**UNIVERSITY OF HUDDERSFIELD**

SCHOOL OF COMPUTING AND ENGINEERING

CIS2360 Relational Databases and Web Integration.

<https://brightspace.hud.ac.uk/d2l/home/226036>

***U2180702***

***MUHAMMAD ZIRKIYA***

Contents

[1.Project introduction 1](#_Toc124783835)

[1.1 introduction 1](#_Toc124783836)

[2.Project Description 1](#_Toc124783837)

[3.Project implementation 2](#_Toc124783838)

[5.References 12](#_Toc124783839)

# 1.Project introduction

# 1.1 introduction

Data-driven web development using PHP is a modern approach to creating dynamic and interactive websites. By utilizing various data sources, such as databases and APIs, PHP allows for efficient manipulation and display of information on the web. This method of web development is becoming increasingly popular as it enables a more personalized and engaging user experience. Additionally, it allows for easy maintenance and scalability of a website, by separating the presentation and data layers. With the rise of big data and internet of things, data-driven web development using PHP is an essential skill for any web developer1.2 project aims and objectives

# 2.Project Description

You will find on Brightspace five SQL scripts that create a database for the famous

Poppleton Dog Show. You are familiar with this application from the tutorial sessions, and

from the SQL Quiz.

Select the file that corresponds to your surname, and take a copy. (The databases generated

are identical in structure but load different data. The data is randomly generated.)

Load the data into the installation of MySQL that is most convenient to you. You may use

Selene, or you could use your own installation, via XAMPP (recommended). Examine the

database if you need to refresh your memory.

The requirements for the website are as follows. Note that there is no need to pay attention.

to styling (although you are welcome to do so if you want) – basic HTML formatting is fine.

You are expected to use PHP and MySQL to achieve the following. Do not use any.

framework, even if it would help. If you do not understand this sentence, ignore it!

The page should display a header (H1) and below it the message:

**“Welcome to Poppleton Dog Show! This year xx owners entered yy dogs in zz**

**events!”.**

**You should replace xx, yy, zz with the correct values found by running SQL queries**

**against the database**.

Add below a list that displays the top ten dogs in the show. This is defined as the ten.

dogs who have the highest average scores, provided they have entered more than

one event. Display just the dog’s name, breed, and average score.

Next to each dog, add the name of the owner and the owner’s email address. The

email address link should start up the default email application to send an email to

the owner. The owner’s name should link to a second page containing the owner’s

contact details.

# 3.Project implementation

RESULTS:

Printed number of owners and their dogs participated in events and value of events as well. Moreover owners name and their email and top 10 dogs with highest average score are also displayed via query.

Graphical user interface

Description automatically generated

**PRINTING\_Owners\_DOGS\_Events**

Graphical user interface

Description automatically generated

/ owner counting

$sql1 = "SELECT id FROM `owners`";

$run1 = $conn->query($sql1);

$ownercount = mysqli\_num\_rows($run1);

// dog-counting

$sql2 = "SELECT id FROM `dogs`";

$run2 = $conn->query($sql2);

$dogscount = mysqli\_num\_rows($run2);

// events-counting

$sql3 = "SELECT \* FROM `events`";

$run3 = $conn->query($sql3);

$eventscount = mysqli\_num\_rows($run3);

?>

<h1>“Welcome to Poppleton Dog Show! This year <?php echo $ownercount;?> owners entered <?php echo $dogscount;?> dogs in <?php echo $eventscount;?>

 events!”.      </h1>

Explanation:

The first query**, $sql1 = "SELECT id FROM owners**";, is selecting all the rows from the "owners" table and storing the result in the variable **$run1**. The **function mysqli\_num\_rows($run1)** is then used to count the number of rows in the result set and stores the count in the variable **$ownercount.**

The second query, **$sql2 = "SELECT id FROM dogs**";, is selecting all the rows from the "dogs" table and storing the result in the variable **$run2**. The function **mysqli\_num\_rows($run2)** is then used to count the number of rows in the result set and stores the count in the variable **$dogscount**.

The third query**, $sql3 = "SELECT \* FROM events**";, is selecting all the rows from the "events" table and storing the result in the variable $run3. The function **mysqli\_num\_rows($run3**) is then used to count the number of rows in the result set and stores the count in the variable $**eventscount.**

Finally, the script is outputting a string that uses the count variables to display the number of owners, dogs and events entered in the dog show.

**PRINTING Top 10 DOGS**

Graphical user interface

Description automatically generated

Explanation:

 //query for top ten dogs

$query = $conn->query("SELECT dogs.name AS Dog\_Name,breeds.name As Breed\_Name,AVG(score) AS Average\_Score,owners.name AS owner\_name,owners.email as Owner\_email

 from dogs INNER JOIn breeds

 on dogs.breed\_id=breeds.id INNER JOIN entries

 on dogs.id=entries.dog\_id INNER JOIN owners on owners.id=dogs.owner\_id

    Group by dog\_id HAVING COUNT(competition\_id)>1   ORDER BY AVG(score) DESC LIMIT 10");

    //$result=mysqli\_query($conn,$query);

    //setting variable for number of rows

$number=1;

Query is used to find.

Dog\_Name: The name of the dog

Breed\_Name: The name of the breed of the dog

Average\_Score: The average score of the dog across all competitions

owner\_name: The name of the owner of the dog

Owner\_email: The email of the owner of the dog

The query is joining the following tables:

dogs: The table containing information about the dogs

breeds: The table containing information about the breeds of the dogs

entries: The table containing information about the entries of the dogs in competitions

owners: The table containing information about the owners of the dogs

The query is grouping the results by the **dog\_id,** which means that all rows with the same **dog\_id** will be grouped together and will be considered as one group.

The HAVING clause is used to filter the groups based on a specified condition. In this case, the query is only selecting groups that have a COUNT of **competition\_id** greater than 1. This means that the query is only selecting dogs that have competed in more than one competition.

Finally, the query is ordering the results by the **AVG(score)** in descending order and limiting the results to the top 10. This means that the query will return the top 10 dogs with the highest average score and will order them in descending order.

**PRINTING OWNER NAME AND EMAIL OF EACH DOG**

Graphical user interface

Description automatically generated

while($row = mysqli\_fetch\_assoc($query))

{

  ?>

   <tr>

   <td><?php echo $number++?></td>

    <td><?php echo $row["Dog\_Name"]?></td>

    <td><?php echo $row ['Breed\_Name']?></td>

    <td><?php echo $row ['Average\_Score']?></td>

    <td><a href= "owner.php?owner\_name=<?php echo $row ['owner\_name']?>"><?php echo $row ['owner\_name']?></a></td>

    <td><a href="mailto: <?php echo $row ['Owner\_email']?>"><?php echo $row ['Owner\_email']?></a></td>

    <tr>

    <?php

}?>

<!-- Ending of Table -->

The while loop is using **the mysqli\_fetch\_assoc()** function to fetch each row of the query result as an associative array. The loop will continue to execute as long as there are more rows in the query result. Inside the loop, the code is using a combination of PHP and HTML to output the data from each row of the query result. The first column of the table is a counter, that is being incremented by 1 for each row, and is used to display the serial number of the top 10 dogs.one column is displaying the **Dog\_Name** from the query result, the third column is displaying the **Breed\_Name** and the fourth column is displaying the Average Score of the dog.one column contains the name of the owner of the dog, which is linked to a new page "**owner.php**" and passed the **owner\_name** as a GET parameter in the URL. The column contains the email of the owner, which is linked to a mailto link, so that the user can click on the email address and it will open their email client.

**OWNERs DETAILS (owner.php)**

**A picture containing diagram

Description automatically generated**

When user click on owner name it takes him to another page on which website display owner details in above provided figure .

Explanation:

//connection to database

include './connection.php';

//getting pressed owner name from welcome page

if (isset($\_REQUEST['owner\_name']) && $\_REQUEST['owner\_name']!='') {

  $name = $\_REQUEST['owner\_name'];

  //if owner is not find

}else{

  die('Owner not found');

}

//query for owner details

 //owner count========================

$query = "SELECT \* FROM `owners` WHERE name='$name'";

$con = $conn->query($query);

$table = mysqli\_fetch\_array($con);

//print\_r($data);

?>

<div class="hero-area">

      <!-- printing owner details in a lable -->

            <ul class="ownoder-details">

      <p class="img-wrap">

            <img width="300px" src="./images/user.png" alt="user">

          </p>

            <li>Name: <?php echo $table['name']?></li>

            <li>Email: <a href="mailto: <?php echo $table ['email']?>"><?php echo $table ['email']?></a></td></li>

            <li>Phone: <a href="Callto: <?php echo $table ['phone']?>"><?php echo $table ['phone']?></a></li>

            <li>Adress: <?php echo $table['address']?></li>

</ul>

</div>

The code first includes a connection file to connect to the database. Then it checks if the request contains **an owner\_name** parameter and if it is not empty, the code assigns this value to the $name variable. If the **owner\_name** parameter is not set or is empty, the script dies and displays "Owner not found".Next it creates a query that selects all columns from the owners table where the name is equal to the value of the $name variable. The query is executed using the **mysqli\_query()** function and the result is stored in the $con variable then uses the **mysqli\_fetch\_array()** function to fetch a single row of data from the query result and store it in **the $table** variable then uses a combination of PHP and HTML to display the owner's details in a list format. It also uses an image of a user and displays the owner's name, email, phone, and address. The email and phone numbers are also turned into links, so that users can click on them to send an email or make a call.

**Index.php**

<?php

session\_start();

// If the user is not logged in, then redirect to login page

if(empty($\_SESSION["username"])){

    echo $\_SESSION["username"];

    header("location: login.php");

    die();

}

// separate file for uniform header

include './header.php';

// separate file for uniform left column

include './leftcolumn.php';

// separate file for uniform right column

include './rightcolumn.php';

// separate file for uniform footer

include './footer.php';

?>

**Explanation:**

this code is checking if a user is logged in by checking if the "username" session variable is set or not.It starts by calling the **session**\_start() function to start a session.Then it checks if the "username" session variable is empty or not using the **empty()** function. If it is empty, it means that the user is not logged in and the code redirects the user to the login.php page using the **header()** function and then terminates the script using the **die()** function.If the user is logged in, the code continues to execute and includes 4 separate files:

**header.php**: This contains file code for the top section of the webpage header, navigation.

**leftcolumn.php**: This file contains the code for the left column of the webpage.

**rightcolumn.php**: This file contains the code for the right column of the webpage.

**footer.php:** This file contains the code for the bottom section of the webpage.

The purpose of including these files is to maintain a consistent layout across different pages of the website. This allows the developer to separate the code for the different sections of the webpage and make it easier to maintain and update.

**Login.php**

A picture containing background pattern

Description automatically generated

<?php

session\_set\_cookie\_params(1800); // 30 minutes \* 60 seconds

if (session\_status() != PHP\_SESSION\_ACTIVE) {

    session\_start();

}

// If the user is already logged in, then redirect to website

if (!empty($\_SESSION["username"])) {

    header("location: index.php");

    die();

}

// Start the session

session\_start();

// If the user is already logged in, then redirect to website

if (!empty($\_SESSION["username"])) {

    header("location: index.php");

    die();

}

// If parameters username and password not properly set, redirect to home page

if (!empty($\_POST['username']) && !empty($\_POST['password'])) {

    // A separate file to hide login details

    include './connection.php';

    // username and password sent from form

    $username = $conn->real\_escape\_string($\_POST['username']);

    $password = $conn->real\_escape\_string($\_POST['password']);

    $query = "SELECT username

            FROM ea\_Admin

            WHERE username = '$username' and password = '$password';";

    // Run Select SQL query

    $results = $conn->query($query);

    $count = $results->num\_rows;

    // Close connection after executing the query

    $conn->close();

    // If result matched given username and password, there must be 1 row

    if ($count == 1) {

        $\_SESSION["username"] = $username;

        header("location: index.php");

        die();

    } else {

        $error = "Invalid username or password";

    }

}

?>

<!DOCTYPE html>

<html>

<head>

    <title>Dream Home</title>

    <link rel="icon" type="image/png" href="./images/DreamHomeFavicon.png"/>

    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>

    <link rel="stylesheet" type="text/css" href="styles/styles.css"/>

</head>

<body class="login">

    <form class="login\_form" action="login.php" method="POST">

        <label for"username">Username:</label><br/>

        <input type="text" name="username" required/><br/><br/>

        <label for="password">Password:</label><br/>

        <input type="password" name="password" required/><br/><br/>

        <?php

        if (!empty($error)) {

            ?>

            <p id="login\_form\_error"><?php echo $error ?></p><br>

            <?php

        }

        ?>

        <input type="submit" value="Login"/>

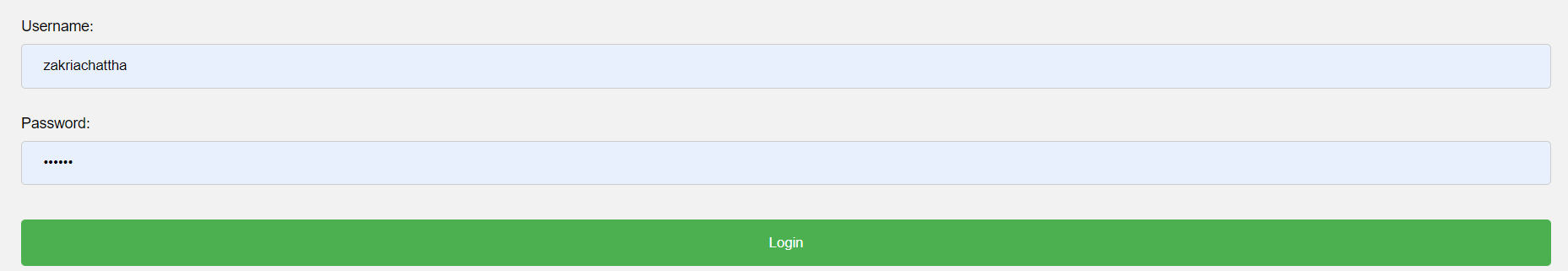
    </form>

</body>

</html>

It starts by setting the cookie parameters for the session to expire after 30 minutes using **session\_set\_cookie\_params(1800);** and then checks if a session is active or not. If it is not active, it starts a new session using **session\_start().**It then checks if the "username" session variable is set or not using the empty() function. If it is set, it means that the user is already logged in and redirects them to the index.php page using the **header()** function and then terminates the script using the die() function.Then it starts the session again and checks if the "username" session variable is set or not. If it is set, it means that the user is already logged in and redirects them to the index.php page using the header() function and then terminates the script using **the die()** function.Then, if the parameters "username" and "password" are properly set in the POST request, it includes the connection file that contains the database connection details and assigns the values of the "username" and "password" parameters to the **$username** and **$password** variables.It then creates a query that selects the username from the **ea\_Admin** table where the username and password match the values entered in the login form. The query is executed using the **$conn->query()** function and the number of rows returned by the query is stored in the **$count** variable.If the number of rows is equal to 1, it means that the entered username and password match a record in the database, so it sets the "username" session variable to the value of the **$username** variable and redirects the user to the index.php page. If the number of rows is not equal to 1, it means that the entered username and password do not match a record in the database, so it sets the $error variable to "Invalid username or password" and displays it on the page.The script then displays an HTML login form that contains fields for the username and password, and a submit button to submit the form. If the $error variable is set, it displays an error message. It is important to note that the script does not properly hash or encrypt the password before comparing it to the one stored in.

**Logout.php**

****

<?php

session\_set\_cookie\_params(1800); // 30 minutes \* 60 seconds

session\_start();

// unset the session variables

unset($\_SESSION["username"]);

//destroy the session

session\_destroy();

//redirect to login page

header("location: login.php");

die();

?>

It starts by setting the cookie parameters for the session to expire after 30 minutes using **session\_set\_cookie\_params(1800);** and then starts the session using **session\_start().**Then it unsets the "username" session variable using the unset() function. This deletes the session variable and the data stored in it.Then it destroys the current session using the **session\_destroy()** function. This deletes all session data and the session cookie.Finally, it redirects the user to the login.php page using the **header()** function, and terminates the script using the **die()** function.The script effectively logs out the user by destroying the session and redirecting to login page. This is useful for when a user wants to end their current session and log in as a different user or when a user wants to end their session when they are finished using the website.

**Connection.php(**for establishing connection with database)

<?php

// Login details (please adjust according to your details)

$servername = "localhost"; // "selene.hud.ac.uk";

$username = "u2180702"; // "U1234567"

$password = "MZ10nov01mz";  // By default, no password for XAMPP. Alternatively,

                 // password for Selene (as provided by IT)

$dbname = "u2180702"; // "U1234567"

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

    die("Connection failed: " . $conn->connect\_error);

}

?>

It starts by defining the login details for the database, including the server name, username, password, and database name. The server name is set to "localhost" which means that the database is running on the same server as the script. The username, password and dbname are specific to the user and the database. It then creates a new instance of the mysqli class, passing in the login details as arguments. This creates a new connection to the database using the mysqli extension. Next, it checks if the connection to the database is successful or not. If the connection is unsuccessful, it will display an error message with the reason for the connection failure. If the connection is successful, it continues to execute the rest of the script.this code is a basic way to connect to a MySQL database using the mysqli extension, it is necessary to include this script in every page that needs to access the database.

# 5.References

* Gilmore, W. J., & Bryla, B. (2007). *Beginning PHP and oracle: From novice to professional*. Apress. <https://doi.org/10.1007/978-1-4302-0367-4_1>
* Morrison, M., & Morrison, J. (2000). *Database-driven web sites*. Course Technology.
* Caya, A. (2018). *Mastering the faster web with PHP, MySQL and JavaScript: Develop state-of-the-art web applications using the latest web technologies* (1st ed.). Packt Publishing.
* Yank, K. (2012). *PHP & MySQL: Novice to ninja* (5th ed.). SitePoint.
* Welling, L., & Thomson, L. (2017;2016;). *PHP and MySQL web development* (Fifth ed.). Addison-Wesley.
* Nixon, R. (2012). *Learning PHP, MySQL, JavaScript, and CSS* (2nd ed.). O'Reilly.

****