users.

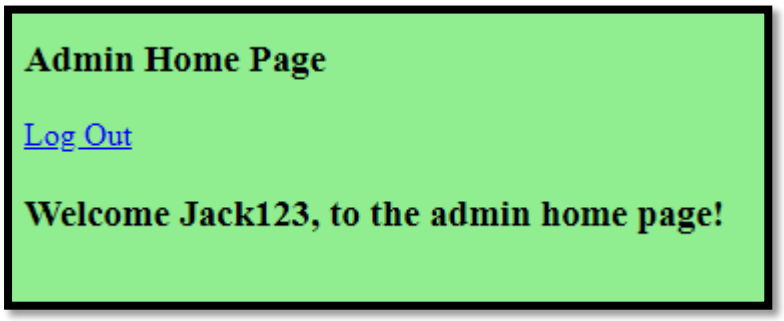
Task 5.1

- Add code that redirects a user to the registration page (index.php) if the user is not logged in as an admin

Task 5.2

- Add code that outputs a personalised greeting if the logged in user is an admin. The greeting should use the admins username.

Example Admin Home Page with greeting message



Task 6 : Modify the Employee Home Page (employeeHome.php)

- You have been provided with a partially completed admin home page (employeeHome.php).
- This page should only be accessible to employee users.

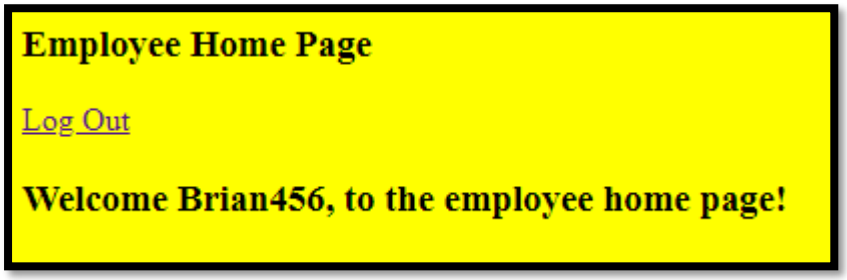
Task 6.1

- Add code that redirects a user to the registration page (index.php) if the user is not logged in as an employee

Task 6.2

- Add code that outputs a personalised greeting if the logged in user is an employee. The greeting should use the employees username.

Example Employee Home Page with greeting message



Task 7 : Review the logout page (logout.php)

The provided logout.php file has been completed and contains the code required to destroy the current session. This has the effect of logging the user out. The logout page is accessed as a link from the employee and admin home pages.

- session_start() resumes the current session
- $_SESSION = array() unsets all session variables
- The following code destroys the session cookie, by setting the expiry to a time in the past:

```
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}
```

- After destroying the session, a redirect is made back to the registration page (index.php):
    - header('location: index.php');

Task 8 Deliverables to moodle:

Upload the following to Moodle:

- The contents of your P08 folder as a ZIP archive.