

Lab Experiment 1: Technologies Involved in AJAX

Objective:

To understand and explore the technologies involved in AJAX (Asynchronous JavaScript and XML) and how they work together to create dynamic web applications.

Software Used:

- Text Editor - Visual Studio Code
- Web Browser - Google Chrome
- Local Server - XAMPP

Methodology/Procedure:

1. Research and Identify Technologies:

- Investigate the core technologies used in AJAX, including HTML, CSS, JavaScript, XMLHttpRequest, JSON, and server-side languages.

2. Create Example Files:

- Create a JSON file (data.json) to simulate data that will be fetched.
- Create an HTML file (index.html) that includes AJAX functionality.

3. Set Up a Local Server:

- Use a local server to serve the HTML and JSON files, ensuring that AJAX requests can be made without cross-origin issues.

4. Implement AJAX Functionality:

- Write JavaScript code to make an AJAX request to the JSON file and display the fetched data.

5. Run the Application:

- Access the HTML file through the local server in a web browser and test the AJAX functionality.

Description:

This experiment illustrates how AJAX utilizes various technologies to facilitate asynchronous data exchange between a client and a server. It combines client-side languages like HTML, CSS, and JavaScript with server-side languages and data formats like JSON to enhance web applications' interactivity and responsiveness.

Technologies Involved:

- **HTML:** Provides the structure of the webpage.
- **CSS:** Styles the webpage to enhance user experience.
- **JavaScript:** Implements the logic to make AJAX calls.
- **XMLHttpRequest:** Enables asynchronous communication with the server.
- **JSON:** Acts as the data format for exchanging information.
- **Server-Side Technologies:** Processes requests and serves data.

Program:

1. JSON File (data.json):

```
{  
  "message": "Today's special: Grilled Salmon with Garlic Mashed Potatoes!"  
}
```

2. HTML File (index.html):

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>AJAX Example</title>  
  <style>  
    body {  
      font-family: Arial, sans-serif;  
      margin: 20px;  
    }  
    button {  
      padding: 10px 15px;  
      font-size: 16px;  
    }  
    #output {
```

```
        margin-top: 20px;
        font-weight: bold;
    }
</style>
<script>
    function fetchData() {
        var xhr = new XMLHttpRequest();
        xhr.open('GET', 'data.json', true);
        xhr.onload = function() {
            if (xhr.status >= 200 && xhr.status < 400) {
                var data = JSON.parse(xhr.responseText);
                document.getElementById('output').innerHTML = data.message;
            } else {
                console.error('Request failed with status:', xhr.status);
            }
        };
        xhr.onerror = function() {
            console.error('Request failed');
        };
        xhr.send();
    }
</script>
</head>
<body>
    <h1>Restaurant Specials</h1>
    <button onclick="fetchData()">Fetch Daily Special</button>
    <div id="output"></div>
</body>
</html>
```

Output:

Upon clicking the "Fetch Daily Special" button, the output displayed on the webpage will be:

- **Output in the browser:** "Today's special: Grilled Salmon with Garlic Mashed Potatoes!"



Result:

The experiment successfully demonstrates the integration of various technologies involved in AJAX. The HTML structure, styled with CSS, interacts with JavaScript to make asynchronous requests to a JSON file. This allows for a dynamic update of the content without a full page reload, showcasing the power and efficiency of AJAX in modern web development.

Lab Experiment 2: jQuery Events and Selectors

Objective:

To understand and explore jQuery events and selectors, and how they facilitate DOM manipulation and enhance user interaction in web applications.

Software Used:

- Text Editor - Visual Studio Code
- Web Browser - Google Chrome
- jQuery Library - linked via CDN

Methodology/Procedure:

1. **Research jQuery Basics:**
 - Study the jQuery library and its core functionalities, focusing on selectors and event handling.
2. **Set Up the Project:**
 - Create an HTML file (index.html) to demonstrate the use of jQuery selectors and events.
3. **Include jQuery Library:**
 - Include the jQuery library in the HTML file either by downloading it or linking it from a CDN.
4. **Implement jQuery Selectors:**
 - Write jQuery code to select various HTML elements using different types of selectors (ID, class, attribute).
5. **Implement jQuery Events:**
 - Write jQuery code to handle events such as click, mouseenter, and mouseleave on selected elements.
6. **Run the Application:**
 - Open the HTML file in a web browser and test the functionality of the jQuery selectors and events.

Description:

This experiment demonstrates how jQuery simplifies DOM manipulation through its selectors and event handling mechanisms. Selectors allow developers to target HTML elements easily, while event methods enable interactive features, improving user experience.

jQuery Events and Selectors Overview

- **Selectors:** jQuery provides a wide range of selectors, including ID selectors (#id), class selectors (.class), and attribute selectors ([attribute=value]). These selectors allow for easy targeting of HTML elements.
- **Events:** jQuery simplifies event handling with methods like .click(), .mouseenter(), and .mouseleave(), allowing developers to respond to user interactions seamlessly.

Program:

1. HTML File (index.html):

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>jQuery Events and Selectors</title>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<style>

    body {

        font-family: Arial, sans-serif;

        margin: 20px;

    }

    .box {

        width: 100px;

        height: 100px;

        background-color: lightblue;

        margin: 10px;

        display: inline-block;

        border: 1px solid #333;

    }

    .active {

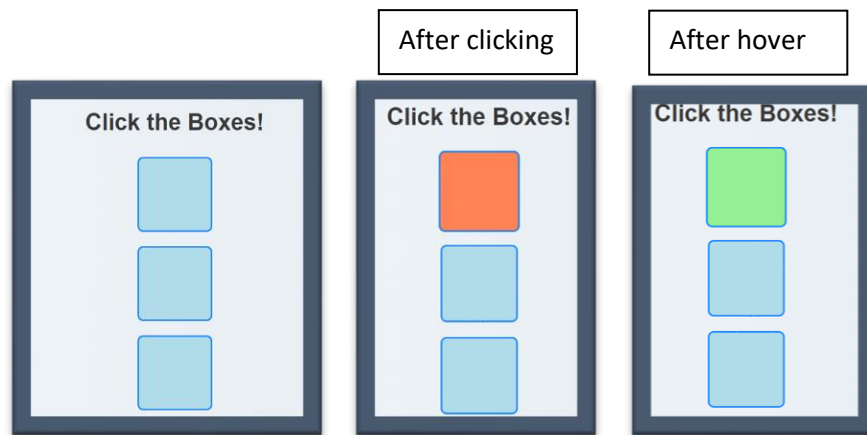
        background-color: coral;
```

```
    }  
</style>  
<script>  
    $(document).ready(function() {  
        // jQuery selector  
        $('.box').click(function() {  
            $(this).toggleClass('active');  
        });  
        $('.box').mouseenter(function() {  
            $(this).css('background-color', 'lightgreen');  
        }).mouseleave(function() {  
            $(this).css('background-color', 'lightblue');  
        });  
    });  
</script>  
</head>  
<body>  
    <h1>Click the Boxes!</h1>  
    <div class="box"></div>  
    <div class="box"></div>  
    <div class="box"></div>  
</body>  
</html>
```

Output

When the user interacts with the boxes:

- **Clicking a box** toggles the class active, changing its color to coral.
- **Hovering over a box** changes its background color to light green, and it reverts back to light blue when the mouse leaves.



Result

The experiment successfully demonstrates the use of jQuery selectors and events. The jQuery library simplifies the process of selecting HTML elements and handling user interactions. By utilizing jQuery, developers can create more dynamic and responsive web applications, enhancing the overall user experience.

Lab Experiment 3: PHP Program to Display a Digital Clock

Objective: To create a PHP program that displays the current time in a digital clock format, demonstrating the use of PHP for server-side scripting and real-time updates.

Software Used:

- XAMPP (for local server)
- PHP (version 7.0 or higher)
- Text Editor - Visual Studio Code

Methodology/Procedure:

1. Set Up Environment:

- Install XAMPP/WAMP/LAMP on your computer.
- Start the local server and ensure Apache and MySQL services are running.

2. Create PHP File:

- Open a text editor and create a new file named digital_clock.php.
- Save the file in the htdocs directory (for XAMPP) or the appropriate web directory for other servers.

3. Write PHP Code:

- Use the date() function to retrieve the current server time.
- Use HTML and CSS to format the clock display.
- Implement JavaScript (if necessary) to update the clock every second.

4. Run the Program:

- Open a web browser and navigate to http://localhost/digital_clock.php.
- Observe the digital clock displaying the current time.

Description: This program utilizes PHP to fetch the current server time and display it in a digital clock format, updating every second. It integrates PHP with HTML and JavaScript to create a dynamic user interface.

Program:

1. PHP File (digital_clock.php):

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Digital Clock</title>

    <style>

        body {

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

            background-color: #282c34;

            color: white;

            font-family: 'Arial', sans-serif;

            font-size: 48px;

        }

    </style>

    <script>

        function updateClock() {

            const now = new Date();

            const hours = String(now.getHours()).padStart(2, '0');

            const minutes = String(now.getMinutes()).padStart(2, '0');

            const seconds = String(now.getSeconds()).padStart(2, '0');

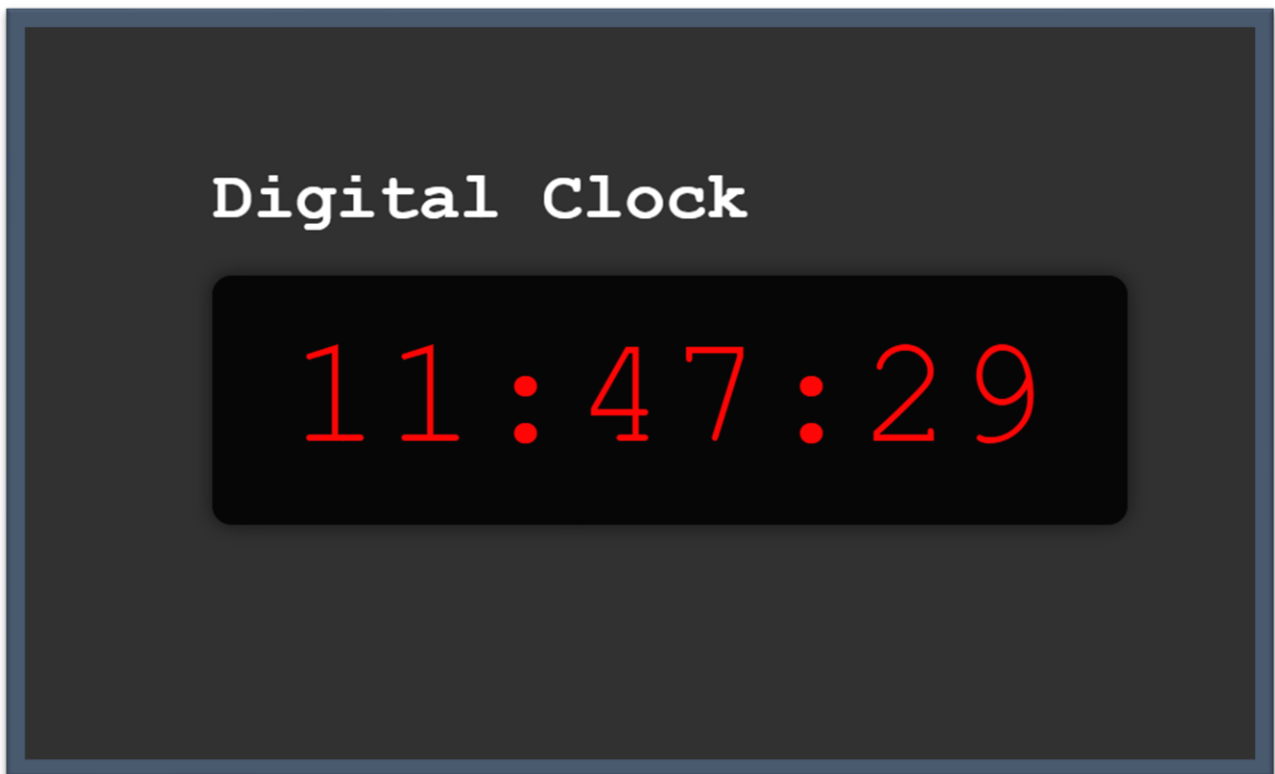
            document.getElementById('clock').textContent = `${hours}:${minutes}:${seconds}`;

        }

        setInterval(updateClock, 1000);
```

```
        window.onload = updateClock;
    </script>
</head>
<body>
    <div id="clock"></div>
</body>
</html>
```

Output: A web page displaying the current time in the format HH:MM:SS, which updates every second.



Result: The program successfully displays a digital clock that accurately shows the current time, updating in real-time. This experiment illustrates the effective use of PHP for server-side scripting combined with client-side JavaScript, enabling the creation of dynamic web applications.

Lab Experiment 4: PHP Program to Keep Track of Visitors

Objective: To create a PHP program that tracks the number of visitors to a webpage by storing the count in a session or a text file, demonstrating session management and file handling in PHP.

Software Used:

- XAMPP (for local server)
- PHP (version 7.0 or higher)
- Text Editor - Visual Studio Code

Methodology/Procedure:

1. Set Up Environment:

- Install XAMPP/WAMP/LAMP on your computer.
- Start the local server and ensure Apache and MySQL services are running.

2. Create PHP File:

- Open a text editor and create a new file named visitor_tracker.php.
- Save the file in the htdocs directory (for XAMPP) or the appropriate web directory for other servers.

3. Write PHP Code:

- Use PHP sessions or file handling to count the number of visits.
- Display the current visitor count on the webpage.

4. Run the Program:

- Open a web browser and navigate to http://localhost/visitor_tracker.php.
- Refresh the page to see the visitor count increment.

Description: This program uses PHP to track and display the number of visitors to a webpage. It demonstrates session management by using PHP sessions or file handling techniques to store and update the visitor count.

Program:

1. PHP File (visitor_tracker.php):

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

// Check if the visitor count is set in the session
if (!isset($_SESSION['visitor_count'])) {
```

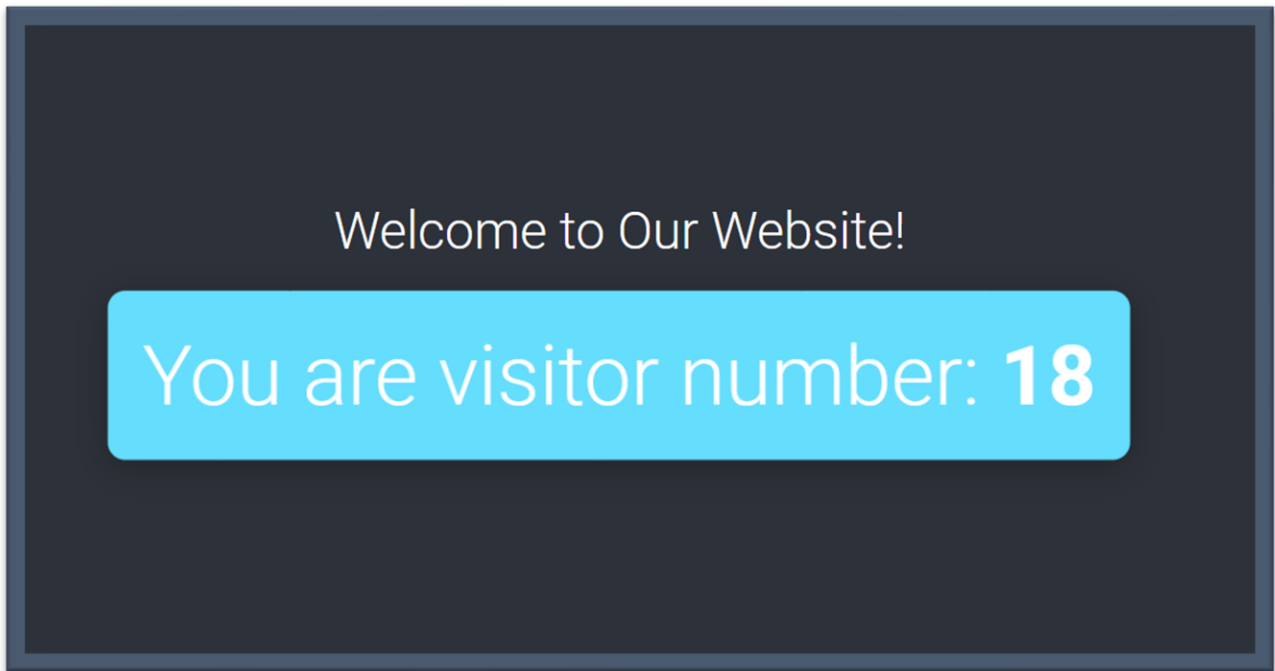
```

    $_SESSION['visitor_count'] = 0; // Initialize count if not set
}
// Increment the visitor count
$_SESSION['visitor_count']++;
// Store visitor count in a variable
$visitorCount = $_SESSION['visitor_count'];
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Visitor Tracker</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #f0f0f0;
            font-family: 'Arial', sans-serif;
            font-size: 24px;
        }
    </style>
</head>
<body>
    <div>
        <h1>Welcome to Our Website!</h1>
        <p>You are visitor number: <strong><?php echo $visitorCount; ?></strong></p>

```

```
</div>  
</body>  
</html>
```

Output: A web page displaying the message "You are visitor number: X", where X is the number of times the page has been visited in the current session.



Result: The program successfully tracks the number of visitors to a webpage by storing the count in a session or a text file, demonstrating session management and file handling in PHP.

Lab Experiment 5: PHP Programs for Calculator and Matrix Operations

Objective: To create PHP programs that implement simple calculator operations, find the transpose of a matrix, and perform multiplication of two matrices, demonstrating the use of PHP for mathematical computations.

Software Used:

- XAMPP (for local server)
- PHP (version 7.0 or higher)
- Text Editor - Visual Studio Code

Methodology/Procedure:

1. Set Up Environment:

- Install XAMPP/WAMP/LAMP on your computer.
- Start the local server and ensure Apache and MySQL services are running.

2. Create PHP Files:

- Create three separate files for each program:
 - calculator.php
 - transpose.php
 - multiply_matrices.php
- Save the files in the htdocs directory (for XAMPP) or the appropriate web directory for other servers.

3. Write PHP Code:

- Implement the calculator operations in calculator.php.
- Write the code for matrix transpose in transpose.php.
- Implement matrix multiplication in multiply_matrices.php.

4. Run the Programs:

- Open a web browser and navigate to each PHP file to test the functionality.

Description: This experiment demonstrates basic mathematical operations using PHP. It includes a simple calculator for basic arithmetic, a function to compute the transpose of a matrix, and a multiplication function for two matrices.

Program Details

1. Simple Calculator (calculator.php):

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Simple Calculator</title>

  <style>

    body { font-family: Arial, sans-serif; margin: 20px; }

    input[type="text"], input[type="number"] { margin: 5px; }

    button { margin-top: 10px; }

  </style>

</head>

<body>

  <h1>Simple Calculator</h1>

  <form method="post">

    <input type="number" name="num1" placeholder="Enter first number" required>

    <input type="number" name="num2" placeholder="Enter second number" required>

    <select name="operation" required>

      <option value="">Select Operation</option>

      <option value="add">Add</option>

      <option value="subtract">Subtract</option>

      <option value="multiply">Multiply</option>

      <option value="divide">Divide</option>

    </select>

    <button type="submit">Calculate</button>

  </form>
```



```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $num1 = $_POST['num1'];
    $num2 = $_POST['num2'];
    $operation = $_POST['operation'];
    switch ($operation) {
        case 'add':
            $result = $num1 + $num2;
            break;
        case 'subtract':
            $result = $num1 - $num2;
            break;
        case 'multiply':
            $result = $num1 * $num2;
            break;
        case 'divide':
            if ($num2 != 0) {
                $result = $num1 / $num2;
            } else {
                $result = "Error: Division by zero!";
            }
            break;
        default:
            $result = "Invalid operation!";
    }
    echo "<h2>Result: $result</h2>";
}
?>
</body>
```

```
</html>
```

2. Transpose of a Matrix (transpose.php):

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Transpose of a Matrix</title>
```

```
</head>
```

```
<body>
```

```
    <h1>Transpose of a Matrix</h1>
```

```
    <form method="post">
```

```
        <textarea name="matrix" rows="5" cols="20" placeholder="Enter matrix (comma separated rows)
e.g. 1,2,3;4,5,6" required></textarea><br>
```

```
        <button type="submit">Transpose</button>
```

```
    </form>
```

```
<?php
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

```
    $input = $_POST['matrix'];
```

```
    $rows = array_map('trim', explode(';', $input));
```

```
    $matrix = array_map(function($row) {
```

```
        return explode(',', $row);
```

```
    }, $rows);
```

```
    $transpose = array();
```

```
    for ($i = 0; $i < count($matrix[0]); $i++) {
```

```
        for ($j = 0; $j < count($matrix); $j++) {
```

```
            $transpose[$i][$j] = $matrix[$j][$i];
```

```

    }
}

echo "<h2>Transposed Matrix:</h2><pre>";
foreach ($transpose as $row) {
    echo implode(" ", $row) . "\n";
}
echo "</pre>";
}
?>
</body>
</html>

```

3. Multiplication of Two Matrices (multiply_matrices.php):

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Matrix Multiplication</title>
</head>
<body>
    <h1>Multiply Two Matrices</h1>
    <form method="post">
        <textarea name="matrix1" rows="5" cols="20" placeholder="Enter first matrix (comma separated rows) e.g. 1,2;3,4" required></textarea><br>
        <textarea name="matrix2" rows="5" cols="20" placeholder="Enter second matrix (comma separated rows) e.g. 1,0;0,1" required></textarea><br>
        <button type="submit">Multiply</button>
    </form>

```

```

</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $input1 = $_POST['matrix1'];
    $input2 = $_POST['matrix2'];
    $matrix1 = array_map(function($row) {
        return explode(',', $row);
    }, array_map('trim', explode(';', $input1)));
    $matrix2 = array_map(function($row) {
        return explode(',', $row);
    }, array_map('trim', explode(';', $input2)));
    $rows1 = count($matrix1);
    $cols1 = count($matrix1[0]);
    $rows2 = count($matrix2);
    $cols2 = count($matrix2[0]);
    if ($cols1 != $rows2) {
        echo "Error: Number of columns in the first matrix must equal the number of rows in the second matrix.";
        exit;
    }
    $result = array_fill(0, $rows1, array_fill(0, $cols2, 0));
    for ($i = 0; $i < $rows1; $i++) {
        for ($j = 0; $j < $cols2; $j++) {
            for ($k = 0; $k < $cols1; $k++) {
                $result[$i][$j] += $matrix1[$i][$k] * $matrix2[$k][$j];
            }
        }
    }
    echo "<h2>Resulting Matrix:</h2><pre>";
    foreach ($result as $row) {

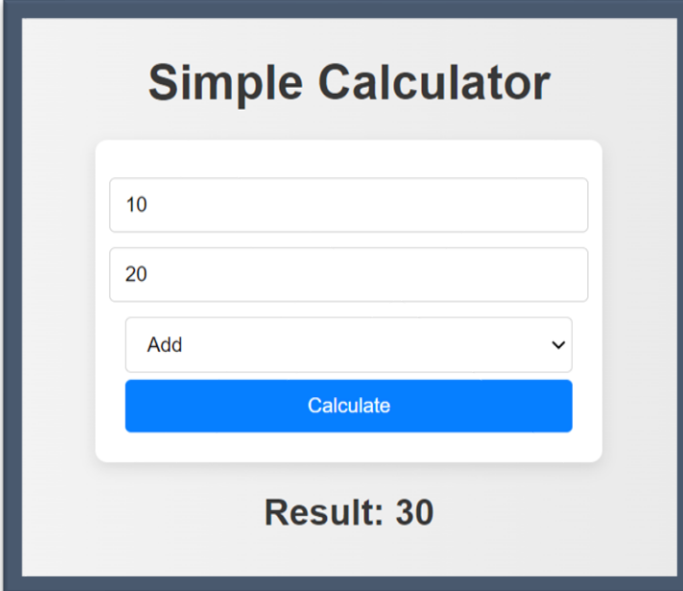
```

```
        echo implode(" ", $row) . "\n";
    }
    echo "</pre>";
}??>
</body>
</html>
```

Output:

- The calculator program allows the user to perform basic arithmetic operations and displays the result.
- The transpose program accepts a matrix input and displays its transpose.
- The multiplication program accepts two matrices and displays the result of their multiplication.

Calculator



Transpose Matrix

Transpose of a Matrix

Enter matrix (comma separated rows) e.g.
1,2,3;4,5,6

Transpose

Transposed Matrix:

1, 4
2, 5
3, 6

Multiplication Transpose Matrix

Multiply Two Matrices

Enter first matrix (comma separated
rows) e.g. 1,2;3,4

Enter second matrix (comma separated
rows) e.g. 1,0;0,1

Multiply

Resulting Matrix:

7, 10
15, 22

Result: These programs successfully demonstrate various mathematical operations using PHP, showcasing the ability to handle user input, perform calculations, and display results dynamically. This experiment highlights the versatility of PHP in web applications.

Lab Experiment 6: PHP Program to Implement File Upload and Download Operations

Objective: To create a PHP program that allows users to upload files to a server and download files from the server, demonstrating file handling capabilities in PHP.

Software Used:

- XAMPP (for local server)
- PHP (version 7.0 or higher)
- Text Editor - Visual Studio Code

Methodology/Procedure:

1. Set Up Environment:

- Install XAMPP/WAMP/LAMP on your computer.
- Start the local server and ensure Apache and MySQL services are running.

2. Create PHP Files:

- Create two PHP files:
 - upload.php (for file upload functionality)
 - download.php (for file download functionality)
- Save the files in the htdocs directory (for XAMPP) or the appropriate web directory for other servers.

3. Create a Directory for Uploads:

- Create a directory named uploads inside the htdocs folder to store uploaded files.

4. Write PHP Code:

- Implement file upload functionality in upload.php.
- Implement file download functionality in download.php.

5. Run the Programs:

- Open a web browser and navigate to each PHP file to test the functionality.

Description: This experiment demonstrates how to handle file uploads and downloads in PHP. It allows users to upload files to a server and provides the option to download those files.

Program Code

1. File Upload and Download (file_upload_download.php):

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>File Upload and Download</title>

    <style>

        body { font-family: Arial, sans-serif; margin: 20px; }

        input[type="file"], button { margin-top: 10px; }

        h2 { margin-top: 30px; }

        ul { list-style-type: none; padding: 0; }

    </style>

</head>

<body>

    <h1>File Upload and Download</h1>

    <!-- File Upload Section -->

    <h2>Upload a File</h2>

    <form method="post" enctype="multipart/form-data">

        <input type="file" name="fileToUpload" required>

        <button type="submit" name="upload">Upload</button>

    </form>

    <?php

    // File Upload Logic

    if (isset($_POST['upload'])) {

        $targetDir = "uploads/";
```



```

$targetFile = $targetDir . basename($_FILES["fileToUpload"]["name"]);

$uploadOk = 1;

$fileType = strtolower(pathinfo($targetFile, PATHINFO_EXTENSION));

// Check if file already exists
if (file_exists($targetFile)) {
    echo "Sorry, file already exists.<br>";
    $uploadOk = 0;
}

// Check file size (limit to 5MB)
if ($_FILES["fileToUpload"]["size"] > 5000000) {
    echo "Sorry, your file is too large.<br>";
    $uploadOk = 0;
}

// Allow certain file formats
if(!in_array($fileType, ['jpg', 'png', 'jpeg', 'gif', 'pdf', 'doc', 'docx'])) {
    echo "Sorry, only JPG, JPEG, PNG, GIF, PDF, DOC, and DOCX files are allowed.<br>";
    $uploadOk = 0;
}

// Attempt to upload the file
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.<br>";
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $targetFile)) {
        echo "The file " . htmlspecialchars(basename($_FILES["fileToUpload"]["name"])) . " has been
uploaded.<br>";
    } else {

```

```

        echo "Sorry, there was an error uploading your file.<br>";
    }
}
?>
<!-- File Download Section -->
<h2>Download Files</h2>
<ul>
    <?php
        $directory = "uploads/";
        if (is_dir($directory)) {
            if ($handle = opendir($directory)) {
                while (false !== ($file = readdir($handle))) {
                    if ($file != "." && $file != "..") {
                        echo "<li><a href='$directory$file' download>$file</a></li>";
                    }
                }
                closedir($handle);
            }
        } else {
            echo "No files found.";
        }
    ?>
</ul>
</body>
</html>

```

Output:

- The upload program allows users to select and upload files, displaying a success or error message based on the outcome.
- The download program lists all files in the uploads directory, providing links for users to download each file.

File Upload and Download

Upload a File

Choose File

No file chosen

Upload

Download Files

| File Name | Action |
|---|---------------------|
| Consultant Invoice Template_FY 24_25 (3).docx | <div>Download</div> |
| MCA_WT_strategic.pdf | <div>Download</div> |

Result: The programs successfully demonstrate file upload and download operations in PHP. The upload functionality checks for file size and type, ensuring only valid files are uploaded. The download functionality retrieves and lists files from the server, allowing users to download them easily.

Lab Experiment 7: PHP Program to Perform Operations on Cookies

Objective: To create a PHP program that allows users to add and delete cookies, demonstrating the use of cookies for storing user information on the client side.

Software Used:

- XAMPP (for local server)
- PHP (version 7.0 or higher)
- Text Editor - Visual Studio Code

Methodology/Procedure:

1. Set Up Environment:

- Install XAMPP/WAMP/LAMP on your computer.
- Start the local server and ensure Apache and MySQL services are running.

2. Create PHP File:

- Create a PHP file named `cookie_operations.php`.
- Save the file in the `htdocs` directory (for XAMPP) or the appropriate web directory for other servers.

3. Write PHP Code:

- Implement functionality for adding and deleting cookies.

4. Run the Program:

- Open a web browser and navigate to `http://localhost/cookie_operations.php` to test the functionality.

Program Code

1. Cookie Operations (`cookie_operations.php`):

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Cookie Operations</title>

    <style>
```

```

        body { font-family: Arial, sans-serif; margin: 20px; }

        input[type="text"], button { margin-top: 10px; }
    </style>
</head>
<body>
    <h1>Cookie Operations</h1>

    <!-- Add Cookie Section -->
    <h2>Add a Cookie</h2>
    <form method="post">
        <input type="text" name="cookie_name" placeholder="Cookie Name" required>
        <input type="text" name="cookie_value" placeholder="Cookie Value" required>
        <button type="submit" name="add_cookie">Add Cookie</button>
    </form>

    <?php
    // Add Cookie Logic
    if (isset($_POST['add_cookie'])) {
        $cookie_name = $_POST['cookie_name'];
        $cookie_value = $_POST['cookie_value'];
        $expiry_time = time() + (86400 * 30); // Cookie expires in 30 days

        setcookie($cookie_name, $cookie_value, $expiry_time, "/"); // "/" makes cookie available in the
        whole domain

        echo "Cookie '$cookie_name' has been added with value '$cookie_value'.<br>";
    }
    ?>

    <!-- Delete Cookie Section -->

```

```
<h2>Delete a Cookie</h2>
```

```
<form method="post">
```

```
    <input type="text" name="delete_cookie_name" placeholder="Cookie Name to Delete" required>
```

```
    <button type="submit" name="delete_cookie">Delete Cookie</button>
```

```
</form>
```

```
<?php
```

```
// Delete Cookie Logic
```

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

```
    $delete_cookie_name = $_POST['delete_cookie_name'];
```

```
    // Set the cookie with an expiry time in the past
```

```
    setcookie($delete_cookie_name, "", time() - 3600, "/");
```

```
    echo "Cookie '$delete_cookie_name' has been deleted.<br>";
```

```
}
```

```
?>
```

```
<!-- Display Current Cookies -->
```

```
<h2>Current Cookies</h2>
```

```
<ul>
```

```
    <?php
```

```
    if (count($_COOKIE) > 0) {
```

```
        foreach ($_COOKIE as $name => $value) {
```

```
            echo "<li>$name: $value</li>";
```

```
        }
```

```
    } else {
```

```
        echo "<li>No cookies found.</li>";
```

```
    }
```

```
?>
```

```
</ul>
</body>
</html>
```

Output:

- The program provides a section to add cookies by specifying a name and value, confirming the addition upon submission.
- It also allows users to delete a cookie by entering its name and displays confirmation of the deletion.
- A list of currently stored cookies is displayed on the page.

Add a Cookie

Input fields: Name (jdp), Value (123)

Button: Add Cookie

Message: Cookie '**jdp**' has been added with value '**123**'.

Delete a Cookie

Input field: Name (dp)

Button: Delete Cookie

Message: Cookie '**dp**' has been deleted.

Current Cookies

| |
|---------------------------------------|
| user: DP |
| PHPSESSID: va2qjg6fm9dqmobdo7cs7capbg |
| jdp: 123 |

Result:

The program successfully demonstrates the creation and deletion of cookies in PHP. It shows how to set a cookie with an expiration time and how to remove a cookie by setting its expiration time to a past value.

Lab Experiment 8: PHP Program to Implement Form Handling

Objective: To create a PHP program that demonstrates form handling techniques, including data validation and processing user input from an HTML form.

Software Used:

- XAMPP (for local server)
- PHP (version 7.0 or higher)
- Text Editor - Visual Studio Code

Methodology/Procedure:

1. Set Up Environment:

- Install XAMPP/WAMP/LAMP on your computer.
- Start the local server and ensure Apache and MySQL services are running.

2. Create PHP File:

- Create a PHP file named form_handling.php.
- Save the file in the htdocs directory (for XAMPP) or the appropriate web directory for other servers.

3. Write HTML and PHP Code:

- Implement a form that captures user input and processes it using PHP.

4. Run the Program:

- Open a web browser and navigate to http://localhost/form_handling.php to test the functionality.

Program Code

1. Form Handling (form_handling.php):

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>PHP Form Handling</title>

    <style>
```



```

body { font-family: Arial, sans-serif; margin: 20px; }

input[type="text"], input[type="email"], textarea { width: 100%; margin: 10px 0; padding: 10px; }

button { margin-top: 10px; padding: 10px 15px; }

.error { color: red; }

.success { color: green; }

</style>

</head>

<body>

<h1>PHP Form Handling</h1>

<?php

// Initialize variables and error messages

$name = $email = $message = "";

$nameErr = $emailErr = $messageErr = "";

$successMsg = "";

// Form handling logic

if ($_SERVER["REQUEST_METHOD"] == "POST") {

    // Validate Name

    if (empty($_POST["name"])) {

        $nameErr = "Name is required";

    } else {

        $name = htmlspecialchars(trim($_POST["name"]));

    }

    // Validate Email

    if (empty($_POST["email"])) {

        $emailErr = "Email is required";

    } else {

        $email = htmlspecialchars(trim($_POST["email"]));

    }

}

}

```

```

        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $emailErr = "Invalid email format";
        }
    }

    // Validate Message
    if (empty($_POST["message"])) {
        $messageErr = "Message is required";
    } else {
        $message = htmlspecialchars(trim($_POST["message"]));
    }

    // If no errors, process the form
    if (empty($nameErr) && empty($emailErr) && empty($messageErr)) {
        $successMsg = "Form submitted successfully!<br>Name: $name<br>Email: $email<br>Message: $message";

        // Reset form fields
        $name = $email = $message = "";
    }
}

?>

<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
    <input type="text" name="name" placeholder="Name" value="<?php echo $name; ?>">
    <span class="error"><?php echo $nameErr; ?></span>

    <input type="email" name="email" placeholder="Email" value="<?php echo $email; ?>">
    <span class="error"><?php echo $emailErr; ?></span>

```

```

        <textarea name="message" placeholder="Your Message" rows="5"><?php echo $message;
?></textarea>

        <span class="error"><?php echo $messageErr; ?></span>

        <button type="submit">Submit</button>

    </form>

    <div class="success"><?php echo $successMsg; ?></div>
</body>
</html>

```

Output:

- The program provides a form for the user to enter their name, email, and a message.
- It validates the inputs, showing error messages for any fields that are incorrectly filled out or left empty.
- Upon successful submission, it displays a success message with the entered information.

The screenshot shows a web form titled "PHP Form Handling". It contains three input fields: "Name", "Email", and "Your Message". Each field has a red error message below it: "Name is required", "Email is required", and "Message is required". At the bottom of the form is a blue "Submit" button.

Form Validation

PHP Form Handling

Your Message

Submit

Form submitted successfully!
Name: Durga Prasad Jyothula
Email: durgaprasad@gmail.com
Message: Hello

Result:

The program successfully demonstrates PHP form handling techniques, including data validation and sanitization. It processes user input and provides feedback, making it clear how form data is handled in PHP.