**Batch:C1**

**Roll No.:16010122221**

**Experiment No. 05**

**TITLE:** Different Methods in PHP

**AIM:** The aim of this experiment is to evaluate and compare the effectiveness and security implications of using different methods in PHP.

# 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: 1. .

# Problem Statement: Description of the application implemented with output:

1. Develop a simple HTML form that collects personal information such as name, email, and age.
2. Create separate PHP scripts to handle form submission and data retrieval for each method: post\_form.php, get\_form.php, and request\_form.php
3. Each script will process the form data according to the respective method ($\_POST,

$\_GET, or $\_REQUEST) and output the submitted information.

Implementation and screen shots of output GET METHOD : SEE URL IN Screen shot

<?php

$nameError = $emailError = $passwordErr = "";

$name = $email = $password = "";

if ($\_SERVER["REQUEST\_METHOD"] == "GET" || $\_SERVER["REQUEST\_METHOD"] == "POST") {

    if (empty($\_REQUEST['name'])) {

        $nameError = "Name is Mandatory";

    } else {

        $name = $\_REQUEST["name"];

        if (!preg\_match("/^[a-zA-Z ]\*$/", $name)) {

            $nameError = "Only letters and white space allowed";

        }

    }

    if (empty($\_REQUEST['email'])) {

        $emailError = "Email is Mandatory";

    } else {

        $email = $\_REQUEST["email"];

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

            $emailError = "Invalid email format";

        } else {

            $domain = substr(strrchr($email, "@"), 1);

            if ($domain != "gmail.com" && $domain != "yahoo.com") {

                $emailError = "Only Gmail and Yahoo email addresses are allowed";

            }

        }

    }

    if (empty($\_REQUEST['password'])) {

        $passwordErr = "Password is Mandatory";

    } else {

        $password = $\_REQUEST["password"];

        if (!preg\_match("/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/", $password)) {

            $passwordErr = "Password must be at least 8 characters long and contain at least one uppercase letter, one lowercase letter, one number, and one special character";

        }

    }

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Bootstrap Form</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

  <div class="container mt-5">

    <div class="row justify-content-center">

      <div class="col-md-6">

        <div class="card">

          <div class="card-header">

            <h4>Sample Form</h4>

          </div>

          <div class="card-body">

            <form method="GET" autocomplete="off">

              <div class="mb-3">

                <label for="name" class="form-label">Name</label>

                <input type="text" class="form-control" id="name" name="name" placeholder="Enter your name">

                <span class="text-danger">*<*?php echo $nameError; ?></span>

              </div>

              <div class="mb-3">

                <label for="email" class="form-label">Email address</label>

                <input type="email" class="form-control" id="email" name="email" placeholder="Enter your email">

                <span class="text-danger">*<*?php echo $emailError; ?></span>

              </div>

              <div class="mb-3">

                <label for="password" class="form-label">Password</label>

                <input type="password" class="form-control" id="password" name="password" placeholder="Enter your password">

                <span class="text-danger">*<*?php echo $passwordErr; ?></span>

              </div>

              <button type="submit" class="btn btn-primary">Submit</button>

            </form>

          </div>

        </div>

      </div>

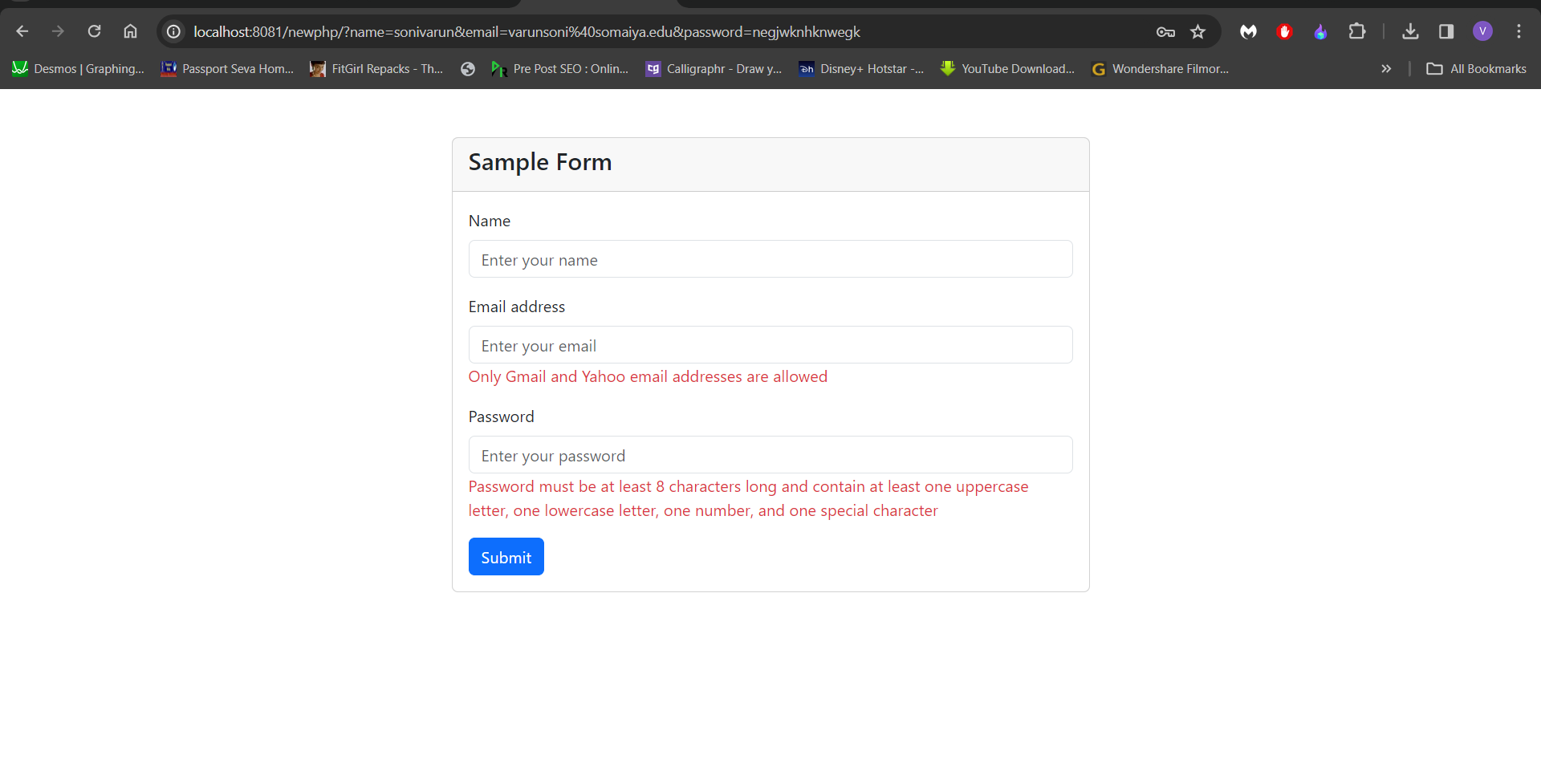
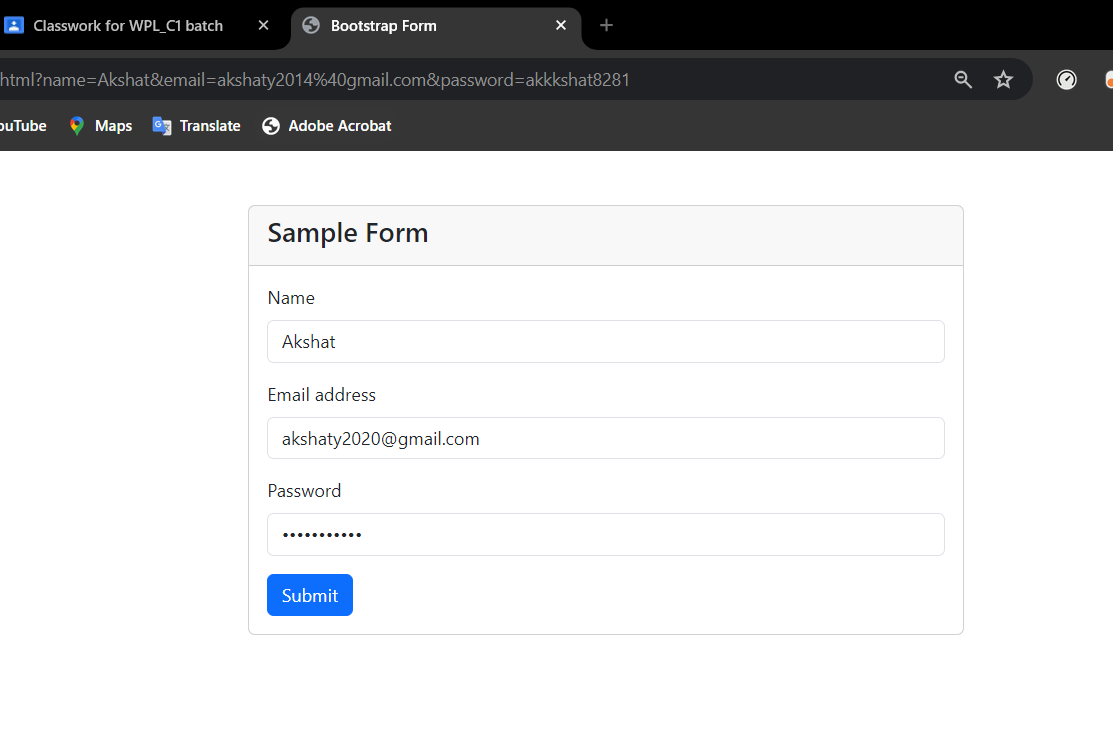
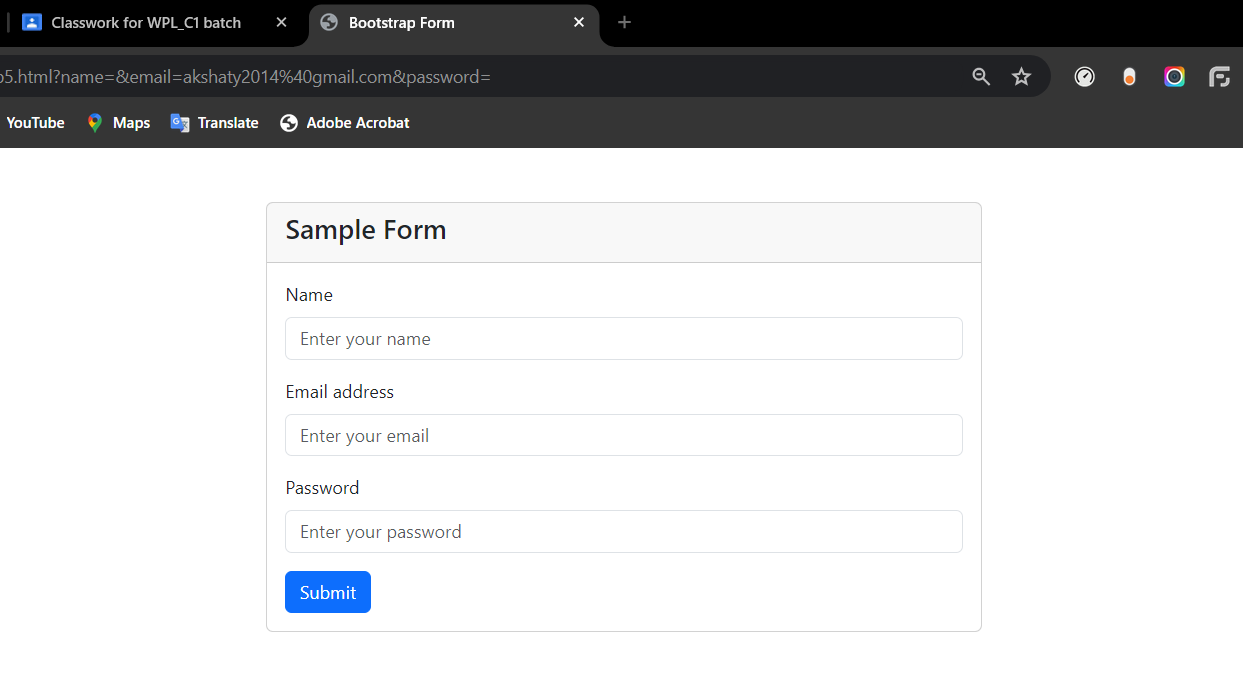
    </div>

  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>



POST METHOD: NOTHING IN URL

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Bootstrap Form</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

  <div class="container mt-5">

    <div class="row justify-content-center">

      <div class="col-md-6">

        <div class="card">

          <div class="card-header">

            <h4>Sample Form</h4>

          </div>

          <div class="card-body">

            <form method="GET" autocomplete="off">

              <div class="mb-3">

                <label for="name" class="form-label">Name</label>

                <input type="text" class="form-control" id="name" name="name" placeholder="Enter your name">

                <span class="text-danger">*<*?php echo $nameError; ?></span>

              </div>

              <div class="mb-3">

                <label for="email" class="form-label">Email address</label>

                <input type="email" class="form-control" id="email" name="email" placeholder="Enter your email">

                <span class="text-danger">*<*?php echo $emailError; ?></span>

              </div>

              <div class="mb-3">

                <label for="password" class="form-label">Password</label>

                <input type="password" class="form-control" id="password" name="password" placeholder="Enter your password">

                <span class="text-danger">*<*?php echo $passwordErr; ?></span>

              </div>

              <button type="submit" class="btn btn-primary">Submit</button>

            </form>

          </div>

        </div>

      </div>

    </div>

  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

<?php

$nameError = $emailError = $passwordErr = "";

$name = $email = $password = "";

if ($\_SERVER["REQUEST\_METHOD"] == "GET" || $\_SERVER["REQUEST\_METHOD"] == "POST") {

    if (empty($\_REQUEST['name'])) {

        $nameError = "Name is Mandatory";

    } else {

        $name = $\_REQUEST["name"];

        if (!preg\_match("/^[a-zA-Z ]\*$/", $name)) {

            $nameError = "Only letters and white space allowed";

        }

    }

    if (empty($\_REQUEST['email'])) {

        $emailError = "Email is Mandatory";

    } else {

        $email = $\_REQUEST["email"];

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

            $emailError = "Invalid email format";

        } else {

            $domain = substr(strrchr($email, "@"), 1);

            if ($domain != "gmail.com" && $domain != "yahoo.com") {

                $emailError = "Only Gmail and Yahoo email addresses are allowed";

            }

        }

    }

    if (empty($\_REQUEST['password'])) {

        $passwordErr = "Password is Mandatory";

    } else {

        $password = $\_REQUEST["password"];

        if (!preg\_match("/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/", $password)) {

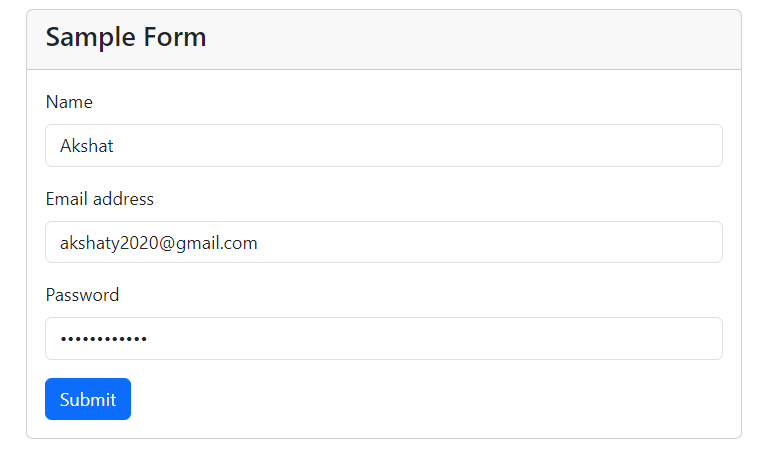
            $passwordErr = "Password must be at least 8 characters long and contain at least one uppercase letter, one lowercase letter, one number, and one special character";

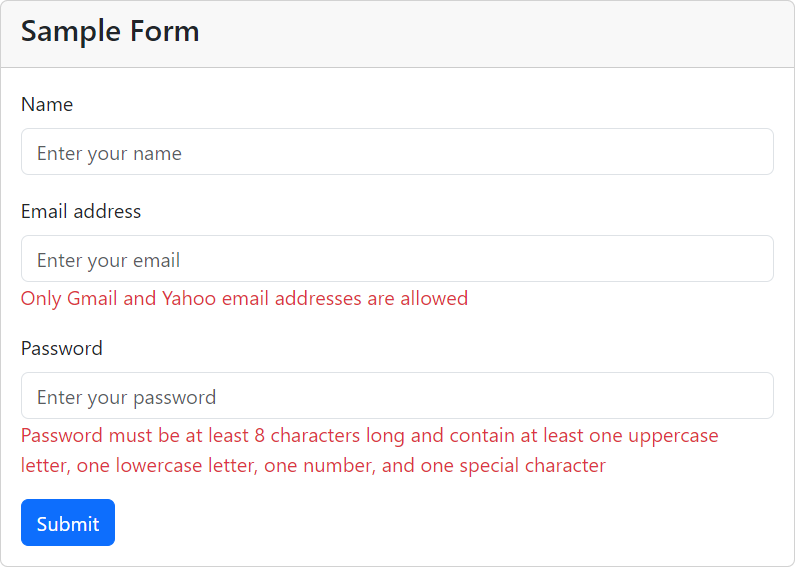
        }

    }

}

?>





**Post Lab Objective with Ans :**

1. Which method ($\_POST, $\_GET, $\_REQUEST) demonstrated the highest level of security, and why?

$\_POST Method Demonstrated the Highest Level of Security

The $\_POST method demonstrated the highest level of security because the data is sent through the HTTP request body, which is not visible in the URL. This makes it more difficult for attackers to intercept or tamper with the data being transmitted.

In contrast, the $\_GET method sends data through the URL as query parameters, making it visible in the browser's address bar and in server logs. This makes the data more susceptible to interception, manipulation, and various types of attacks such as cross-site scripting (XSS) and cross-site request forgery (CSRF).

The $\_REQUEST superglobal array is a combination of $\_GET, $\_POST, and

$\_COOKIE, which means it inherits the security vulnerabilities associated with each of these methods.

1. From a developer's perspective, which method ($\_POST, $\_GET, $\_REQUEST) is the most convenient to implement and maintain?

$\_GET Method is the Most Convenient for Developers

From a developer's perspective, the $\_GET method is generally the most convenient to implement and maintain. This is because the data is sent through the URL, making it easier to debug, bookmark, and share links with specific parameters.

The $\_POST method requires additional form handling and parsing on the server-side, which can be more complex to set up and maintain, especially in larger applications.

The $\_REQUEST superglobal array, while convenient in some cases, can introduce security risks if not handled properly. It's generally recommended to use either $\_GET or $\_POST explicitly, depending on the specific use case and security requirements.

However, it's important to note that convenience should never take precedence over security. In scenarios where sensitive data is being transmitted (e.g., login credentials, financial information), the $\_POST method should always be preferred due to its superior security characteristics.