**Comprehensive Documentation for the Calculator Tool**

A complete description of the Calculator Tool exists within this documentation which runs through PHP with HTML alongside CSS. Through the tool users can execute basic math functions like addition and subtraction in addition to multiplication and division and perform complex operations with logarithms. The user interface in the application remains friendly while the forms demonstrate dynamic capabilities and the results page showcases a stylish appearance**.**

**1. Introduction**

The Calculator Tool functions as an online mathematical tool that enables users to run different arithmetic operations. Rephrase the following sentence. The application offers users a basic interface through which they can enter numbers along with operations before viewing results. The tool is implemented using:

• HTML for the user interface.

The PHP system manages the backend operations for computing alongside HTML provides frontend services.

The result page uses CSS for its presentation layout.

The system provides automatic form adjustments depending on the chosen operation so users can interact without interruption.

**2. Features**

The calculator tool offers functionality that includes three main features.

Basic Arithmetic Operations:

o Addition (+)

o Subtraction (-)

o Multiplication (\*)

o Division (/)

Advanced Mathematical Operations:

o Logarithm (log) – Natural logarithm (base e).

• Dynamic Form Behavior**:**

The calculator tool has a function which disables the entry of Number 2 input when users choose the Log button.Input Validation:

* + Ensures that inputs are numeric.
  + Handles edge cases like division by zero and invalid logarithm inputs.
* **Beautified Result Page:**
  + Displays the result in a visually appealing and user-friendly format.

**3. File Structure**

The program contains three files which form its structure:

calculator.html:

Users enter numbers through this main form alongside selecting their desired operation type.

calculator.php:

The backend script executes form data processing while performing mathematical operations before showing the output result.

3. CSS Styles:

The CSS styles are embedded directly in the calculator.php file for result page design.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4. Implementation Details**

4.1 HTML Form (Frontend)

Users enter the application through the form interface implemented in HTML. It includes:

• Two input fields for numbers (Number 1 and Number 2).

The user interface contains a selection field for choosing operations.

Users can submit the form using a single "Calculate" button.

Dynamic Behavior

A JavaScript script automatically conceals the Number 2 input field through its operation selection to logarithm.

The form adjusts automatically when users select an operation through a trigger from the on change event that activates toggleNumber2Input() to respond to the form.

**Code Snippet**

html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Calculator</title>

    <style>

        body {

            background: #121212;

            color: #ffffff;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

            margin: 0;

            font-family: Arial, sans-serif;

        }

        .calc-card {

            background: #1f1f1f;

            padding: 20px;

            border-radius: 15px;

            box-shadow: 0 5px 15px rgba(0, 0, 0, 0.7);

            width: 300px;

        }

        .display {

            background: #000;

            color: #0f0;

            padding: 15px;

            font-size: 24px;

            text-align: right;

            border-radius: 10px;

            margin-bottom: 10px;

        }

        input[type="number"] {

            width: 100%;

            padding: 12px;

            margin: 10px 0;

            font-size: 18px;

            background: #333;

            border: none;

            border-radius: 8px;

            color: white;

            text-align: right;

        }

        .button-grid {

            display: grid;

            grid-template-columns: repeat(4, 1fr);

            gap: 10px;

        }

        button {

            padding: 15px;

            font-size: 18px;

            background: #444;

            color: #fff;

            border: none;

            border-radius: 8px;

            cursor: pointer;

            transition: background 0.2s;

        }

        button:hover {

            background: #666;

        }

        .operation {

            background: #ff5722;

        }

        .operation:hover {

            background: #e64a19;

        }

        .clear {

            background: #d32f2f;

        }

        .clear:hover {

            background: #b71c1c;

        }

    </style>

</head>

<body>

    <form action="calculator.php" method="post">

        <h1>Calculator</h1><br>

        <label for="num1">Number 1:</label>

        <input type="text" id="num1" name="num1" required><br><br>

        <label for="num2">Number 2:</label>

        <input type="text" id="num2" name="num2" required><br><br>

        <div class="button-grid">

            <button type="submit" name="operation" value="add" class="operation">+</button>

            <button type="submit" name="operation" value="subtract" class="operation">-</button>

            <button type="submit" name="operation" value="multiply" class="operation">×</button>

            <button type="submit" name="operation" value="divide" class="operation">÷</button>

            <button type="submit" name="operation" value="log">log₁₀</button>

            <button type="reset" class="clear">C</button>

        </div>

        </select><br><br>

    </form>

    <script>

        toggleNumber2Input();

    </script>

</body>

</html>

**4.2 PHP Backend (Logic)**

The calculator.php script enables form submission and calculates results which are then displayed to users.

The script includes integrity checks to validate that only numeric values are submitted.

The script contains a switch block that deals with different calculation operations.

The script implements error management for special situations that include dividing by zero or entering incorrect values for logarithms.

**Code Snippet**

php

<?php

$num1 = $\_POST['num1'];

$operation = $\_POST['operation'];

if (!is\_numeric($num1)) {

    die("Invalid input. Please enter a numeric value for Number 1.");

}

if ($operation !== 'log') {

    $num2 = $\_POST['num2'];

    if (!is\_numeric($num2)) {

        die("Invalid input. Please enter a numeric value for Number 2.");

    }

}

switch ($operation) {

    case 'add':

        $result = $num1 + $num2;

        $operationSymbol = "+";

        break;

    case 'subtract':

        $result = $num1 - $num2;

        $operationSymbol = "-";

        break;

    case 'multiply':

        $result = $num1 \* $num2;

        $operationSymbol = "×";

        break;

    case 'divide':

        if ($num2 == 0) {

            die("Division by zero is not allowed.");

        }

        $result = $num1 / $num2;

        $operationSymbol = "÷";

        break;

    case 'log':

        if ($num1 <= 0) {

            die("Logarithm is only defined for positive numbers.");

        }

        $result = log($num1);

        $operationSymbol = "log";

        break;

    default:

        die("Invalid operation selected.");

}

echo "

<!DOCTYPE html>

<html lang='en'>

<head>

    <meta charset='UTF-8'>

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

    <title>Calculation Result</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #121212;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

            margin: 0;

        }

        .result-container {

            background-color: #ffffff;

            padding: 20px 40px;

            border-radius: 10px;

            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

            text-align: center;

        }

        h1 {

            color:rgb(151, 106, 99);

            font-size: 24px;

            margin-bottom: 20px;

        }

        .result {

            font-size: 28px;

            font-weight: bold;

            color: #007bff;

            margin: 20px 0;

        }

        .operation {

            font-size: 20px;

            color:rgb(151, 106, 99);

        }

        .back-link {

            display: inline-block;

            margin-top: 20px;

            padding: 10px 20px;

            background-color: #007bff;

            color: #fff;

            text-decoration: none;

            border-radius: 5px;

            transition: background-color 0.3s ease;

        }

        .back-link:hover {

            background-color: #0056b3;

        }

    </style>

</head>

<body>

    <div class='result-container'>

        <h1>Calculation Result</h1>

        <div class='operation'>

            Operation: $operationSymbol

        </div>

        <div class='result'>

            Result: $result

        </div>

        <a href='calculator.html' class='back-link'>Go Back</a>

    </div>

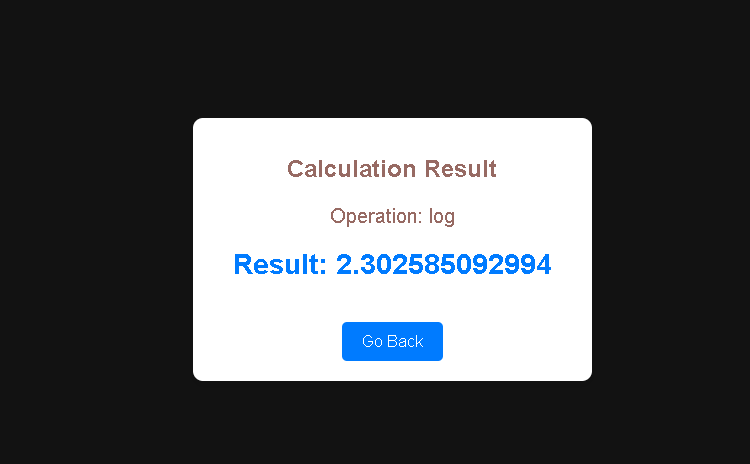
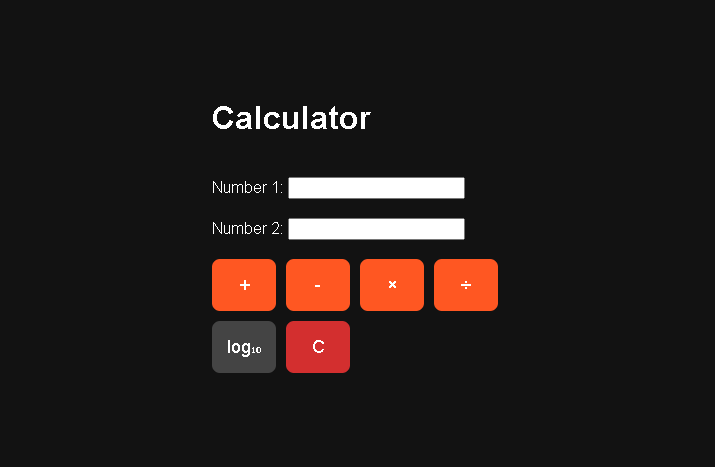
</body>

</html>

";

?>

**4.3 CSS Styling**

****

**5. Setup Instructions**

* Prerequisites:

A web server with PHP support (e.g., Apache, Nginx).

PHP serves as the essential programming language because the system requires version 7.0 or higher as its minimum requirement.

* Steps:

The HTML form code should be saved as calculator.html or calculator.php in one file.

The PHP backend code should receive the name calculator.php for file storage.

Place the two files at the web server root directory.

Access the application via a web browser (e.g., http://localhost/index.html).

6. Testing Instructions

* Basic Arithmetic Operations:

The user must submit two numbers and choose an operation from (+, -, \*, /).

The aching result needs to display exactly the same value as defined in the expected outcome.

* Logarithm Calculation:

Select the "Logarithm" operation.

Enter a positive number for Number 1.

Users should check that the displayed result shows the natural logarithm of the entered value. Error Handling:

Test invalid inputs (e.g., non-numeric values, division by zero, negative numbers for logarithms).

* + Ensure appropriate error messages are displayed.

**7. Error Handling**

The program deals with these errors during operation:

• Non-numeric inputs: Displays an error message.

• Division by zero: Displays an error message.

The application generates an error notice for all non-positive numerical inputs in logarithm functions.

**8. Future Enhancements**

The system needs to provide improved mathematical functionality which includes square root and exponentiation capabilities.

The PHP backend needs unit testing as part of its implementation procedure.

The feature to track previous calculations should be implemented as a history function.

The calculator's accessibility improves when the application receives ARIA labels that enable keyboard navigation.

**Conclusion**

Users can perform mathematical operations at various levels through the robust and convenient Versatile Calculator Tool application. Users find this application practical for math operations because it has responsive forms and input validation while presenting results in an elegant way. Users can deploy and use the application through step-by-step setup and testing protocol.