Criterion C: Development

Table of Contents

Database

For my Digital Diary System, I will be making use of the MAMP phpmyadmin localhost Apache server, where I will be managing my database.

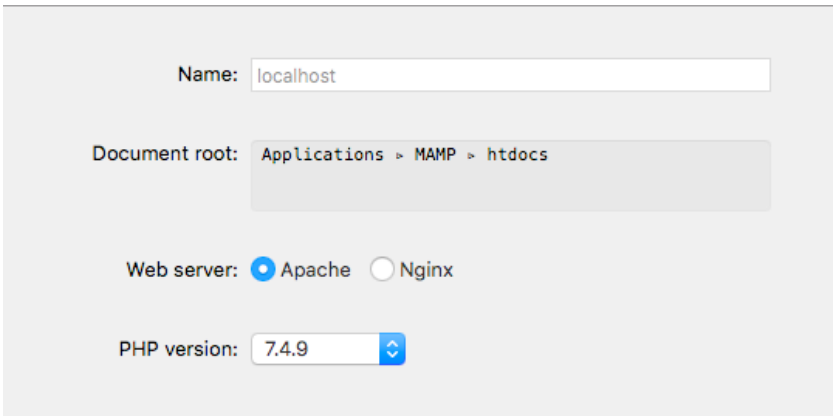

Image 1: server connection

I have created the following 6 tables:

1. User: This table stores the id, name, birthdate, username and password of the user while signing up. It is also used for logging the user in by validating the username and password.

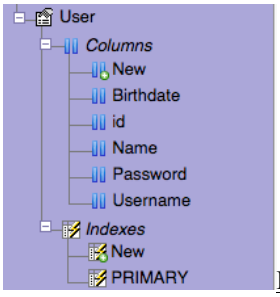

Image 1: Table User

2. Entries: This table stores the id, title, body, name of the image and the date and time the entry was created at by the user.

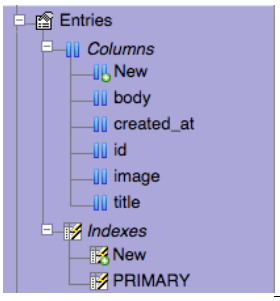

Image 2: Table Entries

3. events: This table stores the id, starting date and time and ending date and time of an event, as well as the title of the event.

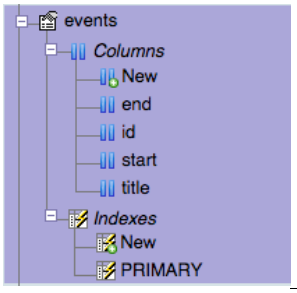

Image 3: Table events

4. Expenses: This table stores the id, month, year, savings, expenses, and final savings of the user.

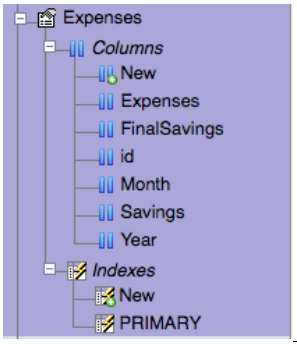

Image 4: Table Expenses

5. help: This table stores id, info and title, the manually inserted data that will be fetched and displayed in the help section of the Digital Diary System that my client asked for[1].

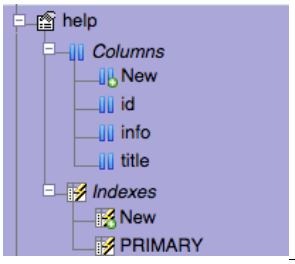

Image 5: Table help

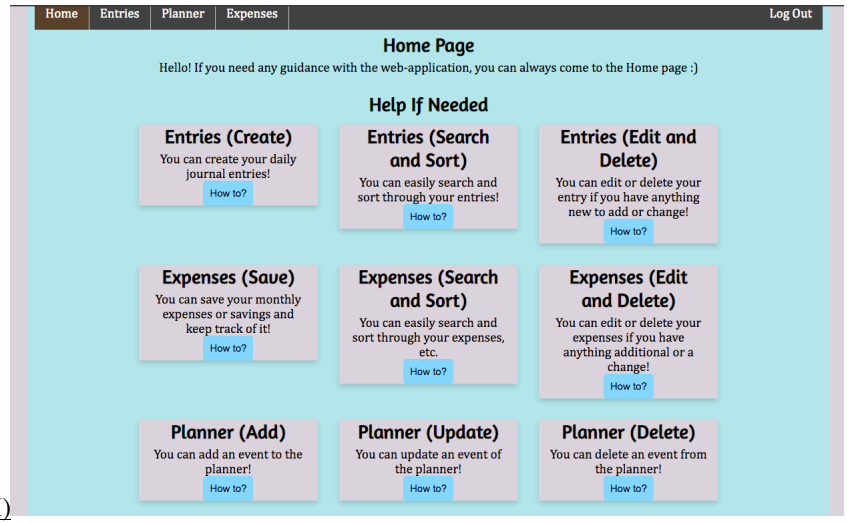

Image 6: Help Page (GUI)

[1] Refer to Appendix A.2 (Transcript of the interview)

SQL Queries

SQL commands such as **insert**, **update**, **delete** and **select** have been used for the functionality of my web-application. The SQL clauses such as **where** and **order by** have been where suitable.

By using MySQLi, the database has been connected to my PHP files, making the operation of my commands successful.

```php
<?php

$conn = mysqli_connect('localhost', 'S_12$M-83', 'J5oM2kLBrNL1wGwk');

//checkking connection
if(!$conn)
{
    echo 'Not connected to server';
}

//checking database connection
if(!mysqli_select_db($conn, 'Computer_IA'))
{
    echo 'Database not selected';
}

?>
```

Image 7: Connecting Database

INSERT

The insert query has been used to insert the user's name, birthdate, username and password in User table in the below given image. However, it is also used to insert month, year, savings, expenses, and final savings in the Expenses table; the title, body and name of the image in the Entries table; and title, start and end (date and time) in the events table.

```php
$sql = "INSERT INTO User (Name, Birthdate, Username, Password) VALUES ('$Name', '$Birthdate', '$Username', '$Password')";
mysqli_query($conn,$sql);
```

Image 8: Inserting into User table

UPDATE

The update query has been used to update the Expenses, events and Entries table.

```php
if(count($_POST)>0) {
    $sql = "UPDATE Expenses set id='" . $_POST['id'] . "', Month='" . $_POST['Month'] . "',
        Year='" . $_POST['Year'] . "', Savings='" . $_POST['Savings'] . "' ,Expenses='" . $_POST['Expenses'] . "',
        FinalSavings='" . $_POST['FinalSavings'] . "'  WHERE id='" . $_POST['id'] . "'";

    mysqli_query($conn, $sql);
    $message = "Successfully Updated!";
}
```

Image 9: Updating the Expenses table

DELETE

The delete query has been used to delete from the Expenses, events and Entries table.

```
$sql = "DELETE FROM Expenses WHERE id='" . $_GET['id'] . "'";
if (mysqli_query($conn, $sql)) {
```

Image 10: Deleting from Expenses table

SELECT

The select query has been used to select from the Entries, events and Expenses table in order to display the data inserted into the database. In addition, as shown below, it is also used to log in the user to her Digital Diary.

```
        $validation = "SELECT * FROM User WHERE Username = '$Username' AND Password = '$Password' ";

        $result = mysqli_query($conn, $validation);

        $num = mysqli_num_rows($result);


        if ($num == 1) {
            $_SESSION['user'] = $Username;
?>
```

Image 11: Selecting Username and Password from the User table

```
    $result = mysqli_query($conn,"SELECT * FROM Expenses WHERE id='" . $_GET['id'] . "'");
```

Image 12: Selecting the data in the Expense table

Preventing SQL Injection

"SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements."[2] In order to prevent this, mysqli_real_escape_string function has been used to escape special characters in a string that is used in an SQL query.

```
    //variables
    $Name = mysqli_real_escape_string($conn, $_POST['name']);
    $Birthdate = mysqli_real_escape_string($conn,$_POST['birthdate']);
    $Username = mysqli_real_escape_string($conn, $_POST['username']);
    $Password_1 = mysqli_real_escape_string($conn, $_POST['password1']);
    $Password_2 = mysqli_real_escape_string($conn,$_POST['password2']);
```

Image 13: Preventing SQLi

Use of in-built PHP Functionalities

The **isset () function** is used to determine that a variable is set or not[3]. Using this, the SQL queries can be carried out on the **$_POST** variable, the values collected from the HTML form using method post[4].

---

[2] (acunetix)

[3] (javatpoint)

Along with the $_POST variable, **$_FILES** and **$_GET variables** have also been used to collect the values from HTML form and carry out the SQL queries on the collected variables. **$_FILES** variable is an associative array containing items uploaded via the form[5], which is used to contain the images uploaded by the user within an entry.

```
if (isset($_POST['add'])) {

    $title = mysqli_real_escape_string($conn, $_POST['title']);
    $body = mysqli_real_escape_string($conn, $_POST['body']);
    $img =  mysqli_real_escape_string($conn, $_FILES['image']['name']); //name of the file
```

Image 14: isset (), $_POST and $_FILES

The **$_GET** variable has been mainly used to store the id, which would be a dynamic id, to delete, update and display (select) the data in a particular row of a given table.

```
<td><a href="Edit/index.php?id=<?php echo $row["id"]; ?>" type="button" class="edit">Edit</a></td>
```

```
id='" . $_GET['id'] . "'
```

Image 15 & 16: Use of $_GET ['id']

It has also been used in the successful functioning of the calculator.

```
<?php
if (isset($_GET['submit'])) {
    $result1 = $_GET['val1'];
    $result2 = $_GET['val2'];
    $operator = $_GET['operator'];
    switch ($operator) {
        case "None":
            echo "You need to select a method!";
            break;
        case "Add":
            echo $result1 + $result2;
            break;
        case "Subtract":
            echo $result1 - $result2;
            break;
        case "Multiply":
            echo $result1 * $result2;
            break;
        case "Divide":
            echo $result1 / $result2;
            break;
    }
}

?>
```

Image 17: Use of $_GET [''] in calculator

The **include** expression includes the specified file. In this case, the file, which contains the database connection, has been included in almost all the files. This helps in the reduction of redundancy of the code.

```
include "initialconn.php";
```

Image 18: Use of $_GET ['id']

---

[4] (w3schools.in)

[5] (tutorialspoint.com)

```php
<?php
session_start();
if (!isset($_SESSION['user'])) {
}

?>
```

PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers[6]. This function has been utilized to increase the usability.

Image 19 & 20: Use of PHP Session

```
<h1>Welcome <?php echo $_SESSION['user']; ?> to your very own Digital Diary</h1>
```

Linear Search

My client wanted a feature that would allow her to search for the month of expenses[7]. She also recommended that the search feature should be present in the Expenses section and not just the Entries section[8]. This required for the use of linear search through the list, for which JavaScript has been used.

```html
<tr>
    <th onclick="sortTable(0)">ID</th>
    <th onclick="sortTable(1)">Month</th>
    <th onclick="sortTable(2)">Year</th>
    <th onclick="sortTable(3)">Savings</th>
    <th onclick="sortTable(4)">Expenses</th>
    <th onclick="sortTable(5)">Final Savings</th>
    <th>Edit</th>
    <th>Delete</th>
</tr>
```

Image 21: List

```html
<script>
    function searchMonth() {
        var input, filter, table, tr, td, i, txtValue;
        input = document.getElementById("myMonth");
        filter = input.value.toUpperCase();
        table = document.getElementById("myTable");
        tr = table.getElementsByTagName("tr");
        for (i = 0; i < tr.length; i++) {
            td = tr[i].getElementsByTagName("td")[1];
            if (td) {
                txtValue = td.textContent || td.innerText;
                if (txtValue.toUpperCase().indexOf(filter) > -1) {
                    tr[i].style.display = "";
                } else {
                    tr[i].style.display = "none";
                }
            }
        }
    }
</script>
```

Image 22: Linear Search (Month)

[6] (javatpoint.com)

[7] Refer to Appendix A.2 (Transcript of the interview)

[8] Refer to Appendix A.6 (Client Feedback)

```
<script>
    function searchYear() {
        var input, filter, table, tr, td, i, txtValue;
        input = document.getElementById("myYear");
        filter = input.value.toUpperCase();
        table = document.getElementById("myTable");
        tr = table.getElementsByTagName("tr");
        for (i = 0; i < tr.length; i++) {
            td = tr[i].getElementsByTagName("td")[2];
            if (td) {
                txtValue = td.textContent || td.innerText;
                if (txtValue.toUpperCase().indexOf(filter) > -1) {
                    tr[i].style.display = "";
                } else {
                    tr[i].style.display = "none";
                }
            }
        }
    }
</script>
```

Image 23: Linear Search (Year)

Bubble Sort

Although my client did not explicitly mention the need for sort feature, being able to sort the table is a very useful and user-friendly trait. For this, I utilized the bubble sort algorithm in JavaScript.

```
<script>
    function sortTable(n) {
        var table, rows, switching, i, x, y, shouldSwitch, dir, switchcount = 0;
        table = document.getElementById("myTable");
        switching = true;
        //Set the sorting direction to ascending:
        dir = "asc";
        /*Make a loop that will continue until
        no switching has been done:*/
        while (switching) {
            //start by saying: no switching is done:
            switching = false;
            rows = table.rows;
            /*Loop through all table rows (except the
            first, which contains table headers):*/
            for (i = 1; i < (rows.length - 1); i++) {
                //start by saying there should be no switching:
                shouldSwitch = false;
                /*Get the two elements you want to compare,
                one from current row and one from the next:*/
                x = rows[i].getElementsByTagName("TD")[n];
                y = rows[i + 1].getElementsByTagName("TD")[n];
                /*check if the two rows should switch place,
                based on the direction, asc or desc:*/
                if (dir == "asc") {
                    if (x.innerHTML.toLowerCase() > y.innerHTML.toLowerCase()) {
                        //if so, mark as a switch and break the loop:
                        shouldSwitch = true;
                        break;
                    }
                } else if (dir == "desc") {
                    if (x.innerHTML.toLowerCase() < y.innerHTML.toLowerCase()) {
                        //if so, mark as a switch and break the loop:
                        shouldSwitch = true;
                        break;
                    }
                }
```

Image 24: Bubble Sort

OOP Concept Use

Inheritance

Using Object-Oriented Programming, the development of the application was easier as the use of **inheritance** made my coding reusable and less redundant.

Inheritance is when a class derives from another class[9]. In this case, the child classes inherit all the public properties and methods from the parent class.

```php
<?php

try {
    class Database
    {
        public $conn;
        public function __construct(){
            $this->conn = mysqli_connect('localhost', 'S_12$M-83', 'J5oM2kLBrNL1wGwk', 'Computer_IA');
        }


    }
} catch (Exception $e) {
    $err = $e->getMessage();
    echo $err;}
```

Image 25: Parent Class Database

```php
<?php

include "database.php";

class DataActions extends Database
{
    public function insert($table, $fields){
        //insert query
        $sql = "";
        $sql .= "INSERT INTO ".$table;
        $sql .= " (".implode(",", array_keys($fields)).") VALUES ";
        $sql .= "('".implode("','", array_values($fields))."')";
        $query = mysqli_query($this->conn,$sql);
        if($query){
            return true;
        }
    }

}

$obj = new DataActions;
if(isset($_POST['save'])){
    $myArray = array(
        "month" => $_POST["month"],
        "year" => $_POST["year"],
        "savings" => $_POST["savings"],
        "expenses" => $_POST["expenses"],
        "finalsavings" => $_POST["finalsavings"]
    );

    if($obj->insert("Expenses",$myArray)){
```

Image 26: Child Class DataActions

[9] (w3schools.com)

```php
    try {
        class OtherActions extends Database{
            public function view($table){
                $sql = "SELECT * FROM ".$table." ";
                $result = $this->conn->query($sql);
                $list = array();
                while ($data = $result->fetch_array()){
                    $list[] = $data;
                }
                return $list;
            }

            public function edit($table, $where){
                $condition = "";
                $list = array();
                foreach ($where as $key => $value) {
                    $condition .= $key. " = '" .$value. "' AND";
                }
                $condition = substr($condition, 0,-5);
                $sql = "SELECT * FROM ".$table." WHERE ".$condition." ";
                $result = $this->conn->query($sql);
                while ($row = $result->fetch_array()){
                    $list[] = $row;
                }
                return $list;

            }
        }
            } catch (Exception $e) {
        $err = $e->getMessage();
        echo $err;
    }
```

Image 27: Child Class OtherActions

Error handling

The Digital Diary System is also wrapped around the try catch block for easier error handling, hence easier coding.

```php
<?php
try {
    include_once '../database/initialconn.php';
    if(count($_POST)>0) {
        $sql = "UPDATE Expenses set id='" . $_POST['id'] . "', Month='" . $_POST['Month'] . "',
            Year='" . $_POST['Year'] . "', Savings='" . $_POST['Savings'] . "' ,Expenses='" . $_POST['Expenses'] . "',
            FinalSavings='" . $_POST['FinalSavings'] . "'  WHERE id='" . $_POST['id'] . "'";

        mysqli_query($conn, $sql);
        $message = "Successfully Updated!";
    }
    $result = mysqli_query($conn,"SELECT * FROM Expenses WHERE id='" . $_GET['id'] . "'");
    $row= mysqli_fetch_array($result);
} catch (mysqli_sql_exception $e) {
    $err = $e->getMessage();
    echo $err;
}
```

Image 28: Try-catch

Use of 2D Arrays

In order to store the images, 2 dimensional or multidimensional arrays have been used.

```php
if (!empty($_FILES['image']['name'])) {
    $fileext = explode('.', $img); //seperating filename from the extension
    $filecheck = strtolower(end($fileext));

    $fileext_store = array('png', 'jpg', 'jpeg');

    if (in_array($filecheck, $fileext_store)) {
        move_uploaded_file($_FILES['image']['tmp_name'], "images/$img");

        $body_enc = encryptthis($body, $key); //encrypt the password for security purpose

        $sql = "INSERT INTO Entries (title, body, image) VALUES ('$title', '$body_enc', '$img')";

        if (mysqli_query($GLOBALS['conn'], $sql)) {
```

Image 29: Use of 2D Array

FullCalendar JavaScript Library

Using AJAX

For this open source library, JQuery AJAX has been used to call PHP to handle the CRUD operations done for the Planner section.

```php
<script>
$(document).ready(function () {
    var calendar = $('#calendar').fullCalendar({
        editable: true,
        events: "crud/fetch.php",
        displayEventTime: false,
        eventRender: function (event, element, view) {
            if (event.allDay === 'true') {
                event.allDay = true;
            } else {
                event.allDay = false;
            }
        },
        selectable: true,
        selectHelper: true,
        select: function (start, end, allDay) {
            var title = prompt('Event Title:');

            if (title) {
                var start = $.fullCalendar.formatDate(start, "Y-MM-DD HH:mm:ss");
                var end = $.fullCalendar.formatDate(end, "Y-MM-DD HH:mm:ss");

                $.ajax({
                    url: 'crud/insert.php',
                    data: 'title=' + title + '&start=' + start + '&end=' + end,
                    type: "POST",
                    success: function (data) {
                        displayMessage("Added Successfully");
                    }
                });
                calendar.fullCalendar('renderEvent',
                        {
                            title: title,
                            start: start,
                            end: end,
                            allDay: allDay
                        },
                true
                        );
            }
            calendar.fullCalendar('unselect');
        },
```

Image 30 & 31: Use of JQuery AJAX to call PHP

```
editable: true,
eventDrop: function (event, delta) {
            var start = $.fullCalendar.formatDate(event.start, "Y-MM-DD HH:mm:ss");
            var end = $.fullCalendar.formatDate(event.end, "Y-MM-DD HH:mm:ss");
            $.ajax({
                url: 'crud/edit.php',
                data: 'title=' + event.title + '&start=' + start + '&end=' + end + '&id=' + event.id,
                type: "POST",
                success: function (response) {
                    displayMessage("Updated Successfully");
                }
            });
        },
eventClick: function (event) {
    var deleteMsg = confirm("Do you really want to delete?");
    if (deleteMsg) {
        $.ajax({
            type: "POST",
            url: "crud/delete.php",
            data: "&id=" + event.id,
            success: function (response) {
                if(parseInt(response) > 0) {
                    $('#calendar').fullCalendar('removeEvents', event.id);
                    displayMessage("Deleted Successfully");
                }
            }
        });
    }
}
```

Encryption and Hash

SHA256

To secure the password from any third parties, SHA256 has been used to hash the password before storing it in a database.

```
$Password = hash('sha256', $Password_1);
```

Image 32: Hashing the password

				id	Name	Birthdate	Username	Password
	Edit	Copy	Delete	1	Test	2021-04-01	Test	9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd1...
	Edit	Copy	Delete	6	Username	2021-04-05	Username	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a...
	Edit	Copy	Delete	10	Test123	2021-04-06	Test123	9b8769a4a742959a2d0298c36fb70623f2dfacda8436237df0...

Image 33: Hashed password in Database

OpenSSL encryption

The below functions in the code is used to encrypt and decrypt the body of the diary entry as requested from my client[10].

[10] Refer to Appendix A.6 (Client Feedback)

```php
function encryptthis($data, $key)
{
    $encryption_key = base64_decode($key);
    $iv = openssl_random_pseudo_bytes(openssl_cipher_iv_length('aes-256-cbc'));
    $encrypted = openssl_encrypt($data, 'aes-256-cbc', $encryption_key, 0, $iv);
    return base64_encode($encrypted . '::' . $iv);
}

function decryptthis($data, $key)
{
    $encryption_key = base64_decode($key);
    list($encrypted_data, $iv) = array_pad(explode('::', base64_decode($data), 2), 2, null);
    return openssl_decrypt($encrypted_data, 'aes-256-cbc', $encryption_key, 0, $iv);
}
```

Image 34: Encrypt and Decrypt functions

```php
if (empty($_FILES['image']['name'])) {
    $body_enc = encryptthis($body, $key);
```

Image 35: Encrypting the body

+ Options

				id	title	image	body	created_at
Edit	Copy	Delete		1	Test 1	photo_test_1.jpeg	N3FlWnA1eS9PN0FMSlhYQklxOHVFdlBaa1N5TWVtejIwR0pjSj...	2021-04-05 05:49:13
Edit	Copy	Delete		11	Test_2	photo_test_2.jpeg	aFdPaVFCMzNxdlRobHJPTHpBUXQ3ZG5ZS3NYMGliRUVkYzNXRV...	2021-04-06 13:07:15

Check all *With selected:* Edit Copy Delete Export

Image 36: Encrypted body in Database

```php
while ($row = $result->fetch_assoc()) {
    $body = decryptthis($row['body'], $key);
    $title = mysqli_real_escape_string($conn, $row['title']);

    $imageURL = '../images/' . $row["image"];
}
```

Image 37: Decrypting the body

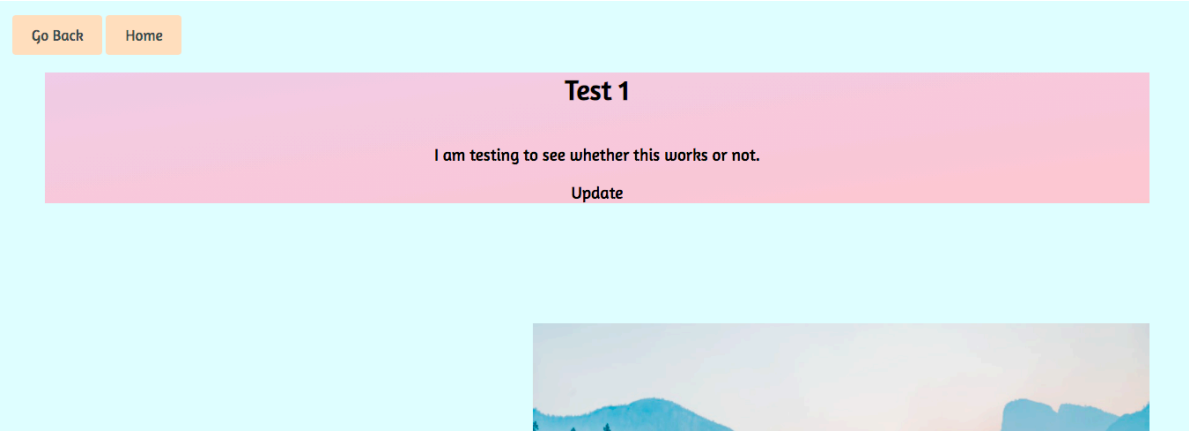

Image 38: Decrypted body while displaying

Error/Alert Notification System

In order to alert the user of the success or failure of different actions, the use of JavaScript's **alert** feature has been made.

```
30                  if (mysqli_query($GLOBALS['conn'], $sql)) {
31
32  ?>
33                      <!DOCTYPE html>
34                      <html>
35
36                      <body>
37
38                          <script>
39                              alert("Created!");
40                          </script>
41
42                      </body>
43
44                      </html>
45
46                  <?php
47
48                      header("refresh:0; url= index.php");
49                  }
50              } else {
51                  ?>
52                  <!DOCTYPE html>
53                  <html>
54
55                  <body>
56
57                      <script>
58                          alert("Extension not available. You can only upload files with the extension jpg, png or jpeg.");
59                      </script>
60
61                  </body>
62
63                  </html>
64
65                  <?php
66
67                  header("refresh:0; url= add.php");
68              }
```

Image 39: JavaScript Alert

```
17          if (!empty($_FILES['image']['name'])) {
18              $fileext = explode('.', $img); //seperating filename from the extension
19              $filecheck = strtolower(end($fileext));
20
21              $fileext_store = array('png', 'jpg', 'jpeg');
22
23              if (in_array($filecheck, $fileext_store)) {
24                  move_uploaded_file($_FILES['image']['tmp_name'], "images/$img");
25
26                  $body_enc = encryptthis($body, $key); //encrypt the password for security purpose
27
28                  $sql = "INSERT INTO Entries (title, body, image) VALUES ('$title', '$body_enc', '$img')";
29
30                  if (mysqli_query($GLOBALS['conn'], $sql)) {
```

Image 40: Setting conditions for image extension

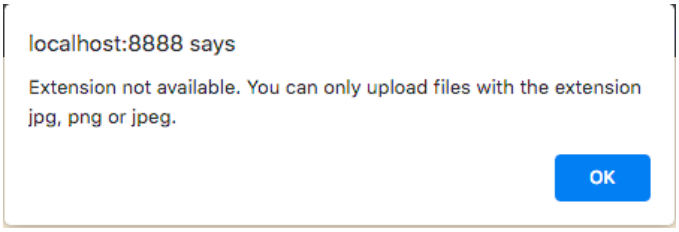

Image 41: Alert in GUI

Utilizing arrays, the customized error notification is pushed out when failed to meet certain conditions.

```
$errors = array();
```

```
    if(empty($Name)){
        array_push($errors, "Name is required");
    }
    if(empty($Birthdate)){
        array_push($errors, "Birthdate is required");
    }
    if(empty($Username)){
        array_push($errors, "Username is required");
    }
    if(empty($Password_1)){
        array_push($errors, "Password is required");
    }

    if($Password_1 != $Password_2){
        array_push($errors, "The two passwords do not match");
    }
```

Image 42, 43 & 44: Pushing error through array

```
<?php if (count($errors) > 0): ?>
    <div class="error">
        <?php foreach ($errors as $error): ?>
            <p> <?php echo $error; ?> </p>
        <?php endforeach ?>
    </div>
<?php endif ?>
```

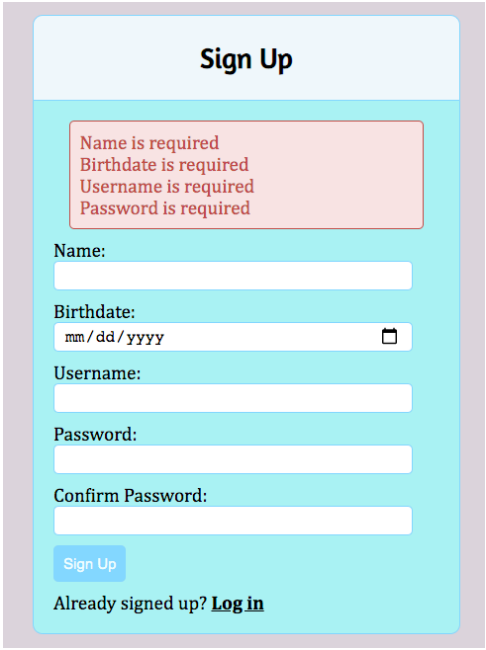

Image 45: Error notification in GUI

Integrating into GUI

Along with using CSS for the styling of the GUI, Font Awesome toolkit, Google Fonts library and Bootstrap have been integrated to make the front-end rich.

```
21    <!-- Font Awesome-->
22    <script src="https://kit.fontawesome.com/ad9fa19830.js" crossorigin="anonymous"></script>
23    <!-- x Font Awesome x -->
24
25    <!-- Google Fonts -->
26    <link href="https://fonts.googleapis.com/css2?family=Amaranth:ital@0;1&display=swap" rel="stylesheet">
27    <link href="https://fonts.googleapis.com/css2?family=Source+Serif+Pro&display=swap" rel="stylesheet">
28    <!-- x Google Fonts x -->
```

Image 46: Font Awesome and Google Fonts

```
22    <!-- CSS BOOTSTRAP -->
23    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet"
24    integrity="sha384-BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqIj61tLrc4wSX0szH/Ev+nYRRuWlolflfl" crossorigin="anonymous">
```

Image 47: Integrating Bootstrap

**Full codes can be found in Appendix B*

Word Count: 919 (**excluding** headings, screenshots/images and captions)

Bibliography

w3schools.com. (n.d.). *How TO - Filter/Search List* . Retrieved from w3schools.com: https://www.w3schools.com/howto/howto_js_filter_lists.asp

w3schools.com. (n.d.). *How TO - Sort a Table* . Retrieved from w3schools.com: https://www.w3schools.com/howto/howto_js_sort_table.asp

w3schools.com. (n.d.). *PHP OOP - Inheritance* . Retrieved from w3schools.com: https://www.w3schools.com/php/php_oop_inheritance.asp#:~:text=PHP%20%2D%20What%20is%20Inheritance%3F,methods%20from%20the%20parent%20class.&text=An%20inherited%20class%20is%20defined%20by%20using%20the%20extends%20keyword.

w3schools.in. (n.d.). *PHP GET and POST*. Retrieved from w3schools.in: https://www.w3schools.in/php/get-post/

acunetix. (n.d.). *What is SQL Injection (SQLi) and How to Prevent It* . Retrieved from acunetix: https://www.acunetix.com/websitesecurity/sql-injection/

javatpoint. (n.d.). *PHP isset() function* . Retrieved from javatpoint: https://www.javatpoint.com/php-isset-function

javatpoint.com. (n.d.). *PHP Session* . Retrieved from javatpoint.com: https://www.javatpoint.com/php-session

Mccullough, M. (2019, February 8). *PHP DATA ENCRYPTION INSERT RETRIEVE MYSQL DATABASE*. Retrieved from A1WEBSITEPRO.COM: https://a1websitepro.com/php-data-encryption-insert-retrieve-mysql-database/

tutorialspoint.com. (n.d.). *PHP $_FILES*. Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/php-files