

Problem Statement 6: Write a javascript code to change the color of text according to user choice from the drop down list with color names.

Objective: To change the color of text with color name.

Description: Changing the color of text according to user choice.

1. **document.getElementById("elementId"):** This method retrieves an HTML element based on its id.
2. **element.value:** Retrieves the current value of a form element, in this case, the selected color from the dropdown.
3. **element.style.color:** Sets or retrieves the color of the text content of an element.

TAGS-

<script> - Embed or reference JavaScript code, either inline or from an external file.

ALGORITHM:

1. HTML Structure:

- 1.1. Declare HTML doctype.
- 1.2. Create HTML head with a title.
- 1.3. Inside the body:
 - 1.3.1. Add an h1 element with id "changeColorText".
 - 1.3.2. Add a label for the color selector dropdown.
 - 1.3.3. Include a select element with id "colorSelector" and onchange calling the function "changeTextColor".
 - 1.3.4. Populate the dropdown with option elements for different colors.

2. JavaScript Function (changeTextColor):

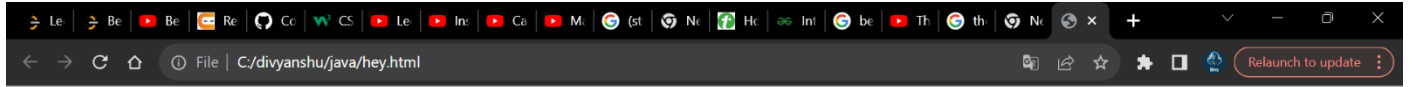
- 2.1. Define a function named changeTextColor.
- 2.2. Inside the function:
 - 2.2.1. Get the selected color from the dropdown using document.getElementById("colorSelector").value.
 - 2.2.2. Set the text color of the h1 element with id "changeColorText" to the selected color using document.getElementById("changeColorText").style.color = selectedColor.

CODE:

```
<!DOCTYPE html>
<html>
<head>
  <title>Text Color Changer</title>
</head>
<body>
  <h1 id="changeColorText">Choose a color:</h1>

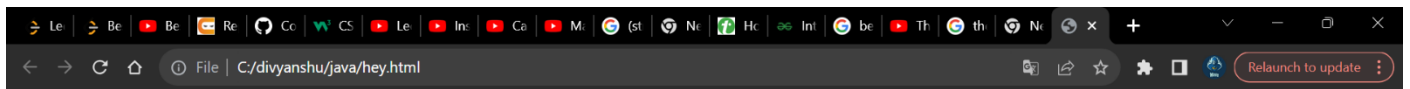
  <label for="colorSelector">Select color:</label>
  <select id="colorSelector" onchange="changeTextColor()">
    <option value="red">Red</option>
    <option value="green">Green</option>
    <option value="blue">Blue</option>
    <option value="yellow">Yellow</option>
  </select>

  <script>
    function changeTextColor()
    {
      var selectedColor = document.getElementById("colorSelector").value;
      document.getElementById("changeColorText").style.color = selectedColor;
    }
  </script>
</body>
</html>
```

OUTPUT:

Choose a color:

Select color:



Choose a color:

Select color:

Problem Statement 7: Write a javascript code to sort an array taken as input from the user.

Objective: To sort an array.

Description: Sorting an array taken as input from the user.

1. `prompt("Enter Element")`:

Function: Requests user input through a dialog box, used here to input array elements.

2. `console.log("Entered Array", a[i])`:

Function: Outputs information to the browser console, logging the message "Entered Array" along with the current array element `a[i]`.

3. `parseInt(a[i])`:

Function: Converts a string (`a[i]`) to an integer, essential for numerical operations like sorting.

ALGORITHM:

1. HTML Structure:

- 1.1. Declare HTML doctype.
- 1.2. Include an HTML head section (empty).
- 1.3. Inside the body:
 - 1.3.1. Add an `h1` element for the main heading.
 - 1.3.2. Add an `h2` element for a subheading related to sorting user-input arrays.
 - 1.3.3. Include paragraphs prompting the user to enter elements and displaying the sorted array.

2. JavaScript Code:

- 2.1. Declare an empty array `a``.
- 2.2. Set the array size (``size``) to 5 (or adjust as needed).
- 2.3. Use a loop to prompt the user to enter elements for the array.
- 2.4. Display each entered array element using `console.log`.
- 2.5. Use a nested loop to implement the bubble sort algorithm for sorting the array.
 - 2.5.1. Compare adjacent elements and swap them if necessary.
- 2.6. Display the sorted array using `console.log`.

CODE:

```
<!DOCTYPE html>
<html>

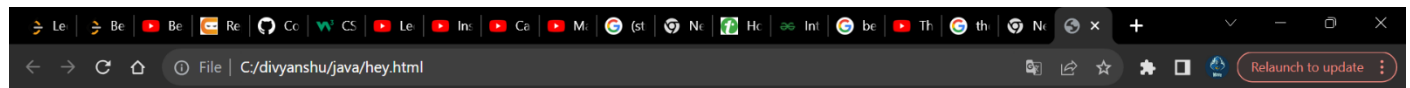
<head></head>

<body>
  <h1>JavaScript Arrays</h1>
  <h2>Sort User-Input Array</h2>

  <p>Enter elements of the array separated by commas:</p>
  <p id="sa">Sorted Array</p>
  <script>
    var a=[];
    var size=5;
    for(var i=0;i<size;i++)
    {a[i]=prompt("Enter Element");
    console.log("Entered Array",a[i])
    }
    for(i=0;i<size;i++)
    {
      for(var j=i+1;j<size;j++)
      {if(parseInt(a[i]) > parseInt(a[j]))
      {var temp=a[i];
      a[i]=a[j];
      a[j]=temp;
      }
      }
    }
    console.log("Sorted Array",a);

  </script>
</body>

</html>
```

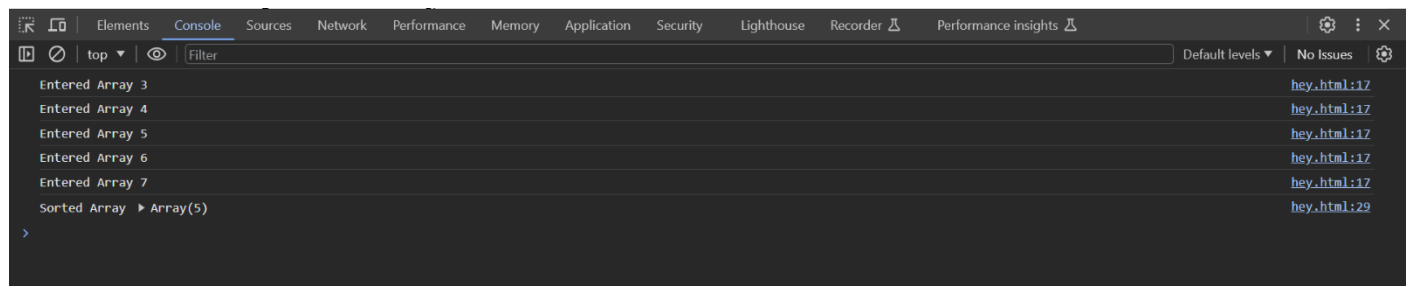
OUTPUT:

JavaScript Arrays

Sort User-Input Array

Enter elements of the array separated by commas:

Sorted Array



Problem Statement 8: Write a javascript code to validate registration from input fields.

Objective: To validate registration form input.

Description: Validating registration form input from the user.

1. `/^[^\\s@]+@[^\\s@]+\\.([^\\s@]+)$/:`

Regular Expression: Used for validating the email format. It ensures that the email has a valid structure.

2. `validateRegistration()`:

Function: Declares a JavaScript function named `validateRegistration` responsible for handling the form validation logic.

ALGORITHM:

1. HTML Structure:

1.1. Declare HTML doctype.

1.2. Include an HTML head section with a title.

1.3. Inside the body:

1.3.1. Add an `h1` element for the main heading.

1.3.2. Create a form with id "registrationForm" containing input fields for username, email, and password.

1.3.3. Include labels for each input field.

1.3.4. Add a button for user registration.

1.3.5. Display a paragraph element with id "validationMessage" for showing validation messages.

2. JavaScript Function (`validateRegistration`):

2.1. Define a function named `validateRegistration`.

2.2. Inside the function:

2.2.1. Get the values of the username, email, and password from the form.

2.2.2. Validate that all fields are non-empty.

2.2.3. Validate the email format using a regular expression.

2.2.4. Validate that the password has a minimum length of 8 characters.

2.2.5. Display appropriate validation messages based on the checks.

2.2.6. If all validations pass, display a success message.

CODE:

```
<!DOCTYPE html>
<html>
<head>
  <title>Registration Form</title>
</head>
<body>

  <h1>Registration Form</h1>

  <form id="registrationForm">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required>
    <br><br>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <br><br>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
    <br><br>

    <button type="button" onclick="validateRegistration()">Register</button>
  </form>

  <p id="validationMessage"></p>

  <script>
    function validateRegistration() {
      // Get form input values
      var username = document.getElementById("username").value;
      var email = document.getElementById("email").value;
      var password = document.getElementById("password").value;

      // Validate non-empty fields
      if (username === "" || email === "" || password === "") {
        document.getElementById("validationMessage").innerText = "All fields are required!";
        return;
      }

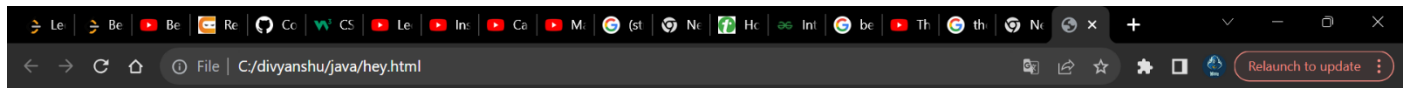
      // Validate email format using a regular expression
      var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!emailRegex.test(email)) {
        document.getElementById("validationMessage").innerText = "Invalid email format!";
        return;
      }

      // Validate password length
      if (password.length < 8) {
        document.getElementById("validationMessage").innerText = "Password must be at least 8
characters long.";
```



```
        return;
    }
    // If all validations pass, display success message
    document.getElementById("validationMessage").innerText = "Registration successful!";
}
</script>

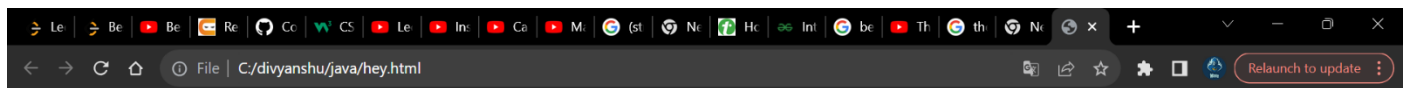
</body>
</html>
```

OUTPUT:

Registration Form

Username: Email: Password:

Password must be at least 8 characters long.



Registration Form

Username: Email: Password:

Registration successful!

Problem Statement 9: Write a javascript code to create a text content with welcome message in red color on button click. Take username as input.

Objective: To create welcome message.

Description: Creating a text with welcome message.

1. function generateWelcomeMessage() { ... }:

Function: Declares a JavaScript function named generateWelcomeMessage responsible for generating and displaying the welcome message.

2. color: red;

Style Property: A style property defined in the <style> section of the HTML. It sets the color of the text to red for the element with the id "welcomeText."

3. welcomeTextElement.style.color = "red";

Property: Sets the color style property of the "welcomeText" element to red.

ALGORITHM:

1. HTML Structure:

1.1. Declare HTML doctype.

1.2. Include an HTML head section with an embedded style to make the text red.

1.3. Inside the body:

1.3.1. Add an h1 element for the main heading.

1.3.2. Create a label and an input field for the user to enter their username.

1.3.3. Add a button to trigger the generation of the welcome message.

1.3.4. Display a paragraph element with id "welcomeText" to show the generated welcome message.

2. JavaScript Function (generateWelcomeMessage):

2.1. Define a function named generateWelcomeMessage.

2.2. Inside the function:

2.2.1. Get the username from the input field.

2.2.2. Access the paragraph element with id "welcomeText" and set its text content to a welcome message that includes the username.

2.2.3. Set the text color of the welcome message to red.

CODE:

```
<!DOCTYPE html>
<html>

<head>
  <style>
    /* Add a style to make the text red */
    #welcomeText {
      color: red;
    }
  </style>
</head>

<body>
  <h1>Welcome Message Generator</h1>

  <label for="username">Enter your username:</label>
  <input type="text" id="username" placeholder="Type your username">

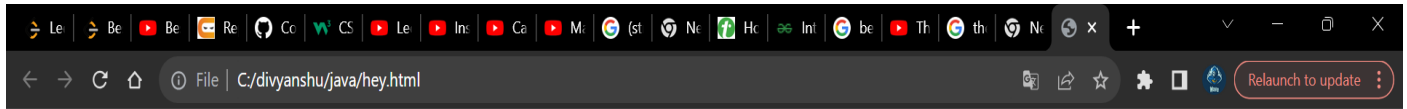
  <button onclick="generateWelcomeMessage()">Generate Welcome Message</button>

  <p id="welcomeText"></p>

  <script>
    function generateWelcomeMessage() {
      // Get the username from the input field
      var username = document.getElementById("username").value;

      // Display the welcome message in red color
      var welcomeTextElement = document.getElementById("welcomeText");
      welcomeTextElement.style.color = "red";
      welcomeTextElement.textContent = "Welcome, " + username + "!";
    }
  </script>
</body>

</html>
```

OUTPUT:

Welcome Message Generator

Enter your username:

Welcome, divyanshu!

Problem Statement 10: Write a javascript code to find whether a given string is 'Snowball String' or not.

Objective: To check whether a string is Snowball String or not.

Description: Checking a givrn string is Snowball String or not.

1. `inputString.split(' ')`:

Function: Splits the input string into an array of words using a space (' ') as the separator

2. `function checkSnowballString() { ... }`:

Function: Declares a JavaScript function named checkSnowballString responsible for checking if the input string is a Snowball String.

ALGORITHM:

1. HTML Structure:

1.1. Declare HTML doctype.

1.2. Inside the body:

1.2.1. Add an h1 element for the main heading.

1.2.2. Create a label and an input field for the user to enter a string.

1.2.3. Add a button to trigger the check for a Snowball string.

1.2.4. Display a paragraph element with id "result" to show the result of the check.

2. JavaScript Function (`checkSnowballString`):

2.1. Define a function named `checkSnowballString`.

2.2. Inside the function:

2.2.1. Get the input string from the input field.

2.2.2. Split the string into an array of words.

2.2.3. Check if the length of each word is in ascending order and consecutive.

2.2.4. Display the result indicating whether the given string is a Snowball String or not.

CODE:

```
<!DOCTYPE html>
<html>

<head></head>

<body>
  <h1>Snowball String Checker</h1>

  <label for="inputString">Enter a string:</label>
  <input type="text" id="inputString" placeholder="Type a string">

  <button onclick="checkSnowballString()">Check Snowball String</button>

  <p id="result"></p>

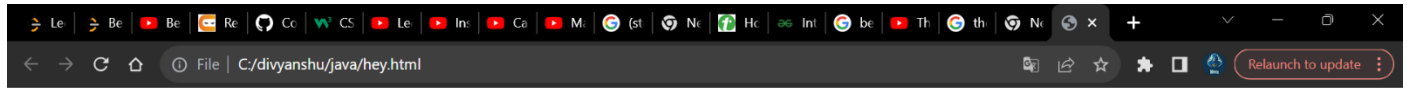
  <script>
    function checkSnowballString() {
      // Get the input string
      var inputString = document.getElementById("inputString").value;

      // Split the string into words
      var words = inputString.split(' ');

      // Check if the length of each word is in ascending order and consecutive
      var isSnowballString = true;
      for (var i = 0; i < words.length - 1; i++) {
        if (words[i].length + 1 !== words[i + 1].length) {
          isSnowballString = false;
          break;
        }
      }

      // Display the result
      var resultElement = document.getElementById("result");
      if (isSnowballString) {
        resultElement.textContent = "The given string is a Snowball String.";
      } else {
        resultElement.textContent = "The given string is not a Snowball String.";
      }
    }
  </script>
</body>

</html>
```

OUTPUT:

Snowball String Checker

Enter a string:

The given string is not a Snowball String.

Problem Statement 11: Write a PHP script to display the prime numbers from 1-100.

Objective: To display the prime numbers.

Description: Displaying prime numbers from 1-100.

1. <?php ... ?> Tags:

These tags are used to enclose the PHP code. Everything between <?php and ?> is interpreted as PHP code.

2. echo Statements:

echo "Prime numbers from 1 to 100 are: ";; Outputs a message to the screen.

echo \$i . " ";; Inside the loop, it prints each prime number followed by a space.

ALGORITHM:

1. isPrime Function:

Input: \$number (integer)

Output: true if \$number is prime, false otherwise

Algorithm:

1. If \$number is less than 2, return false (since 0 and 1 are not prime).
2. Use a for loop starting from 2 up to the square root of \$number.
3. Inside the loop, check if \$number is divisible evenly by \$i. If yes, return false, as it means \$number has a divisor other than 1 and itself.
4. If the loop completes without finding a divisor, return true.

2. Main Loop:

Loop from \$i = 1 to \$i = 100.

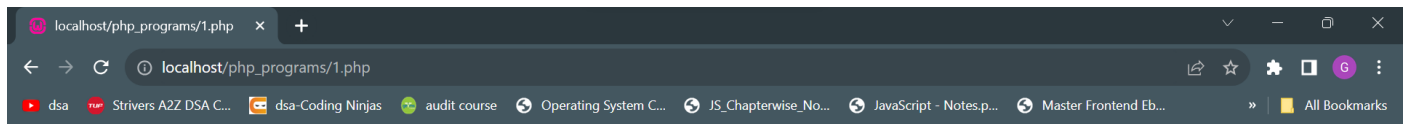
For each \$i, call the isPrime function.

If isPrime(\$i) returns true, print \$i followed by a space.

CODE:

```
<?php
function isPrime($number)
{
    if ($number < 2) {
        return false;
    }
    for ($i = 2; $i <= sqrt($number); $i++) {
        if ($number % $i == 0) {
            return false;
        }
    }
    return true;
}

echo "Prime numbers from 1 to 100 are: ";
for ($i = 1; $i <= 100; $i++) {
    if (isPrime($i)) {
        echo $i . " ";
    }
}
?>
```

OUTPUT:

Prime numbers from 1 to 100 are: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

Problem Statement 12: Write a PHP script to search a given digit in a number.

Objective: To search a given digit in a number.

Description: Searching a digit in a number.

ALGORITHM:

1. searchDigit Function:

Input: \$number (integer), \$digit (integer)

Output: true if \$digit is found in \$number, false otherwise

Algorithm:

1. Convert the \$number to a string to facilitate individual digit access.
2. Iterate through each digit in the number using a loop.
3. Inside the loop, check if the current digit is equal to the search \$digit.
4. If a match is found, return true.
5. If the loop completes without finding the \$digit, return false.

2. Example Usage:

Set an example \$number (e.g., \$number = 12345) and \$digitToSearch (e.g., \$digitToSearch = 3).

Call the searchDigit function with the example values.

Display the result based on whether the digit is found or not.

CODE:

```
<?php
function searchDigit($number, $digit)
{
    $numberString = (string)$number;

    for ($i = 0; $i < strlen($numberString); $i++)
    {
        if ($numberString[$i] == $digit) {
            return true; // Found the digit, return true
        }
    }

    return false;
}

$number = 34789;
$digitToSearch = 7;

if (searchDigit($number, $digitToSearch)) {
    echo "The digit $digitToSearch is found in the number $number.";
} else {
    echo "The digit $digitToSearch is not found in the number $number.";
}
?>
```

OUTPUT:



The digit 7 is found in the number 34789.

Problem Statement 13: Design an associative, array in PHP, with ‘Course’ as key and ‘Semester’ as value and perform following operations (using predefined functions):

- i) Sort this array with respect to the key, in ascending order.
- ii) Sort this array with respect to the value, in descending order.

Objective: To sort array in ascending and descending order.

Description: Sorting associative array in ascending and descending order.

ALGORITHM:

1. Initialize an Associative Array:
Create an associative array named \$courses with course names as keys and corresponding semester numbers as values.
2. Sort Array by Key in Ascending Order (ksort):
Use the ksort function to sort the array by keys in ascending order.
3. Print Sorted Array by Key:
Print the sorted array to display the courses sorted by key in ascending order.
4. Sort Array by Value in Descending Order (arsort):
Use the arsort function to sort the array by values in descending order.
5. Print Sorted Array by Value:
Print the sorted array to display the courses sorted by value in descending order

CODE:

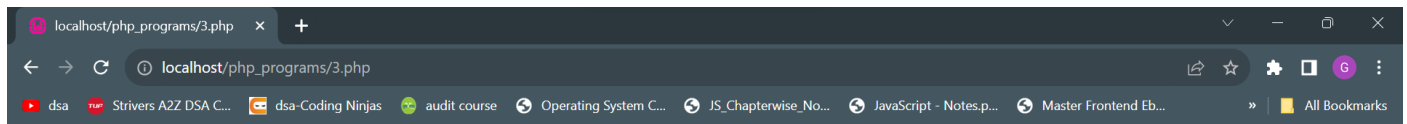
```
<?php
// Associative array with 'Course' as key and 'Semester' as value
$courses = array(
    'Math' => 3,
    'Physics' => 2,
    'History' => 1,
    'Computer Science' => 4
);

// i) Sort array by key in ascending order
ksort($courses);

echo "Sorted by key in ascending order:\n";
print_r($courses); echo"<br><br>";

// ii) Sort array by value in descending order
arsort($courses);

echo "\nSorted by value in descending order:\n";
print_r($courses);
?>
```

OUTPUT:

Sorted by key in ascending order: Array ([Computer Science] => 4 [History] => 1 [Math] => 3 [Physics] => 2)

Sorted by value in descending order: Array ([Computer Science] => 4 [Math] => 3 [Physics] => 2 [History] => 1)

Problem Statement 14: Write a PHP script to sort strings using user defined functions.

Objective: To sort strings using user defined functions

Description: Sorting strings.

ALGORITHM:

1. Define the customStringSort Function:
Input: An array of strings, \$strings.
Output: The array of strings sorted by length in ascending order.
2. Sort the Array using usort:
Use the usort function to sort the array of strings based on the length of each string.
The comparison function compares the lengths of two strings (\$a and \$b).
The result of the comparison determines the order in which the strings appear in the sorted array.
3. Example Usage:
Create an array of unsorted strings, \$unsortedStrings.
Print the unsorted strings.
4. Call the customStringSort Function:
Pass the unsorted array of strings to the customStringSort function.
Receive the sorted array of strings.
5. Print the Sorted Strings:
Print the sorted array of strings, now arranged by length in ascending order.

CODE:

```
<?php
function customStringSort($strings)
{
    usort($strings, function($a, $b) {
        return strlen($a) - strlen($b);
    });

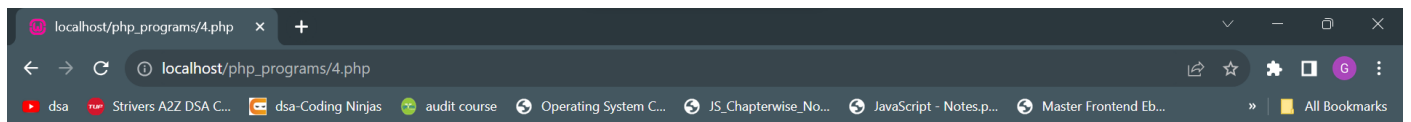
    return $strings;
}

// Example usage:
$unsortedStrings = array("Apple", "Banana", "Orange", "Grapes", "Cherry");

echo "Unsorted Strings:\n";
print_r($unsortedStrings); echo"<br><br>";

$sortedStrings = customStringSort($unsortedStrings);

echo "\nSorted Strings by Length:\n";
print_r($sortedStrings);
?>
```


OUTPUT:

Unsorted Strings: Array ([0] => Apple [1] => Banana [2] => Orange [3] => Grapes [4] => Cherry)

Sorted Strings by Length: Array ([0] => Apple [1] => Banana [2] => Orange [3] => Grapes [4] => Cherry)

Problem Statement 15 Write a PHP script that will give the following Validations on an HTML form :

- i) The name field should not be left blank.
- ii) The name field should not contain numbers and special characters.
- iii) Course can only be 'MCA', 'BCA' and 'BTECH'.
- iv) Email should contain '@' and '.' symbol.

Objective: To validate form input using database.

Description: Validating form input from the user using database.

ALGORITHM:

1. Define Validation Functions:
 1. validateName(\$name):
Check if the name is not left blank.
Check if the name contains only letters and spaces.
 2. validateCourse(\$course):
Check if the course is one of the allowed values ("MCA", "BCA", "BTECH").
 3. validateEmail(\$email):
Check if the email contains '@' and '.' symbols.
2. Form Submission Check:
Check if the form is submitted using the \$_SERVER["REQUEST_METHOD"] variable.
3. Get Form Data:
If the form is submitted, get the form data using \$_POST.
4. Validation:
Call the validation functions for each form field and store the error messages in variables (\$nameError, \$courseError, \$emailError).
5. Display Validation Errors or Success Message:
If any validation error messages are present, display them.
If there are no validation errors, display a success message.
6. HTML Form:
Create an HTML form with fields for name, course (as a dropdown), and email.
Use the method="post" attribute in the form tag to submit data to the same page.
Include labels and input/select elements for each form field.

CODE:

```
<?php
// Function to validate the name field
function validateName($name)
{
    // Check if the name is not left blank
    if (empty($name)) {
        return "Name cannot be left blank.";
    }

    // Check if the name contains only letters and spaces
    if (!preg_match("/^[a-zA-Z ]+$/", $name)) {
        return "Name should only contain letters and spaces.";
    }

    return "";
}

// Function to validate the course field
function validateCourse($course)
{
    // Check if the course is one of the allowed values
    $allowedCourses = array("MCA", "BCA", "BTECH");
    if (!in_array($course, $allowedCourses)) {
        return "Invalid course selection.";
    }

    return "";
}

// Function to validate the email field
function validateEmail($email)
{
    // Check if the email contains '@' and '.' symbols
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        return "Invalid email format.";
    }

    return "";
}

// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Get form data
    $name = $_POST["name"];
    $course = $_POST["course"];
    $email = $_POST["email"];

    // Validate form data
    $nameError = validateName($name);
    $courseError = validateCourse($course);
```

```
$emailError = validateEmail($email);

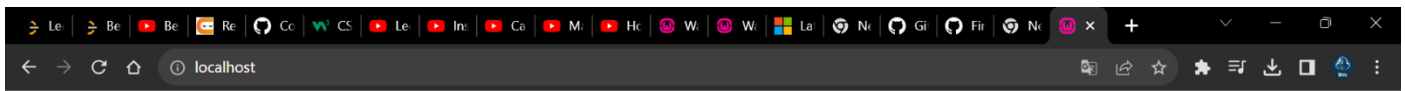
// Display validation errors or success message
if ($nameError || $courseError || $emailError) {
    echo "Validation Errors:\n";
    echo "- $nameError\n";
    echo "- $courseError\n";
    echo "- $emailError\n";
} else {
    echo "Form submitted successfully!";
}
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form Validation</title>
</head>
<body>
    <form method="post" action="">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name"><br>

        <label for="course">Course:</label>
        <select id="course" name="course">
            <option value="MCA">MCA</option>
            <option value="BCA">BCA</option>
            <option value="BTECH">BTECH</option>
        </select><br>

        <label for="email">Email:</label>
        <input type="text" id="email" name="email"><br>

        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

OUTPUT:

Form submitted successfully!

Name:

Course: ▼

Email:

Problem Statement 16: Write a PHP script to find the occurrence of 'is' in a string and replace it by 'and'.

Objective: To find the occurrence of word and replace with other.

Description: Finding the occurrence of word and replace with other.

ALGORITHM:

1. Define replaceIsWithAnd Function:
Input: \$inputString (string)
Output: \$outputString with 'is' replaced by 'and'
2. Replace 'is' with 'and':
Use the str_replace function to replace all occurrences of 'is' with 'and' in the \$inputString.
3. Example Usage:
Set an example \$inputString.
Print the original string.
4. Call the replaceIsWithAnd Function:
Pass the example \$inputString to the replaceIsWithAnd function.
Receive the modified string.
5. Print the Modified String:
Print the modified string, now with 'is' replaced by 'and'.

CODE:

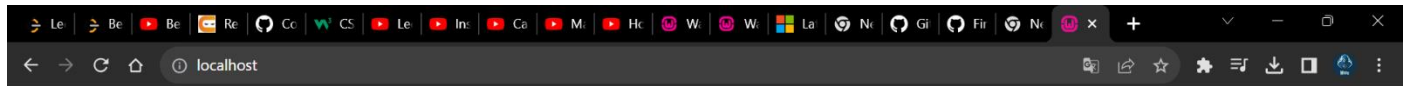
```
<?php
// Function to replace 'is' with 'and'
function replaceIsWithAnd($inputString)
{
    $outputString = str_replace('is', 'and', $inputString);
    return $outputString;
}

// Example usage:
$inputString = "This is a sample string. It is used for demonstration.";

echo "Original String:\n";
echo $inputString . "\n";echo"<br><br>";

$outputString = replaceIsWithAnd($inputString);

echo "\nString after replacing 'is' with 'and':\n";
echo $outputString . "\n";echo"<br><br>";
?>
```

OUTPUT:

Original String: This is a sample string. It is used for demonstration.

String after replacing 'is' with 'and': Thand and a sample string. It and used for demonstration.

Problem Statement 17: Write a PHP script that will create a Database 'Hospital' and perform following operations :

- i) Create a table 'Doctors' and put all these details : Id, Name, Specialization, Date_of_joining, Total_experience, Age, Contact_number.
- ii) Create table 'Rooms'/'Wards' and put all these details : Type, Total_no_of_beds, charges_per_day, number_of_pateints.
- iii) Insert values in these 2 tables using forms.

The fields should be validated as :

- i) No field should be left blank.
- ii) Fields like age,no_of_beds, etc, should only contain numeric values.
- iii) The no_of_beds should not exceed 20.
- iv) The room should be either 'general', 'ICU' or 'private'.
- v) The specialization should be 'physician', 'surgeon', 'dentist', etc.
- vi) Show the details of all the doctors and rooms.
- vii) Based on specialization, search the details of the doctors and display.

Objective: To handle database.

Description: Handling database with some operations.

ALGORITHM:

1. Database Setup:
Create the Hospital database and Doctors/Rooms tables if they don't exist.
2. Validation Functions:
Define functions to validate different input fields.
3. Form Submission:
When the user submits the form:
 1. Validate and insert doctor information into the Doctors table.
 2. Validate and insert room information into the Rooms table.
4. Display Data:
Display details of all doctors and rooms.
Display doctors based on a specified specialization.
5. Close Connection:
Close the database connection

CODE:**Backe_end.php**

```
<?php
// Database connection parameters
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Hospital";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create the database
$sql = "CREATE DATABASE IF NOT EXISTS $dbname";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully<br>";
} else {
    echo "Error creating database: " . $conn->error . "<br>";
}

// Select the database
$conn->select_db($dbname);

// Create 'Doctors' table
$sql = "CREATE TABLE IF NOT EXISTS Doctors (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Specialization VARCHAR(255) NOT NULL,
    Date_of_joining DATE NOT NULL,
    Total_experience INT NOT NULL,
    Age INT NOT NULL,
    Contact_number VARCHAR(15) NOT NULL
)";
$conn->query($sql);

// Create 'Rooms' table
$sql = "CREATE TABLE IF NOT EXISTS Rooms (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Type VARCHAR(20) NOT NULL,
    Total_no_of_beds INT NOT NULL,
    Charges_per_day INT NOT NULL,
    Number_of_patients INT NOT NULL
)";
$conn->query($sql);

// Function to validate numeric fields
```

```
function validateNumeric($value)
```

```
{  
    return is_numeric($value);  
}
```

```
// Function to validate age and number of beds
```

```
function validateAgeOrBeds($value)
```

```
{  
    return (is_numeric($value) && $value > 0 && $value <= 20);  
}
```

```
// Function to validate room type
```

```
function validateRoomType($value)
```

```
{  
    $allowedTypes = array("general", "ICU", "private");  
    return in_array(strtolower($value), $allowedTypes);  
}
```

```
// Function to validate specialization
```

```
function validateSpecialization($value)
```

```
{  
    $allowedSpecializations = array("physician", "surgeon", "dentist");  
    return in_array(strtolower($value), $allowedSpecializations);  
}
```

```
// Function to insert data into 'Doctors' table
```

```
function insertDoctor($conn, $name, $specialization, $dateOfJoining, $totalExperience, $age,  
$contactNumber)
```

```
{  
    $sql = "INSERT INTO Doctors (Name, Specialization, Date_of_joining, Total_experience, Age,  
Contact_number)  
VALUES ('$name', '$specialization', '$dateOfJoining', $totalExperience, $age, '$contactNumber')";  
    return $conn->query($sql);  
}
```

```
// Function to insert data into 'Rooms' table
```

```
function insertRoom($conn, $type, $totalBeds, $chargesPerDay, $numPatients)
```

```
{  
    $sql = "INSERT INTO Rooms (Type, Total_no_of_beds, Charges_per_day, Number_of_patients)  
VALUES ('$type', $totalBeds, $chargesPerDay, $numPatients)";  
    return $conn->query($sql);  
}
```

```
// Handle form submissions
```

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

```
    // Validate and insert data into 'Doctors' table
```

```
    $doctorName = $_POST["doctor_name"];  
    $doctorSpecialization = $_POST["doctor_specialization"];  
    $doctorJoiningDate = $_POST["doctor_joining_date"];  
    $doctorExperience = $_POST["doctor_experience"];  
    $doctorAge = $_POST["doctor_age"];  
    $doctorContactNumber = $_POST["doctor_contact_number"];
```

```
    if (
```

```
!empty($doctorName) && validateSpecialization($doctorSpecialization) &&
!empty($doctorJoiningDate) && validateNumeric($doctorExperience) &&
validateAgeOrBeds($doctorAge) && !empty($doctorContactNumber)
) {
    insertDoctor($conn, $doctorName, $doctorSpecialization, $doctorJoiningDate, $doctorExperience,
$doctorAge, $doctorContactNumber);
    echo "Doctor details inserted successfully<br>";
} else {
    echo "Invalid data for inserting doctor details";
}

// Validate and insert data into 'Rooms' table
$roomType = $_POST["room_type"];
$roomBeds = $_POST["room_beds"];
$roomCharges = $_POST["room_charges"];
$roomPatients = $_POST["room_patients"];

if (
    validateRoomType($roomType) && validateAgeOrBeds($roomBeds) &&
    validateNumeric($roomCharges) && validateNumeric($roomPatients)
) {
    insertRoom($conn, $roomType, $roomBeds, $roomCharges, $roomPatients);
    echo "Room details inserted successfully<br>";
} else {
    echo "Invalid data for inserting room details<br>";
}
}

// Display all doctors and rooms
echo "<br>Details of all Doctors:<br>";
$result = $conn->query("SELECT * FROM Doctors");
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        print_r($row);
    }
} else {
    echo "No records found in Doctors table<br>";
}

echo "<br>Details of all Rooms:<br>";
$result = $conn->query("SELECT * FROM Rooms");
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        print_r($row);
    }
} else {
    echo "No records found in Rooms table<br>";
}

// Display doctors based on specialization
echo "<br>Search Doctors by Specialization (e.g., physician):<br>";
$specializationToSearch = "physician";
$result = $conn->query("SELECT * FROM Doctors WHERE Specialization='$specializationToSearch'");
if ($result->num_rows > 0) {
```

```
while ($row = $result->fetch_assoc()) {  
    print_r($row);  
}  
} else {  
    echo "No records found for the specialization: $specializationToSearch<br>";  
}  
  
// Close connection  
$conn->close();  
?>
```

Index.html

```
<!DOCTYPE html>  
<html lang="en">  
  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Hospital Management System</title>  
</head>  
  
<body>  
    <h2>Doctor Information</h2>  
    <form method="post" action="backend.php">  
        <label for="doctor_name">Doctor Name:</label>  
        <input type="text" name="doctor_name" required>  
        <br>  
  
        <label for="doctor_specialization">Specialization:</label>  
        <input type="text" name="doctor_specialization" required>  
        <br>  
  
        <label for="doctor_joining_date">Joining Date:</label>  
        <input type="date" name="doctor_joining_date" required>  
        <br>  
  
        <label for="doctor_experience">Total Experience (in years):</label>  
        <input type="number" name="doctor_experience" required>  
        <br>  
  
        <label for="doctor_age">Age:</label>  
        <input type="number" name="doctor_age" required>  
        <br>  
  
        <label for="doctor_contact_number">Contact Number:</label>  
        <input type="text" name="doctor_contact_number" required>  
        <br>  
  
    <h2>Room Information</h2>  
    <label for="room_type">Room Type:</label>  
    <input type="text" name="room_type" required>  
    <br>  
  
    <label for="room_beds">Total Number of Beds:</label>
```

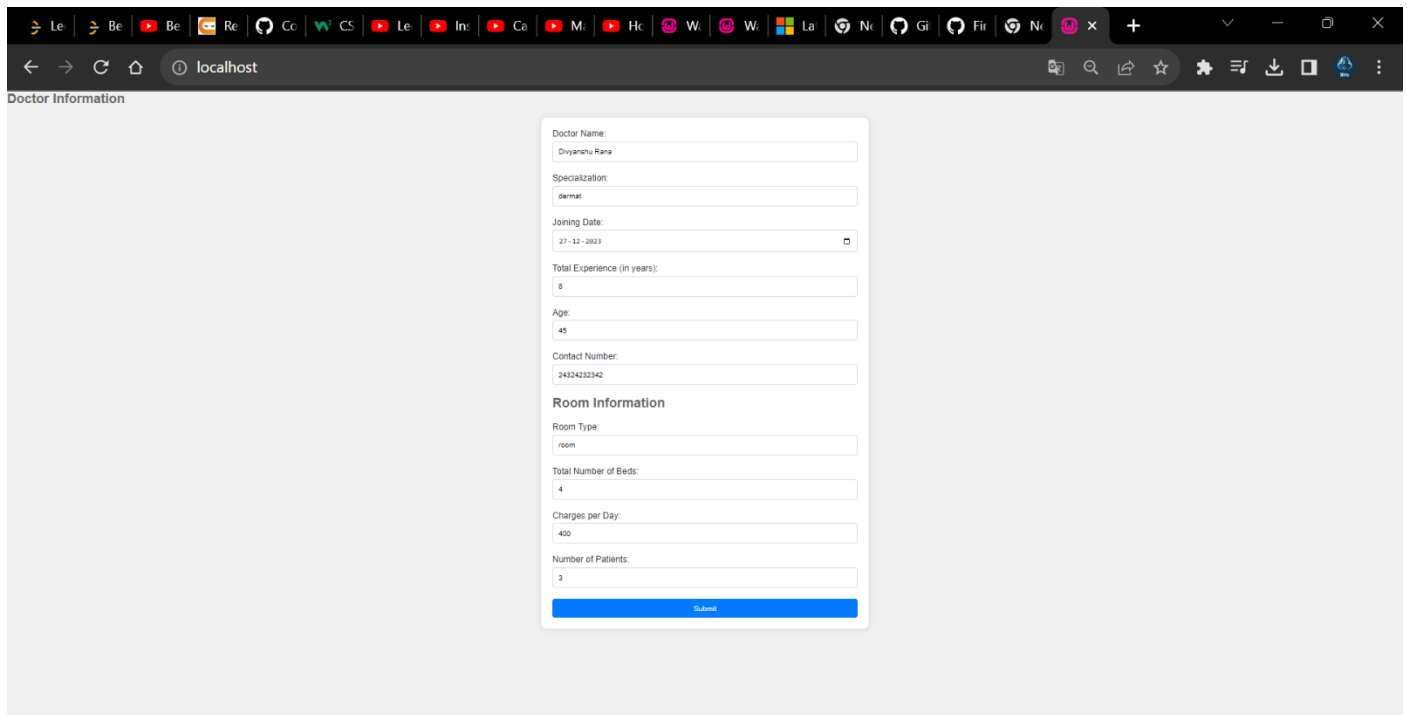
```
<input type="number" name="room_beds" required>
<br>

<label for="room_charges">Charges per Day:</label>
<input type="number" name="room_charges" required>
<br>
<label for="room_patients">Number of Patients:</label>
<input type="number" name="room_patients" required>
<br>

<input type="submit" name="submit" value="Submit">
</form>
</body>

</html>
```

OUTPUT:



The screenshot shows a web browser window with a dark theme. The address bar displays 'localhost'. The page content is a form titled 'Doctor Information' and 'Room Information'. The form is divided into two sections: 'Doctor Information' and 'Room Information'. The 'Doctor Information' section contains fields for Doctor Name, Specialization, Joining Date, Total Experience (in years), Age, and Contact Number. The 'Room Information' section contains fields for Room Type, Total Number of Beds, Charges per Day, and Number of Patients. A blue 'Submit' button is located at the bottom of the form.

Doctor Information

Doctor Name:
Divyanshu Rana

Specialization:
dermat

Joining Date:
27-12-2023

Total Experience (in years):
8

Age:
45

Contact Number:
2432423242

Room Information

Room Type:
room

Total Number of Beds:
4

Charges per Day:
400

Number of Patients:
2

Submit

Problem Statement 18: Write all the steps required to create a React App.

Objective: To write all the steps required to create a React App.

Description: Writing the steps required to create a React App.

1. Install Node.js and npm:

- Make sure to have Node.js and npm (Node Package Manager) installed on machine. Download and install from the official website: [Node.js](https://nodejs.org/).

2. Install Create React App:

- Open terminal or command prompt.
- Run the following command to install Create React App globally:
`npm install -g create-react-app`

3. Create a New React App:

- Run the following command to create a new React app:
`npx create-react-app my-react-app`
Replace "my-react-app" with the desired name of React app.

4. Navigate to the App Directory:

- Change into the newly created app directory:
`cd my-react-app`

5. Run the React App:

- Start the development server by running:
`npm start`
- This will open new React app in a web browser at `http://localhost:3000/`. We can see the live changes as we edit our code.

6. Understand Project Structure:

- Familiarize with the project structure. Key directories include:
src: Contains React application source code.
public: Contains the HTML file and other static assets.

7. Start Coding:

- Open the src directory and explore the index.js file. This is the entry point for your React application.
- Edit the existing components or create new ones in the src directory.
- Save your changes, and the development server will automatically reload the app.

8. Learn and Explore:

- Familiarize with React concepts like components, state, and props.
- Explore the [React documentation](https://reactjs.org/docs/getting-started.html) for in-depth information on React features.

9. Build and Deploy (Optional):

- When ready to deploy app, build a production-ready bundle using:
`npm run build`
- The build output will be located in the build directory. Then deploy this output to a web server or hosting service.

10. Enjoy Developing:

- Now set up is ready with a basic React app. Continue to enhance app by adding components, managing state, and exploring additional libraries and tools as needed.

Problem Statement 19: Design a React App to create a component Person with props. Render the component.

Objective: To create a component Person with props.

Description: Creating a component Person with props.

ALGORITHM:

1. Start:
The application begins execution.
2. Person Component:
Function Person is defined as a React functional component.
It takes in props and renders a div containing a heading (h2) for the name, and paragraphs (p) for age and occupation.
3. App Component:
Function App is defined as a React functional component.
Sample person data (personData) is created with a name, age, and occupation.
The component renders a div with the class name "App" containing an h1 heading for "React Person App."
It renders an instance of the Person component, passing the sample person data as props.
4. Export:
The App component is exported as the default export for use in other parts of the application.

CODE:**App.js**

```
import React from 'react';
import './App.css';

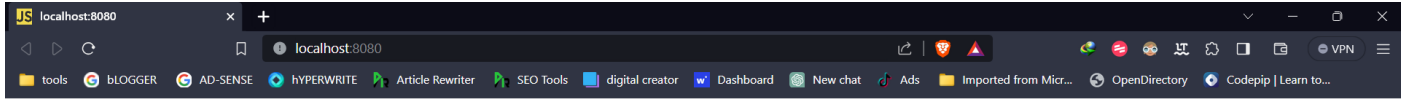
// Person component
const Person = (props) => {
  return (
    <div>
      <h2>{props.name}</h2>
      <p>Age: {props.age}</p>
      <p>Occupation: {props.occupation}</p>
    </div>
  );
};

function App() {
  // Sample person data
  const personData = {
    name: 'Divyanshu Rana',
    age: 24,
    occupation: 'Software Engineer',
  };

  return (
    <div className="App">
      <h1>React Person App</h1>
      { /* Render the Person component with props */ }
      <Person
        name={personData.name}
        age={personData.age}
        occupation={personData.occupation}
      />
    </div>
  );
}

export default App;
```

OUTPUT:



React Person App

Divyanshu Rana

Age: 24

Occupation: Software Engineer|

Problem Statement 20: Design a React App to create a class component 'Student' with constructor. Render the class component.

Objective: To render class components.

Description: Rendering the class components.

ALGORITHM:

1. Student Class Component:
Create a class component named Student that extends Component.
In the constructor, call the parent constructor using super(props).
Initialize the component state with default values for name, age, and major.
Define the render method to display the student information using JSX.
 - a. Render an h2 heading for "Student Information."
 - b. Display the name, age, and major using the values from the component's state.
2. App Function Component:
Create a functional component named App.
Return a JSX structure within a div with the class name "App."
 - a. Render an h1 heading for "React Student App."
 - b. Render an instance of the Student class component.
3. Rendering:
The App component is rendered by the root of the React application.
Within the App component, an instance of the Student class component is rendered.
4. Export:
Export the App component as the default export.

CODE:**App.js**

```
import React, { Component } from 'react';
import './App.css';

// Student class component
class Student extends Component {
  constructor(props) {
    super(props);

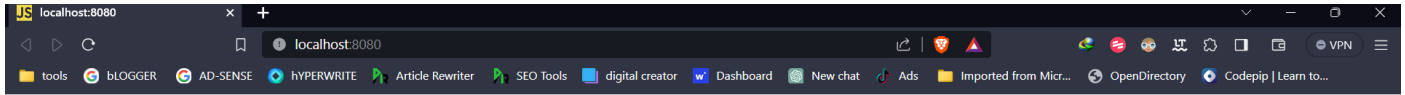
    // Initialize state in the constructor
    this.state = {
      name: 'Divyanshu Rana',
      age: 25,
      major: 'Computer Science',
    };
  }

  render() {
    return (
      <div>
        <h2>Student Information</h2>
        <p>Name: {this.state.name}</p>
        <p>Age: {this.state.age}</p>
        <p>Major: {this.state.major}</p>
      </div>
    );
  }
}

function App() {
  return (
    <div className="App">
      <h1>React Student App</h1>
      { /* Render the Student class component */ }
      <Student />
    </div>
  );
}

export default App;
```

OUTPUT:



React Student App

Student Information

Name: Divyanshu Rana

Age: 25

Major: Computer Science

Problem Statement 21: Create a Web Site implementing HTML, CSS, JavaScript and PHP.

Objective: To create a website.

Description: Creating a website using HTML,CSS,JavaScript and PHP.

CODE:

Login.php

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

// Check if the user is already logged in, if yes then redirect him to welcome page
if(isset($_SESSION["loggedin"]) && $_SESSION["loggedin"] === true){
    header("location: welcome.php");
    exit;
}

// Include config file
require_once "config.php";

// Define variables and initialize with empty values
$username = $password = "";
$username_err = $password_err = $login_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){

    // Check if username is empty
    if(empty(trim($_POST["username"]))) {
        $username_err = "Please enter username.";
    } else {
        $username = trim($_POST["username"]);
    }

    // Check if password is empty
    if(empty(trim($_POST["password"]))) {
        $password_err = "Please enter your password.";
    } else {
        $password = trim($_POST["password"]);
    }

    // Validate credentials
    if(empty($username_err) && empty($password_err)){
        // Prepare a select statement
        $sql = "SELECT id, username, password FROM users WHERE username = ?";

        if($stmt = mysqli_prepare($link, $sql)){
            // Bind variables to the prepared statement as parameters
            mysqli_stmt_bind_param($stmt, "s", $param_username);
```

```
// Set parameters
$param_username = $username;
// Attempt to execute the prepared statement
if(mysqli_stmt_execute($stmt)){
    // Store result
    mysqli_stmt_store_result($stmt);

    // Check if username exists, if yes then verify password
    if(mysqli_stmt_num_rows($stmt) == 1){
        // Bind result variables
        mysqli_stmt_bind_result($stmt, $id, $username, $hashed_password);
        if(mysqli_stmt_fetch($stmt)){
            if(password_verify($password, $hashed_password)){
                // Password is correct, so start a new session
                session_start();

                // Store data in session variables
                $_SESSION["loggedin"] = true;
                $_SESSION["id"] = $id;
                $_SESSION["username"] = $username;

                // Redirect user to welcome page
                header("location: welcome.php");
            } else{
                // Password is not valid, display a generic error message
                $login_err = "Invalid username or password.";
            }
        }
    } else{
        // Username doesn't exist, display a generic error message
        $login_err = "Invalid username or password.";
    }
} else{
    echo "Oops! Something went wrong. Please try again later.";
}

// Close statement
mysqli_stmt_close($stmt);
}

// Close connection
mysqli_close($link);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        body{ font: 14px sans-serif; }
```

```

.wrapper{ width: 360px; padding: 20px; }
</style>
</head>
<body>
<div class="wrapper">
  <h2>Login</h2>
  <p>Please fill in your credentials to login.</p>

  <?php
  if(!empty($login_err)){
    echo '<div class="alert alert-danger">' . $login_err . '</div>';
  }
  ?>

  <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
    <div class="form-group">
      <label>Username</label>
      <input type="text" name="username" class="form-control <?php echo (!empty($username_err)) ?
'is-invalid' : ''; ?>" value="<?php echo $username; ?>">
      <span class="invalid-feedback"><?php echo $username_err; ?></span>
    </div>
    <div class="form-group">
      <label>Password</label>
      <input type="password" name="password" class="form-control <?php echo
(!empty($password_err)) ? 'is-invalid' : ''; ?>">
      <span class="invalid-feedback"><?php echo $password_err; ?></span>
    </div>
    <div class="form-group">
      <input type="submit" class="btn btn-primary" value="Login">
    </div>
    <p>Don't have an account? <a href="register.php">Sign up now</a>.</p>
  </form>
</div>
</body>
</html>

```

Register.php

```

<?php
// Include config file
require_once "config.php";

// Define variables and initialize with empty values
$username = $password = $confirm_password = "";
$username_err = $password_err = $confirm_password_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){

  // Validate username
  if(empty(trim($_POST["username"]))) {
    $username_err = "Please enter a username.";
  } elseif(!preg_match('/^[a-zA-Z0-9_]+$/', trim($_POST["username"]))) {
    $username_err = "Username can only contain letters, numbers, and underscores.";
  }
}

```



```
} else{
// Prepare a select statement
$sql = "SELECT id FROM users WHERE username = ?";
if($stmt = mysqli_prepare($link, $sql)){
    // Bind variables to the prepared statement as parameters
    mysqli_stmt_bind_param($stmt, "s", $param_username);

    // Set parameters
    $param_username = trim($_POST["username"]);

    // Attempt to execute the prepared statement
    if(mysqli_stmt_execute($stmt)){
        /* store result */
        mysqli_stmt_store_result($stmt);

        if(mysqli_stmt_num_rows($stmt) == 1){
            $username_err = "This username is already taken.";
        } else{
            $username = trim($_POST["username"]);
        }
    } else{
        echo "Oops! Something went wrong. Please try again later.";
    }

    // Close statement
    mysqli_stmt_close($stmt);
}

// Validate password
if(empty(trim($_POST["password"]))) {
    $password_err = "Please enter a password.";
} elseif(strlen(trim($_POST["password"])) < 6){
    $password_err = "Password must have atleast 6 characters.";
} else{
    $password = trim($_POST["password"]);
}

// Validate confirm password
if(empty(trim($_POST["confirm_password"]))) {
    $confirm_password_err = "Please confirm password.";
} else{
    $confirm_password = trim($_POST["confirm_password"]);
    if(empty($password_err) && ($password != $confirm_password)){
        $confirm_password_err = "Password did not match.";
    }
}

// Check input errors before inserting in database
if(empty($username_err) && empty($password_err) && empty($confirm_password_err)){

    // Prepare an insert statement
    $sql = "INSERT INTO users (username, password) VALUES (?, ?)";
```

```
if($stmt = mysqli_prepare($link, $sql)){
    // Bind variables to the prepared statement as parameters
    mysqli_stmt_bind_param($stmt, "ss", $param_username, $param_password);
// Set parameters
    $param_username = $username;
    $param_password = password_hash($password, PASSWORD_DEFAULT); // Creates a password
hash

    // Attempt to execute the prepared statement
    if(mysqli_stmt_execute($stmt)){
        // Redirect to login page
        header("location: login.php");
    } else{
        echo "Oops! Something went wrong. Please try again later.";
    }

    // Close statement
    mysqli_stmt_close($stmt);
}

// Close connection
mysqli_close($link);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sign Up</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        body{ font: 14px sans-serif; }
        .wrapper{ width: 360px; padding: 20px; }
    </style>
</head>
<body>
    <div class="wrapper">
        <h2>Sign Up</h2>
        <p>Please fill this form to create an account.</p>
        <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
            <div class="form-group">
                <label>Username</label>
                <input type="text" name="username" class="form-control <?php echo (!empty($username_err)) ?
'is-invalid' : ''; ?>" value="<?php echo $username; ?>">
                <span class="invalid-feedback"><?php echo $username_err; ?></span>
            </div>
            <div class="form-group">
                <label>Password</label>
                <input type="password" name="password" class="form-control <?php echo
(!empty($password_err)) ? 'is-invalid' : ''; ?>" value="<?php echo $password; ?>">
                <span class="invalid-feedback"><?php echo $password_err; ?></span>
            </div>
```

```

<div class="form-group">
  <label>Confirm Password</label>
  <input type="password" name="confirm_password" class="form-control <?php echo
(!empty($confirm_password_err)) ? 'is-invalid' : ''; ?>" value="<?php echo $confirm_password; ?>">
  <span class="invalid-feedback"><?php echo $confirm_password_err; ?></span>
</div>
<div class="form-group">
  <input type="submit" class="btn btn-primary" value="Submit">
  <input type="reset" class="btn btn-secondary ml-2" value="Reset">
</div>
<p>Already have an account? <a href="login.php">Login here</a>.</p>
</form>
</div>
</body>
</html>

```

Welcome.php

```

<?php
// Initialize the session
session_start();

// Check if the user is logged in, if not then redirect him to login page
if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true){
  header("location: login.php");
  exit;
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Welcome</title>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <style>
    body{ font: 14px sans-serif; text-align: center; }
  </style>
</head>
<frameset rows = "8%, 79%, *">
  <frameset>
    <frame name="header" src="header.html" scrolling="off">
  </frameset>

  <frameset cols="15%,70%,10%" frameborder>
    <frame name = "left" SRC="hyperlinks.html">
    <frame name = "main" src = "https://images.unsplash.com/photo-1535957998253-26ae1ef29506?ixid=MnwXmJAJ3fDB8MHxzZWYy2h8MjN8fHdvcmtd8ZW58MHx8MHx8&ixlib=rb-1.2.1&w=1000&q=80" />
    <frame name = "right" src = "ad.html" />
  </frameset>

  <frameset>
    <frame name="footer" src="footer.html" scrolling="off">
  </frameset>

```

```
<noframes>
<body>Your browser does not support frames.

<!--<h1 class="my-5">Hi, <b><?php echo htmlspecialchars($_SESSION["username"]); ?></b>.
Welcome to our site.</h1>-->
<p>
  <a href="reset-password.php" class="btn btn-warning">Reset Your Password</a>
  <a href="logout.php" class="btn btn-danger ml-3">Sign Out of Your Account</a>
</p>
</body>
</noframes>
</frameset>
</html>
```

Header.html

```
<!DOCTYPE HTML>
<html>
  <body Bgcolor = "LightBlue">
    <marquee direction="left" scrollamount="15" >
      <Font Color= Blue Size = 5 >
        <b>WELCOME To WEBSITE about Gurpreet Kaur Jassal having FRAMES</b>
      </Font>
    </marquee>
  </body>
</html>
```

Footer.html

```
<!DOCTYPE html>
<html>
  <body bgcolor = "black">
    <font Color = "darkgrey">
      <p Align = "Center">
        For any query, get in touch with us throught:<br>
        Mail: gurpreetkaurjassal@icloud.com<br>
        Contact number: 7017660739
      </p>
    </font>
  </body>
</html>
```

Ad.html

```
<!DOCTYPE html>
<html>
  <head>
    <Font Size = 4>
    <u>Advertisments</u>
  </Font>
</head>


<p>But them by clicking here:
<a href =
"https://www.amazon.in/dp/B084Z4BMFF/ref=redir_mobile_desktop?_encoding=UTF8&aaxitk=8e34708c
```

```

<p>But them by clicking here:
<a href = "https://www.amazon.in/Bersache-Combo-Casual-Sneakers-
Multicolor/dp/B0744KSQ82/ref=lp_14072897031_1_1" target="_blank" rel="noopener noreferrer">Click
here</a>
    &nbsp;  </p>
</html>
```

OUTPUT:

