

|  |
| --- |
| **TITLE: :** **Develop and Demonstrate the use of Form Handling and Validation in PHP** |

**AIM:** To develop web forms using PHP form and Validation.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

The expected outcomes aim to enhance understanding of the implications and trade-offs associated with different methods of form data handling in PHP.

Books/ Journals/ Websites referred:

Steve Prettyman ,“Learn PHP 8 Using MySQL, JavaScript, CSS3, and HTML5”, Apress 2nd / 2020 edition.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Statement:** Design and implement an application to demonstrate HTML form integration with PHP for data collection and processing.

Utilize the registration page designed in Experiment No. 1 and create PHP scripts to handle form submission and data processing as follows:

1. Create separate PHP scripts for handling form submissions using different methods:

* **post\_registration.php**: Processes the registration form data using the $\_POST method.
* **get\_registration.php**: Handles the form data using the $\_GET method.
* **request\_registration.php**: Retrieves the submitted data using the $\_REQUEST method.

2. Each script should validate the input fields (e.g., check for valid email format and ensure mandatory fields are filled) and display the submitted registration details in a structured format.

**Implementation and screenshots of output**

**Signup using post**

<?php

require 'config.php';

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

    $name = $\_POST['name'] ?? '';

    $email = $\_POST['email'] ?? '';

    $password = $\_POST['password'] ?? '';

    if (empty($name) || empty($email) || empty($password)) {

        echo "<script>

                alert('All fields are required.');

                window.location.href = 'signup.php';

              </script>";

        exit;

    }

    if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

        echo "<script>

                alert('Invalid email format.');

                window.location.href = 'signup.php';

              </script>";

        exit;

    }

    if (strlen($password) < 6) {

        echo "<script>

                alert('Password must be at least 6 characters long.');

                window.location.href = 'signup.php';

              </script>";

        exit;

    }

    $password = password\_hash($password, PASSWORD\_DEFAULT);

    $stmt = $conn->prepare("SELECT email FROM USERS WHERE email = ?");

    $stmt->bind\_param("s", $email);

    $stmt->execute();

    $stmt->store\_result();

    if ($stmt->num\_rows > 0) {

        echo "<script>

                alert('User already exists! Please login.');

                window.location.href = 'login.php';

              </script>";

        exit;

    } else {

        $stmt = $conn->prepare("INSERT INTO USERS(name, email, password) VALUES(?,?,?)");

        $stmt->bind\_param("sss", $name, $email, $password);

        $stmt->execute();

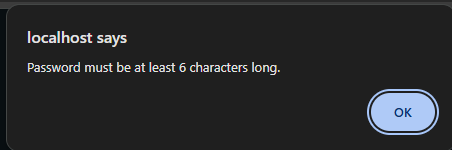
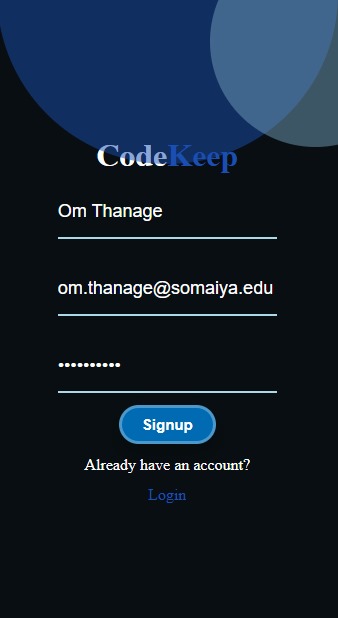
        header("Location: login.php");

        exit;

    }

}

?>

****

**Login.php**

<?php

require 'config.php';

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

    $email = $\_POST['email'] ?? '';

    $password = $\_POST['password'] ?? '';

    $stmt = $conn->prepare("SELECT name, email, password FROM USERS WHERE email=?");

    if (!$stmt) {

        die("Error preparing statement: " . $conn->error);

    }

    $stmt->bind\_param("s", $email);

    $stmt->execute();

    $stmt->store\_result();

    if ($stmt->num\_rows > 0) {

        $stmt->bind\_result($name, $email, $hashed\_pw);

        $stmt->fetch();

        if (password\_verify($password, $hashed\_pw)) {

            $\_SESSION['name'] = $name;

            $\_SESSION['email'] = $email;

            header("Location: dashboard.php");

            exit;

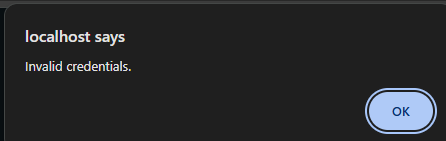
        }

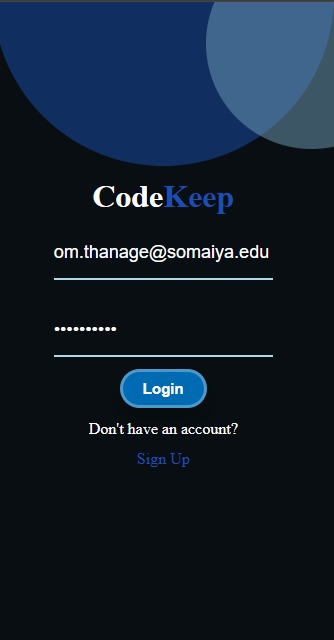
    }

    echo "<script>alert('Invalid credentials.');</script>";

}

?>

****



**All users**

<?php

require 'config.php';

$sql = "SELECT name, email FROM users";

$result = $conn->query($sql);

?>



**Conclusion:**

From this experiment we learnt how to do form validation with crud operations

**Post Lab Objective with Answer :**

1. Which method ($\_POST, $\_GET, $\_REQUEST) is the most secure, and why?

Ans. $\_POST is the most secure because data is not visible in the URL like $\_GET. This prevents exposure in browser history or server logs. It also allows sending large data compared to $\_GET. However, it’s still not fully secure unless used with HTTPS. Always validate inputs to prevent hacking risks.

1. From a developer's perspective, which method ($\_POST, $\_GET, $\_REQUEST) is easier to use, and why?

Ans. $\_GET is easier to use because data is visible in the URL, making it simple to debug and bookmark. It’s great for search queries and sharing links. $\_POST hides data, making debugging harder but better for secure data transfer. $\_REQUEST combines both but is not recommended due to security risks.

1. How does the $\_GET method handle data transmission, and what are its limitations?

Ans. $\_GET sends data in the URL as a query string (?name=John). It’s easy to use but has a character limit (~2000). Data is cached and stored in browser history, so it’s not safe for sensitive info. It’s best for search queries, filters, and navigation, but avoid it for passwords or personal data.