Date of submission: 07/02/2025
Batch: E-2 Roll No.: 16010123325
Div: E
Student Name: Shreyans Tatiya
Experiment No: 5
Staff In-charge:

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

Code-

HTML file-

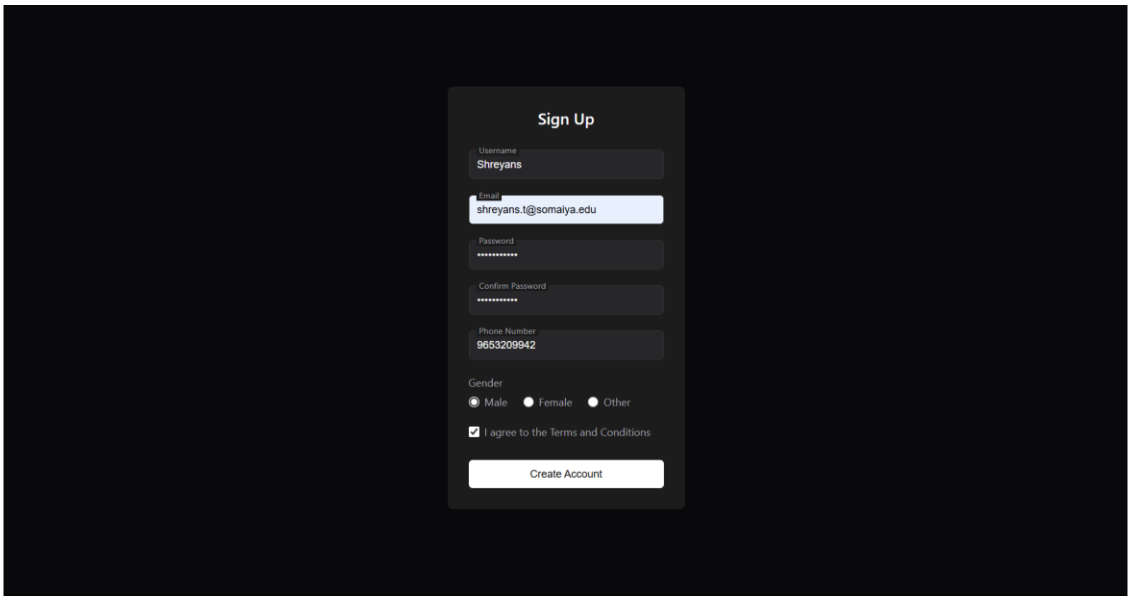
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Sign Up Form</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <form id="myForm" action="process.php" method="POST">
            <h2>Sign Up</h2>
            <div class="form-group">
                <input type="text" id="username" name="username"
placeholder=" " required>
                <label for="username">Username</label>
            </div>
            <div class="form-group">
                <input type="email" id="email" name="email"
placeholder=" " required>
                <label for="email">Email</label>
            </div>
            <div class="form-group">
                <input type="password" id="password" name="password"
placeholder=" " required>
                <label for="password">Password</label>
            </div>
            <div class="form-group">
                <input type="password" id="confirm-password"
name="confirm-password" placeholder=" " required>
                <label for="confirm-password">Confirm Password</label>
            </div>
            <div class="form-group">
                <input type="tel" id="phone" name="phone" placeholder="
" required>
                <label for="phone">Phone Number</label>
            </div>
            <div class="form-group">
                <label class="group-label">Gender</label>
                <div class="radio-group">
                    <label class="radio-label">
                        <input type="radio" name="gender" value="male"
required>
                        <span>Male</span>
                    </label>
                    <label class="radio-label">
                        <input type="radio" name="gender"
value="female">
                        <span>Female</span>
                    </label>
                    <label class="radio-label">
                        <input type="radio" name="gender" value="other">
                        <span>Other</span>
                    </label>
                </div>
            </div>
            <div class="form-group checkbox-group">
                <label class="checkbox-label">
                    <input type="checkbox" id="terms" name="terms"
required>
                    <span>I agree to the Terms and Conditions</span>
                </label>
            </div>
            <button type="submit">Create Account</button>
        </form>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

PHP File-

```
<?php
require_once 'db_connect.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST["username"];
    $email = $_POST["email"];
    $password = $_POST["password"];
    $confirm_password = $_POST["confirm-password"];
    $phone = $_POST["phone"];
    $gender = $_POST["gender"];
    $terms = isset($_POST["terms"]) ? true : false;

    $errors = [];

    // Username validation
    if (strlen($username) < 3) {
        $errors[] = "Username must be at least 3 characters long";
    }

    // Email validation
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $errors[] = "Please enter a valid email address";
    }

    // Password validation
    if (strlen($password) < 6) {
        $errors[] = "Password must be at least 6 characters long";
    }

    // Confirm password validation
    if ($password !== $confirm_password) {
        $errors[] = "Passwords do not match";
    }

    // Phone number validation
    if (!preg_match("/^\+?[\d\s-]{10,}$/", $phone)) {
        $errors[] = "Please enter a valid phone number";
    }

    // Gender validation
    if (!in_array($gender, ['male', 'female', 'other'])) {
        $errors[] = "Please select a valid gender";
    }

    // Terms validation
    if (!$terms) {
        $errors[] = "You must agree to the Terms and Conditions";
    }

    if (empty($errors)) {
        try {
            // Hash the password for security
            $hashed_password = password_hash($password,
PASSWORD_DEFAULT);

            // Prepare SQL statement
            $stmt = $pdo->prepare("INSERT INTO users (username, email,
password, phone, gender, terms) VALUES (?, ?, ?, ?, ?, ?)");

            // Execute with parameters
            $success = $stmt->execute([
                $username,
                $email,
                $hashed_password,
                $phone,
                $gender,
                $terms ? 1 : 0
            ]);

            if ($success) {
                echo json_encode([
                    "success" => true,
                    "message" => "Registration successful!"
                ]);
            } else {
                echo json_encode([
                    "success" => false,
                    "errors" => ["Database error: Failed to save user"]
                ]);
            }

        } catch(PDOException $e) {
            // Handle database errors (like duplicate email)
            if ($e->getCode() == 23000) { // Duplicate entry error
                echo json_encode([
                    "success" => false,
                    "errors" => ["This email address is already
registered"]
                ]);
            } else {
                echo json_encode([
                    "success" => false,
                    "errors" => ["Database error: " . $e->getMessage()]
                ]);
            }
        }
```

```
    } else {
        // Return validation errors
        echo json_encode(["success" => false, "errors" => $errors]);
    }
} else {
    // If not a POST request, redirect to the form page
    header("Location: index.html");
    exit();
}
?>
```

Output-



Sign Up

Username: Shreyans

Email: shreyans.t@somaiya.edu

Password: ••••••••••••

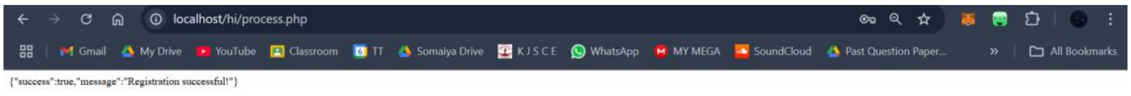
Confirm Password: ••••••••••••

Phone Number: 9653209942

Gender

[✓ Male] Female Other

☑ I agree to the Terms and Conditions



```
{"success":true,"message":"Registration successful!"}
```

Conclusion:

The above experiment gives an introduction to PHP and its utilization in form handling and validation allowing us to create registration and sign in pages for websites.

Post Lab Objective with Answer :

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

The $_POST method is the most secure among $_POST, $_GET, and $_REQUEST because it does not expose data in the URL. Unlike $_GET, which appends data to the URL and can be logged in browser history or server logs, $_POST sends data in the request body, making it less visible and harder to intercept.

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

From a developer's perspective, $_REQUEST is the easiest to use because it automatically retrieves data from both $_GET and $_POST methods without needing to specify which one was used. This simplifies handling form submissions when the request method may vary.

However, $_REQUEST is not recommended for security reasons, as it can make debugging harder and introduce unintended data overwrites. Developers usually prefer $_POST **for sensitive data** (e.g., passwords) and $_GET for retrieving data.

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

The $_GET method transmits data via the URL query string, appending key-value pairs to the URL (e.g., example.com/page.php?name=Shreya&age=20).

Disadvantages-

- The GET method should not be used while sending any sensitive information.
- A limited amount of data can be sent using method = "get". This limit should not exceed 2048 characters.
- For security reasons, never use the GET method to send highly sensitive information like username and password, because it shows them in the URL.
- The GET method cannot be used to send binary data (such as images or word documents) to the server.