



# MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

## VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

## MISSION

The Institute shall endeavour to incorporate the following basic missions in the teaching methodology:

### **Engineering Hardware – Software Symbiosis**

Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

### **Life – Long Learning**

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

### **Liberalization and Globalization**

The Institute endeavours to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

### **Diversification**

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

### **Digitization of Learning Processes**

The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.

### **Entrepreneurship**

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.



## **MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

### **COMPUTER SCIENCE & ENGINEERING DEPARTMENT**

#### **VISION**

“To be centre of excellence in education, research and technology transfer in the field of computer engineering and promote entrepreneurship and ethical values.”

#### **MISSION**

“To foster an open, multidisciplinary and highly collaborative research environment to produce world-class engineers capable of providing innovative solutions to real life problems and fulfil societal needs.”

# Department of Computer Science and Engineering

## Rubrics for Lab Assessment

Rubrics		0	1	2	3
		Missing	Inadequate	Needs Improvement	Adequate
R1	Is able to identify the problem to be solved and define the objectives of the experiment.	No mention is made of the problem to be solved.	An attempt is made to identify the problem to be solved but it is described in a confusing manner, objectives are not relevant, objectives contain technical/conceptual errors or objectives are not measurable.	The problem to be solved is described but there are minor omissions or vague details. Objectives are conceptually correct and measurable but may be incomplete in scope or have linguistic errors.	The problem to be solved is clearly stated. Objectives are complete, specific, concise, and measurable. They are written using correct technical terminology and are free from linguistic errors.
R2	Is able to design a reliable experiment that solves the problem.	The experiment does not solve the problem.	The experiment attempts to solve the problem but due to the nature of the design the data will not lead to a reliable solution.	The experiment attempts to solve the problem but due to the nature of the design there is a moderate chance the data will not lead to a reliable solution.	The experiment solves the problem and has a high likelihood of producing data that will lead to a reliable solution.
R3	Is able to communicate the details of an experimental procedure clearly and completely.	Diagrams are missing and/or experimental procedure is missing or extremely vague.	Diagrams are present but unclear and/or experimental procedure is present but important details are missing.	Diagrams and/or experimental procedure are present but with minor omissions or vague details.	Diagrams and/or experimental procedure are clear and complete.
R4	Is able to record and represent data in a meaningful way.	Data are either absent or incomprehensible.	Some important data are absent or incomprehensible.	All important data are present, but recorded in a way that requires some effort to comprehend.	All important data are present, organized and recorded clearly.
R5	Is able to make a judgment about the results of the experiment.	No discussion is presented about the results of the experiment.	A judgment is made about the results, but it is not reasonable or coherent.	An acceptable judgment is made about the result, but the reasoning is flawed or incomplete.	An acceptable judgment is made about the result, with clear reasoning. The effects of assumptions and experimental uncertainties are considered.

## LAB ASSESSMENT SHEET

[illegible]

6	Write a simple PHP program to demonstrate use of Simple function and Parametrized function.									
7	Develop web page with data validation.									
8	Write simple PHP program to- a. Set cookies and read it. b. Demonstrate session management									
9	Develop a simple application to- a. Enter data into database. b. Retrieve and present data from database.									
10	Develop a simple application to Update, Delete table data from database.									

## Experiment-1

- Aim:**
- Install and configure PHP, web server, MySQL
  - Write a program to print “Welcome to PHP”.
  - Write a simple PHP program using expressions and operators.

**Theory:** To set up a PHP environment, you need to install a web server (like Apache or Nginx), PHP, and MySQL. This stack, often referred to as LAMP (Linux, Apache, MySQL, PHP), enables dynamic web applications. After installation, configure the web server to process PHP files, typically found in the server's document root. A simple PHP program to display a message uses the echo statement, while expressions and operators can be demonstrated with arithmetic operations. PHP is a server-side scripting language designed for web development but can also be used for general-purpose programming.

### **a. Install and configure PHP, web server, MySQL:**

#### **Step 1: Update Package Index**

- Open the terminal.
- Update the system's package index and upgrade installed packages.

#### **Step 2: Install Apache Web Server**

- Install the Apache web server.
- Start the Apache service.
- Enable Apache to start automatically at boot.

#### **Step 3: Install MySQL**

- Install the MySQL server.
- Secure the MySQL installation by following the prompts to set a root password and other security options.

#### **Step 4: Install PHP**

- Install PHP along with the necessary extensions for Apache and MySQL.

#### **Step 5: Create a PHP Info File**

- Create a PHP file in the web server's root directory to test the PHP installation.

#### **Step 6: Restart Apache**

- Restart the Apache server to apply changes.

#### **Step 7: Test the Installation**

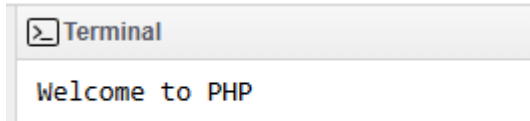
- Open a web browser and navigate to the PHP info file to check if PHP is working properly.
- Log in to the MySQL shell to confirm MySQL is functioning.

## B. Write a program to print “Welcome to PHP”

### Code:

```
<?php
echo "Welcome to PHP";
?>
```

### Output:

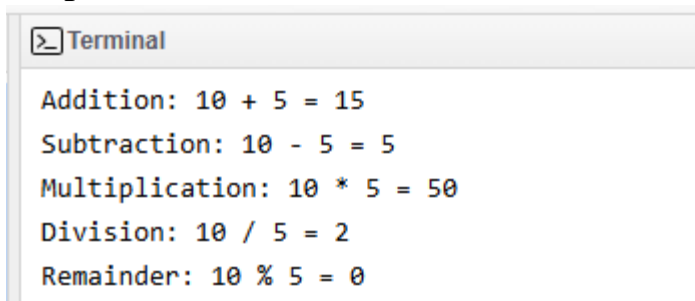


## C. Write a simple PHP program using expressions and operators.

### Code:

```
<?php
$a = 10;
$b = 5;
$sum = $a + $b;      // Addition
$difference = $a - $b; // Subtraction
$product = $a * $b;   // Multiplication
$quotient = $a / $b;  // Division
$remainder = $a % $b; // Modulus
echo "Addition: $a + $b = $sum\n";
echo "Subtraction: $a - $b = $difference\n";
echo "Multiplication: $a * $b = $product\n";
echo "Division: $a / $b = $quotient\n";
echo "Remainder: $a % $b = $remainder\n";
?>
```

### Output:



## Experiment-2

**Aim:** Write a PHP program to demonstrate the use of Decision making control structures using:

a. If statement b. If-else statement c. Switch statement

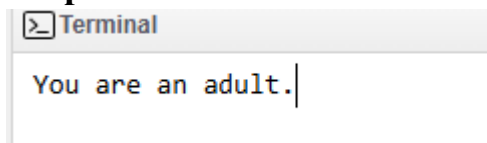
**Theory:** Decision-making control structures in PHP allow you to execute different code blocks based on certain conditions. The if statement evaluates a condition and executes a block of code if the condition is true. The if-else statement extends this by providing an alternative block of code to execute when the condition is false. The switch statement offers a more efficient way to compare a variable against multiple values, executing the corresponding block of code for the matching case. These control structures are essential for creating dynamic and responsive applications, enabling the program to make decisions based on user input or other criteria.

### A. If statement:

**Code:**

```
<?php
$age = 20;
if ($age >= 18) {
    echo "You are an adult.";
}
?>
```

**Output:**



### B. If-else Statement:

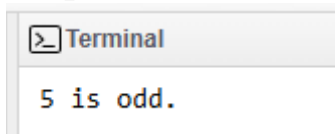
**Code:**

```
<?php
$number = 5;
if ($number % 2 == 0) {
    echo "$number is even.";
} else {
```



```
    echo "$number is odd.";
}
?>
```

### Output:

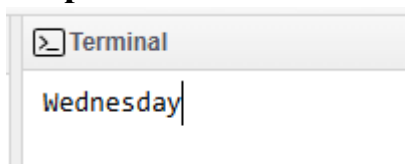


### C. Switch Statements:

#### Code:

```
<?php
$day = 3;
switch ($day) {
    case 1:
        echo "Monday";
        break;
    case 2:
        echo "Tuesday";
        break;
    case 3:
        echo "Wednesday";
        break;
    default:
        echo "Another day";
}
?>
```

### Output:



### Experiment-3

**Aim:** Write a PHP program to demonstrate the use of Looping structures using:  
a. while statement b. do-while statement c. for statement d. foreach statement

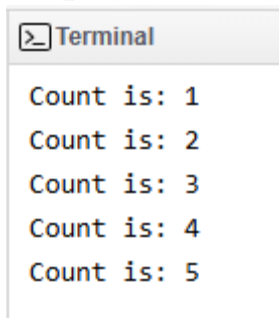
**Theory:** Looping structures in PHP allow you to execute a block of code repeatedly based on certain conditions. The while statement continues to execute as long as the specified condition is true. The do-while statement functions similarly but guarantees at least one execution of the loop body, as the condition is checked after the loop's body runs.

#### a. While statement:

**Code:**

```
<?php
$count = 1;
while ($count <= 5) {
    echo "Count is: $count\n";
    $count++;
}
?>
```

**Output:**



```
Terminal
Count is: 1
Count is: 2
Count is: 3
Count is: 4
Count is: 5
```

#### b. Do while Statement:

**Code:**

```
<?php
$count = 1;
do {
    echo "Count is: $count\n";
    $count++;
}
```

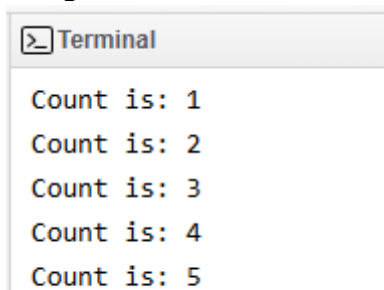
```
} while ($count <= 5);  
?>
```

### c. For Statements:

#### Code:

```
<?php  
for ($i = 1; $i <= 5; $i++) {  
    echo "Count is: $i\n";  
}  
?>
```

#### Output:



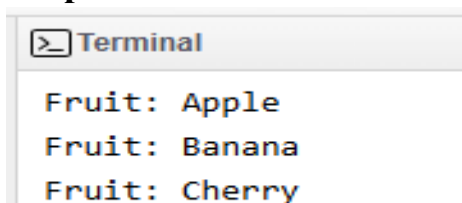
```
Terminal  
Count is: 1  
Count is: 2  
Count is: 3  
Count is: 4  
Count is: 5
```

### d. For each Statements:

#### Code:

```
<?php  
$fruits = array("Apple", "Banana", "Cherry");  
foreach ($fruits as $fruit) {  
    echo "Fruit: $fruit\n";  
}  
?>
```

#### Output:



```
Terminal  
Fruit: Apple  
Fruit: Banana  
Fruit: Cherry
```

## Experiment-4

**Aim:** Write a PHP program for creating and manipulating-

a. Indexed array b. Associative array c. Multidimensional array

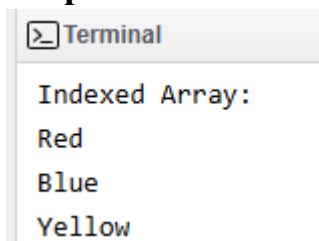
**Theory:** In PHP, arrays are versatile data structures that can hold multiple values. Indexed arrays store values with numeric indices, allowing for easy access and iteration. They are useful when the order of elements matters. Associative arrays utilize named keys instead of numeric indices, enabling more meaningful access to data by associating values with specific labels, making them ideal for key-value pairs. Multidimensional arrays are arrays that contain other arrays, facilitating the organization of complex data structures like matrices or tables.

### a. Indexed Array:

**Code:**

```
<?php
$colors = array("Red", "Green", "Blue");
$colors[] = "Yellow";
unset($colors[1]);
echo "Indexed Array:\n";
foreach ($colors as $color) {
    echo "$color\n";
}
?>
```

**Output:**



```
Terminal
Indexed Array:
Red
Blue
Yellow
```

### b. Associative array:

**Code:**

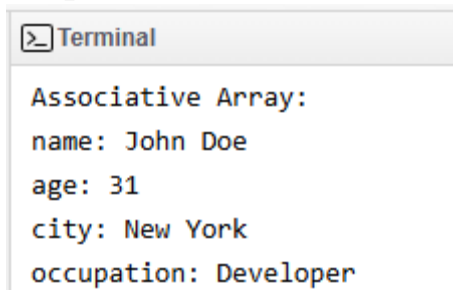
```
<?php
$person = array(
    "name" => "John Doe",
```

```

        "age" => 30,
        "city" => "New York"
    );
    $person["age"] = 31;
    $person["occupation"] = "Developer";
    echo "Associative Array:\n";
    foreach ($person as $key => $value) {
        echo "$key: $value\n";
    }
    ?>

```

### Output:



```

Terminal
Associative Array:
name: John Doe
age: 31
city: New York
occupation: Developer

```

### c. Multidimensional Array:


#### Code:

```

<?php
$students = array(
    array("name" => "Alice", "age" => 20),
    array("name" => "Bob", "age" => 22),
    array("name" => "Charlie", "age" => 23)
);
$students[1]["age"] = 23;
$students[] = array("name" => "David", "age" => 21);
echo "Multidimensional Array:\n";
foreach ($students as $student) {
    echo "Name: " . $student["name"] . ", Age: " . $student["age"] . "\n";
}
?>

```

## Output:

 Terminal

```
Multidimensional Array:  
Name: Alice, Age: 20  
Name: Bob, Age: 23  
Name: Charlie, Age: 23  
Name: David, Age: 21
```

## Experiment-5

**Aim:** a. Write a PHP program to -

i. Calculate length of string.

ii. Count the number of words in string without using string functions.

b. Write a simple PHP program to demonstrate use of various built-in string functions.

**Theory:** In PHP, strings are sequences of characters used to represent text. The length of a string can be determined by counting the number of characters it contains, which can be done through iteration rather than built-in functions. Counting words in a string involves identifying spaces between words and tallying the transitions from non-space to space characters. PHP also provides numerous built-in string functions that simplify string manipulation. Functions like `strlen()` determine string length, `strtoupper()` and `strtolower()` convert cases, `str_replace()` replaces substrings, and `substr()` extracts portions of strings.

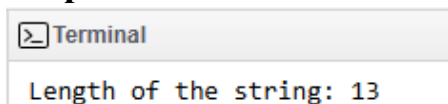
### a. Program for

I. Calculate length of string:

**Code:**

```
<?php
$string = "Hello, World!";
$length = 0;
for ($i = 0; isset($string[$i]); $i++) {
    $length++;
}
echo "Length of the string: $length\n";
?>
```

**Output:**

A screenshot of a terminal window with a title bar that says "Terminal". The terminal displays the output of the PHP program: "Length of the string: 13".

```
Terminal
Length of the string: 13
```

II. Count the number of words in string without using string functions.

**Code:**

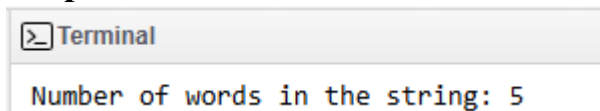
```
<?php
$string = "This is a sample string";
```

```

$wordCount = 0;
$inWord = false;
for ($i = 0; isset($string[$i]); $i++) {
    if ($string[$i] !== ' ') {
        if (!$inWord) {
            $inWord = true;
            $wordCount++;
        }
    } else {
        $inWord = false;
    }
}
echo "Number of words in the string: $wordCount\n";
?>

```

### Output:



A terminal window with a title bar that says "Terminal". Inside the terminal, the output of the PHP script is displayed: "Number of words in the string: 5".

### b. PHP Program Demonstrating Built-in String Functions:

#### Code:

```

<?php
// Demonstrating various built-in string functions
$string = "Hello, World!";

// 1. strlen() - Get the length of the string
$length = strlen($string);
echo "Length of the string: $length\n";

// 2. strtoupper() - Convert to uppercase
$upper = strtoupper($string);
echo "Uppercase: $upper\n";

// 3. strtolower() - Convert to lowercase
$lower = strtolower($string);
echo "Lowercase: $lower\n";

// 4. str_replace() - Replace a substring

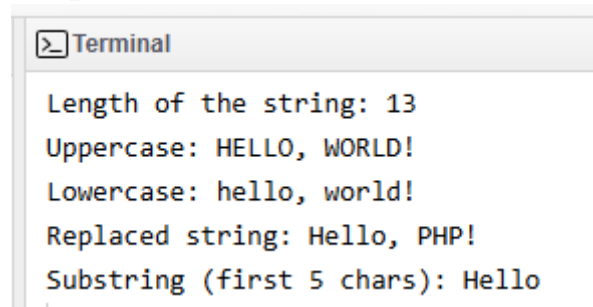
```



```
$replaced = str_replace("World", "PHP", $string);  
echo "Replaced string: $replaced\n";
```

```
// 5. substr() - Get a substring  
$substring = substr($string, 0, 5);  
echo "Substring (first 5 chars): $substring\n";  
?>
```

## Output:

A terminal window with a title bar that says "Terminal". The window contains the following output from a PHP script:

```
Length of the string: 13  
Uppercase: HELLO, WORLD!  
Lowercase: hello, world!  
Replaced string: Hello, PHP!  
Substring (first 5 chars): Hello
```

## Experiment-6

**Aim:** Write a simple PHP program to demonstrate use of Simple function and Parametrized function

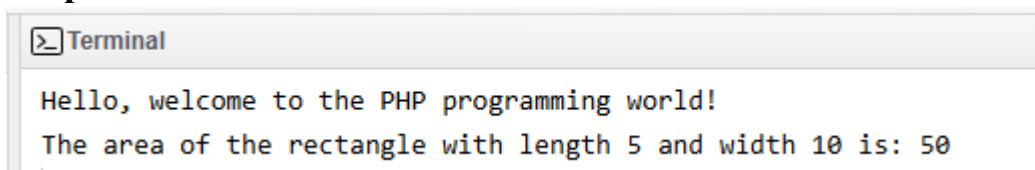
**Theory:** Functions in PHP are reusable blocks of code that perform specific tasks, helping to organize and simplify programming. A simple function does not require any parameters; it can be called without passing any values, allowing it to execute predefined actions, such as printing messages. In contrast, a parameterized function accepts inputs (parameters) when called, enabling it to perform operations based on the values provided. This flexibility allows for more dynamic and versatile code, as the same function can operate on different data. Functions enhance code readability and maintainability, making them a fundamental concept in PHP programming and software development.

### Code:

```
<?php
// Simple function: Prints a greeting message
function greet() {
    echo "Hello, welcome to the PHP programming world!\n";
}

// Parameterized function: Calculates the area of a rectangle
function calculateArea($length, $width) {
    $area = $length * $width;
    return $area;
}
greet();
$length = 5;
$width = 10;
$area = calculateArea($length, $width);
echo "The area of the rectangle with length $length and width $width is: $area\n";
?>
```

### Output:

A terminal window with a title bar that says "Terminal". The window contains two lines of text: "Hello, welcome to the PHP programming world!" followed by a new line, and "The area of the rectangle with length 5 and width 10 is: 50".

```
Terminal
Hello, welcome to the PHP programming world!
The area of the rectangle with length 5 and width 10 is: 50
```

## Experiment-7

**Aim:** Develop web page with data validation.

**Theory:** Data validation in PHP is crucial for ensuring that user inputs are accurate and secure before processing them. It involves checking the data provided by users against predefined rules to prevent errors and protect against malicious inputs. Common validation techniques include checking for empty fields, ensuring the correct format (e.g., email addresses), and sanitizing inputs to avoid security vulnerabilities like SQL injection. PHP offers built-in functions like `filter_var()` for validating and sanitizing data effectively. Implementing robust data validation enhances the reliability of web applications, improving user experience while safeguarding against invalid data submissions and potential threats.

**Code:**

```
<?php
$name = $email = "";
$nameErr = $emailErr = "";
$valid = true;
// Process form data when submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty(trim($_POST["name"]))) {
        $nameErr = "Name is required.";
        $valid = false;
    } else {
        $name = trim($_POST["name"]);
    }
    if (empty(trim($_POST["email"]))) {
        $emailErr = "Email is required.";
        $valid = false;
    } elseif (!filter_var(trim($_POST["email"]), FILTER_VALIDATE_EMAIL)) {
        $emailErr = "Invalid email format.";
        $valid = false;
    } else {
        $email = trim($_POST["email"]);
    }
}
?>
<!DOCTYPE html>
<html lang="en">
```

```

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>PHP Data Validation Example</title>
</head>
<body>
  <h2>Simple Form with Data Validation</h2>
  <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>"
method="post">
    <div>
      <label for="name">Name:</label>
      <input type="text" name="name" value="<?php echo
htmlspecialchars($name); ?>">
      <span style="color:red;"><?php echo $nameErr; ?></span>
    </div> <div>
      <label for="email">Email:</label>
      <input type="text" name="email" value="<?php echo
htmlspecialchars($email); ?>">
      <span style="color:red;"><?php echo $emailErr; ?></span>
    </div> <div>
      <input type="submit" value="Submit">
    </div> </form>
    <?php
    if ($valid && $_SERVER["REQUEST_METHOD"] == "POST") {
      echo "<h3>Submitted Data:</h3>";
      echo "Name: " . htmlspecialchars($name) . "<br>";
      echo "Email: " . htmlspecialchars($email) . "<br>";
    }
    ?>
  </body></html>

```

**Output:**

## Simple Form with Data Validation

Name:

Email:

## Experiment - 8

**Aim:** Write simple PHP program to -

a. Set cookies and read it. b. Demonstrate session management

**Theory:** Cookies and sessions are essential for maintaining state in PHP applications. Cookies are small text files stored on the client-side, used to remember user information across sessions, such as preferences or login details. They are set using `setcookie()` and can be accessed via the `$_COOKIE` superglobal. Sessions, on the other hand, store data on the server-side, enabling temporary data storage for individual users. Sessions are initiated with `session_start()` and data is accessed using the `$_SESSION` superglobal. They are beneficial for tracking user activities without exposing sensitive data to the client, enhancing security in web applications.

### a. Set cookies and read it

**Code:**

```
<?php
// Set a cookie
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1
day

// Read the cookie
if(isset($_COOKIE[$cookie_name])) {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value: " . $_COOKIE[$cookie_name];
} else {
    echo "Cookie '" . $cookie_name . "' is not set.";
}
?>
```

**Output:**

```
Cookie 'user' is not set.
```

## **b. Demonstrate session management:**

### **Code:**

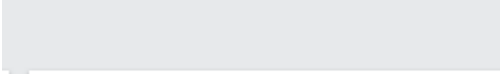
```
<?php
// Start the session
session_start();

// Set session variables
$_SESSION["username"] = "JohnDoe";
$_SESSION["email"] = "john@example.com";

// Access session variables
echo "<br>Session Variables:<br>";
echo "Username: " . $_SESSION["username"] . "<br>";
echo "Email: " . $_SESSION["email"] . "<br>";

// Destroy the session
session_destroy();
?>
```

### **Output:**



```
Session Variables:
Username: JohnDoe
Email: john@example.com
```

## Experiment-9

**Aim:** Develop a simple application to -

a. Enter data into database. b. Retrieve and present data from database.

**Theory:** This PHP application demonstrates basic database operations: data insertion and retrieval. The application consists of an HTML form for users to enter their name and email, which submits to the insert.php script. This script connects to a MySQL database, validates the input, and inserts it into a users table. Upon successful insertion, it provides a link to view the data. The retrieve.php script connects to the same database, retrieves all user records, and presents them in an HTML table. This example highlights how PHP interacts with MySQL for basic CRUD operations, essential for dynamic web applications.

### a. Enter data into a database:

**Code:**

```
<?php
$servername = "localhost"; // Change to your database host if necessary
$username = "your_database_username"; // Replace with your database username
$password = "your_database_password"; // Replace with your database password
$dbname = "your_database_name"; // Replace with your database name

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $sql = "CREATE TABLE IF NOT EXISTS users (
        id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(255) NOT NULL,
        email VARCHAR(255) NOT NULL
    )";
    $conn->exec($sql);
    $name = $email = "";
    $successMessage = $errorMessage = "";
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $name = trim($_POST["name"]);
        $email = trim($_POST["email"]);
```





```

    }
    if (!empty($errorMessage)) {
        echo "<p style='color:red;'>$errorMessage</p>";
    }
    ?>
</body>
</html>

```

### Output:



**Enter User Data**

Name:

Email:

### b. Retrieve and present data from database.

#### Code:

```

<?php
// Simulated user data (normally you would retrieve this from a database)
$users = [
    ['id' => 1, 'name' => 'John Doe', 'email' => 'john@example.com'],
    ['id' => 2, 'name' => 'Jane Smith', 'email' => 'jane@example.com'],
    ['id' => 3, 'name' => 'Alice Johnson', 'email' => 'alice@example.com']
];

// Function to display users
function displayUsers($users) {
    if (count($users) > 0) {
        echo "<table border='1'>";
        echo "<tr><th>ID</th><th>Name</th><th>Email</th></tr>";
        foreach ($users as $user) {
            echo "<tr>";
            echo "<td>" . htmlspecialchars($user['id']) . "</td>";

```

```

        echo "<td>" . htmlspecialchars($user['name']) . "</td>";
        echo "<td>" . htmlspecialchars($user['email']) . "</td>";
        echo "</tr>";
    }
    echo "</table>";
} else {
    echo "<p>No users found.</p>";
}
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simulated User Data</title>
</head>
<body>
    <h2>User Data</h2>
    <?php displayUsers($users); ?>
</body>
</html>

```

**Output:**

## User Data

ID	Name	Email
1	John Doe	john@example.com
2	Jane Smith	jane@example.com
3	Alice Johnson	alice@example.com

## Experiment - 10

**Aim:** Develop a simple application to Update, Delete table data from database.

**Theory:** A simple application to update and delete data from a database allows users to manage their data effectively. Updating involves changing existing records, while deleting removes records entirely. These operations are crucial for maintaining accurate and relevant information within the database. Using SQL commands, such as UPDATE and DELETE, these actions can be performed safely and efficiently. Proper validation and confirmation messages are essential to prevent accidental data loss. In PHP, these operations can be integrated with HTML forms to enable user interaction, allowing users to select which records to modify or remove easily.

### Code:

```
<?php
// Simulated database using an array
$users = [
    ["id" => 1, "name" => "John Doe", "email" => "john@example.com"],
    ["id" => 2, "name" => "Jane Smith", "email" => "jane@example.com"],
];

// Initialize message variables
$successMessage = "";
$errorMessage = "";

// Update operation (simulated)
if ($_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST['update'])) {
    $id = $_POST['id'];
    $name = $_POST['name'];
    $email = $_POST['email'];

    foreach ($users as &$user) {
        if ($user['id'] == $id) {
            $user['name'] = $name;
            $user['email'] = $email;
            $successMessage = "User updated successfully.";
            break;
        }
    }
}
```

```
    }  
  }  
}
```

```
// Delete operation (simulated)  
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['delete']))  
{  
    $id = $_POST['id'];  
  
    foreach ($users as $key => $user) {  
        if ($user['id'] == $id) {  
            unset($users[$key]);  
            $successMessage = "User deleted successfully.";  
            break;  
        }  
    }  
}  
}  
?>
```

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Update and Delete User Data</title>  
</head>  
<body>  
    <h2>User Data</h2>  
    <table border="1">  
        <tr>  
            <th>ID</th>  
            <th>Name</th>  
            <th>Email</th>  
            <th>Actions</th>  
        </tr>  
        <?php foreach ($users as $user): ?>  
            <tr>  
                <td><?php echo htmlspecialchars($user['id']); ?></td>  
                <td><?php echo htmlspecialchars($user['name']); ?></td>  
                <td><?php echo htmlspecialchars($user['email']); ?></td>
```

```

<td>
    <form action="" method="post">
        <input type="hidden" name="id" value="<?php echo $user['id']; ?>">
        <input type="text" name="name" value="<?php echo
htmlspecialchars($user['name']); ?>" required>
        <input type="email" name="email" value="<?php echo
htmlspecialchars($user['email']); ?>" required>
        <input type="submit" name="update" value="Update">
        <input type="submit" name="delete" value="Delete" onclick="return
confirm('Are you sure you want to delete this user?');">
    </form>
</td>
</tr>
<?php endforeach; ?>
</table>

<?php
// Display success message
if ($successMessage) {
    echo "<p style='color:green;'>$successMessage</p>";
}
?>
</body>
</html>

```

## Output:

### User Data

ID	Name	Email	Actions			
1	John Doe	john@example.com	John Doe	john@example.com	Update	Delete
2	Jane Smith	jane@example.com	Jane Smith	jane@example.com	Update	Delete