Batch:- B-2 **Roll No:-** 16010122151

Experiment / assignment / tutorial No. 5

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 andtrade-offs associated with different methods of form data handling in PHP.

Books/ Journals/ Websites referred:

1. .

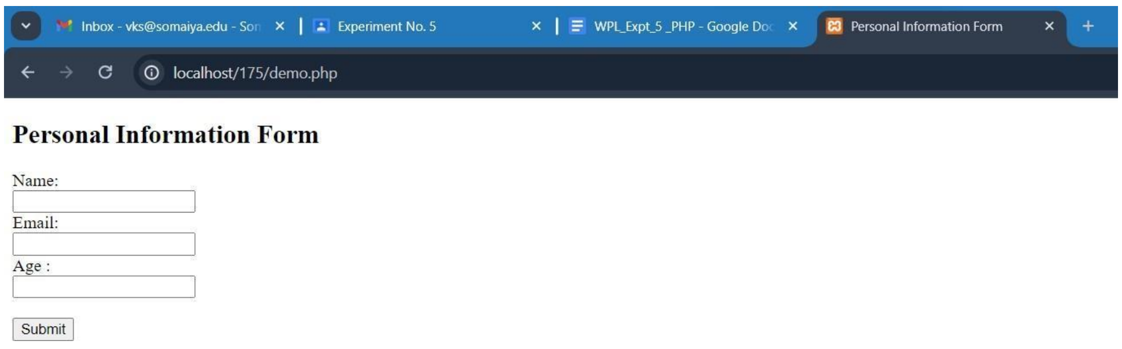
Problem Statement: Description of the application implemented with output:

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:

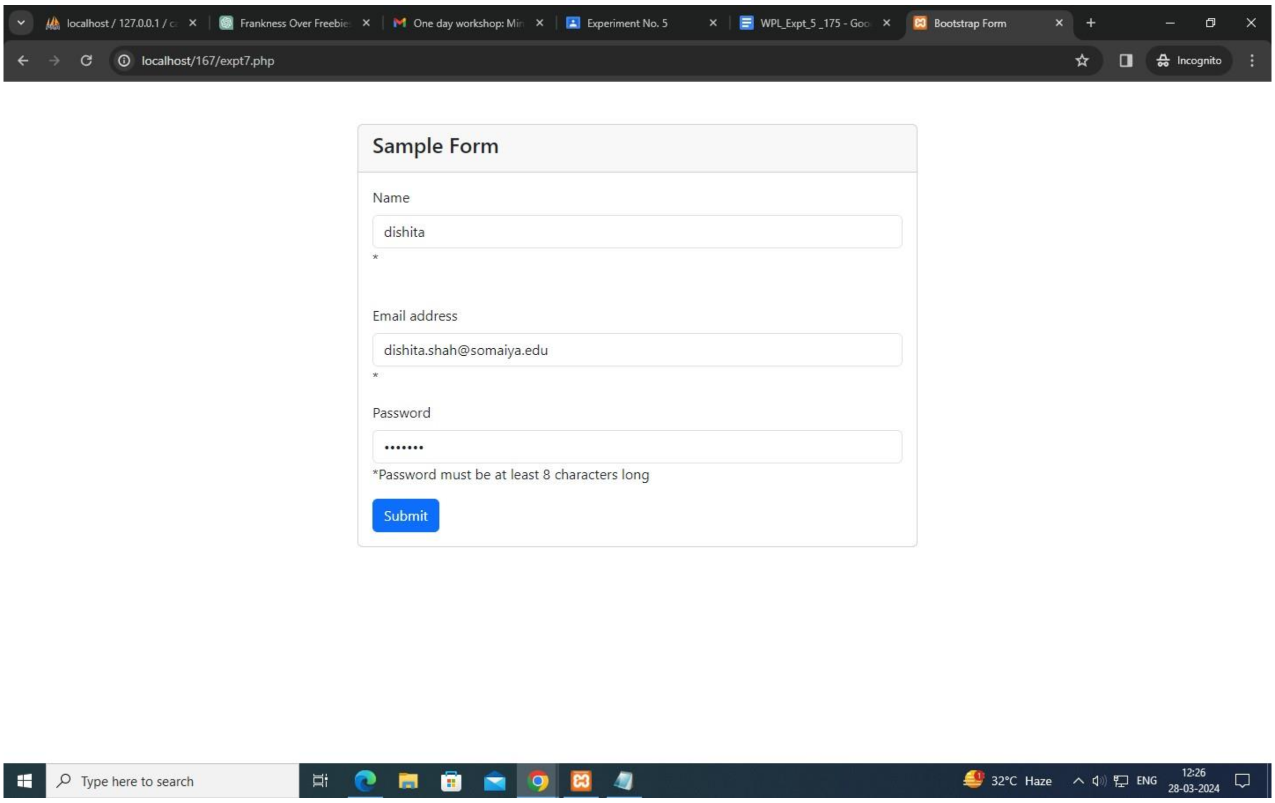
1.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Personal Information Form</title>
</head>
<body>
    <h2>Personal Information Form</h2>
    <form action="submit.php" method="post">
        <label for="name">Name:</label><br>
        <input type="text" id="name" name="name" required><br>

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

        <label for="age">Age :</label><br>
        <input type="number" id="age" name="age" required><br><br>

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


Inbox - vks@somaiya.edu - Son
Experiment No. 5
WPL_Expt_5_PHP - Google Doc
Personal Information Form
localhost/175/demo.php
Personal Information Form
Name:
Email:
Age :
Submit

2. POST method:-

```
<?php
$nameError= $emailError=$passwordErr="";
$name=$email=$pass="";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
      $nameError = "Name is mandatory";
    } else {
      $name = $_POST["name"];
      // check if name only contains letters and whitespace
      if (!preg_match("/^[a-zA-Z-']*$/", $name)) {
        $nameError = "Only letters allowed";
      }
    }

    if (empty($_POST["email"])) {
        $emailError = "Email is mandatory";
      } else {
        $email = $_POST["email"];
        // check if e-mail address is well-formed
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
          $emailError = "Invalid email format";
        }
      }
      if (empty($_POST["password"])) {
        $passwordErr = "Password is required";
    } else {
        $pass = $_POST["password"];
        if (strlen($pass) < 8) {
            $passwordErr = "Password must be at least 8 characters long";
                                                                  }              elseif
(!preg_match("/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*()-_+=])[A-Za-z\d!@
#$%^&*()-_+=]{8,}$/", $pass)) {
            $passwordErr = "Password must contain at least one uppercase letter";
    }
    }
    }
?>
<?php

?>
<!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>

    <form action="" method="post">
        <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>
                            <div class="mb-3">
                                <label for="name" class="form-label">Name</label>
                                                <input type="text" class="form-control" name="name"
placeholder="Enter your name" value="<?php echo $name;?>">
                                <span class="error">*<?php echo $nameError;?></span>
                                <br><br>
                            </div>
                            <div class="mb-3">
                                <label for="email" class="form-label">Email address</label>
                                            <input type="email" class="form-control" name="email"
placeholder="Enter your email" value="<?php echo $email;?>">
                                <span class="error">*<?php echo $emailError;?></span>
                            </div>
                            <div class="mb-3">
                                <label for="password" class="form-label">Password</label>
                                <input type="password" class="form-control" name="password"
placeholder="Enter your password" value="<?php echo $pass;?>">
                                <span class="error">*<?php echo $passwordErr;?></span>
                            </div>
                            <button type="submit" class="btn btn-primary">Submit</button>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</form>

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

</html>
```



localhost/167/expt7.php

Sample Form

Name

dishita

*

Email address

dishita.shah@somaiya.edu

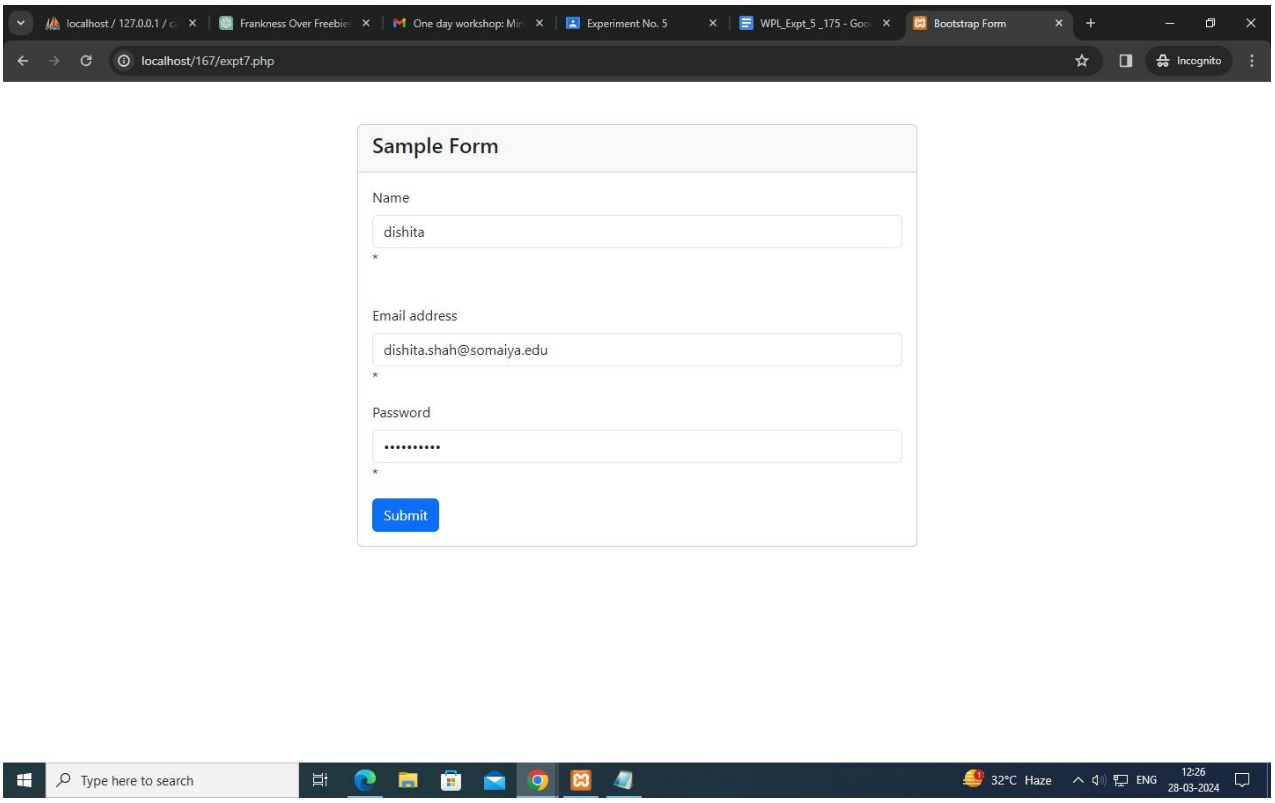
*

Password

•••••••

*Password must be at least 8 characters long

Submit



localhost/167/expt7.php

Sample Form

Name

dishita

*

Email address

dishita.shah@somaiya.edu

*

Password

••••••••••

*

Submit

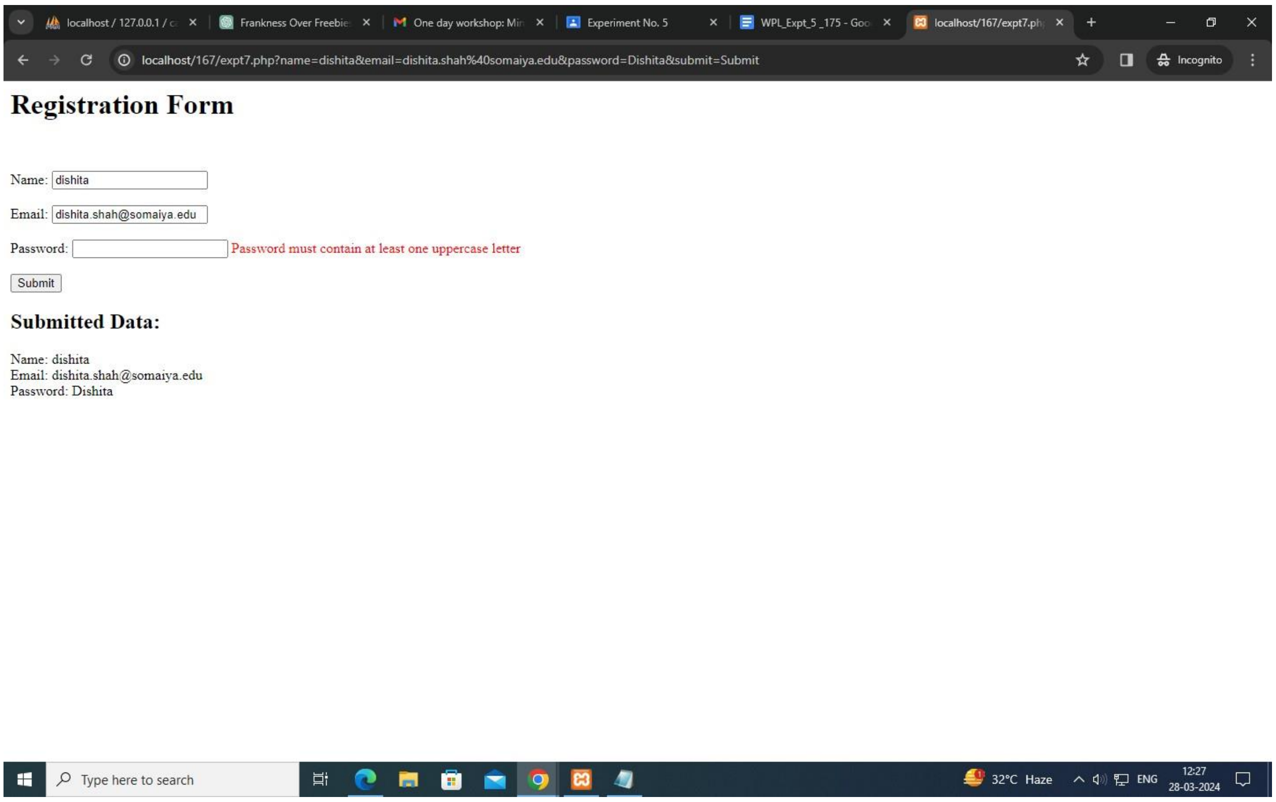
3. Get method:-

```
<?php
        // Define variables and set to empty values
        $nameErr = $emailErr = $passwordErr = "";
        $name = $email = $password = "";

        if ($_SERVER["REQUEST_METHOD"] == "GET") {
        // Name validation
if (empty($_GET["name"])) {
        $nameErr = "Name is required";
        } else {
        $name = $_GET["name"];
        // Check if name only contains letters and whitespace
        if (!preg_match("/^[a-zA-Z ]*$/", $name)) {
                $nameErr = "Only letters and white space allowed";
        }
        }

        // Email validation
        if (empty($_GET["email"])) {
        $emailErr = "Email is required";
        } else {
        $email = $_GET["email"];
        // Check if email is valid
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
                $emailErr = "Invalid email format";
```

```
            }

            }



            // Password validation

            if (empty($_GET["password"])) {

                $passwordErr = "Password is required";

            } else {

            $password = $_GET["password"];
if
    (!preg_match("/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*()-_+=])[A-Za-z\d!@#$%^&*()-_+=]{8,}$/", $password))

            {

                $passwordErr = "Password must contain at least one uppercase letter";

            }



            }}

            ?>

    <html>

    <head>

    <style>

    .error {color: #FF0001;}

    </style>

    </head>

    <body>

    <h1>Registration Form</h1>

    <br><br>

            <form method="get" action="<?php echo
    htmlspecialchars($_SERVER["PHP_SELF"]); ?>">

            Name: <input type="text" name="name" value="<?php echo $name; ?>">
```

```
<span class="error"><?php echo $nameErr; ?></span>

<br><br>



Email: <input type="text" name="email" value="<?php echo $email; ?>">

<span class="error"><?php echo $emailErr; ?></span>

<br><br>

Password: <input type="password" name="password">

<span class="error"><?php echo $passwordErr; ?></span>

<br><br>



<input type="submit" name="submit" value="Submit">

</form>



<?php

// Display submitted data

if (!empty($name) && !empty($email) && !empty($password)) {

echo "<h2>Submitted Data:</h2>";

echo "Name: $name <br>";

echo "Email: $email <br>";






// Note: You should hash and store the password securely in a real-world scenario

echo "Password: $password <br>";

}

?>

</body>
```

```
</html>
```


localhost/167/expt7.php?name=dishita&email=dishita.shah%40somaiya.edu&password=Dishita&submit=Submit

Registration Form

Name: dishita

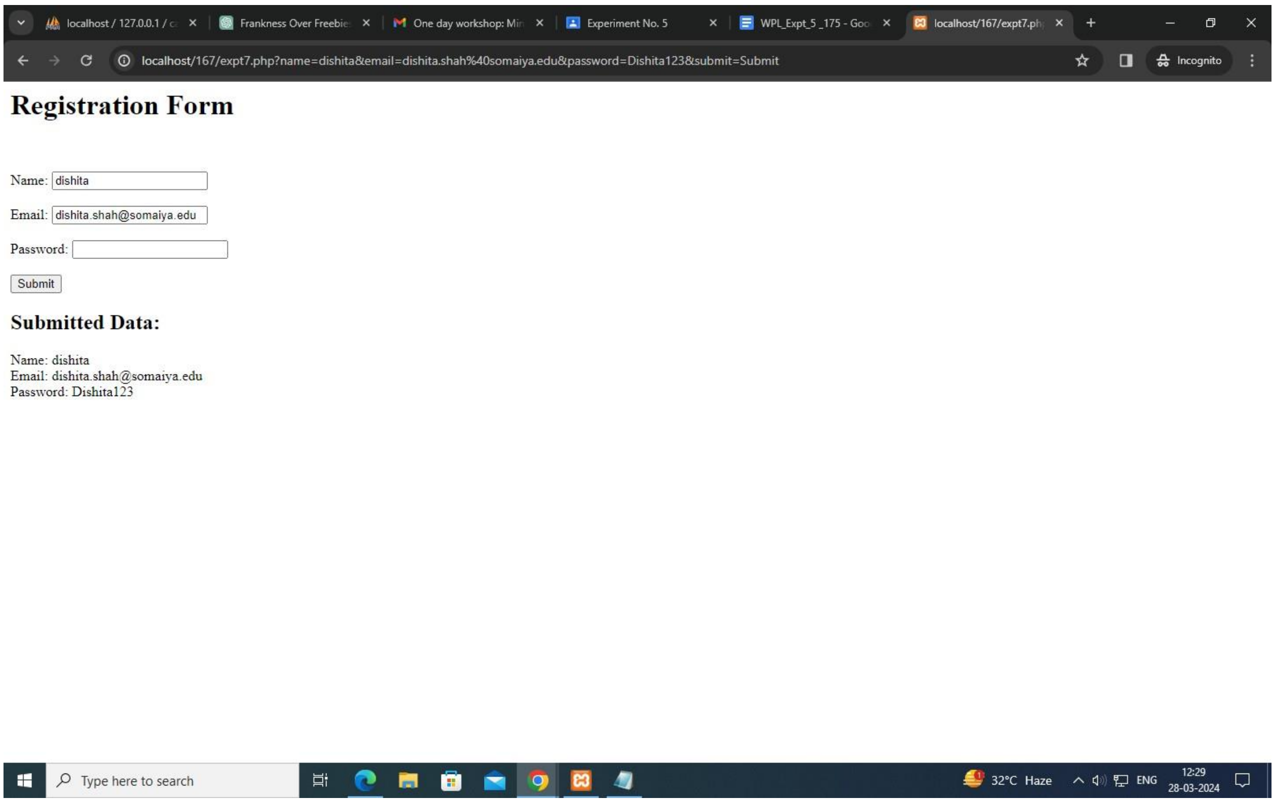
Email: dishita.shah@somaiya.edu

Password: Password must contain at least one uppercase letter

Submit

Submitted Data:

Name: dishita
Email: dishita.shah@somaiya.edu
Password: Dishita


localhost/167/expt7.php?name=dishita&email=dishita.shah%40somaiya.edu&password=Dishita123&submit=Submit

Registration Form

Name: dishita

Email: dishita.shah@somaiya.edu

Password:

Submit

Submitted Data:

Name: dishita
Email: dishita.shah@somaiya.edu
Password: Dishita123

Post Lab Objective with Ans :

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

Ans-

- Visibility: Data sent via $_POST is not visible in the URL, unlike $_GET. This makes it less susceptible to shoulder surfing or being logged in browser history.

- Data Size: $_POST can handle larger amounts of data compared to $_GET. $_GET data is limited by the maximum length of a URL, which can vary across different servers and browsers. Sending large amounts of data via $_GET can result in truncation or errors.
- Security Implications: Data sent via $_GET is visible in the URL, which can lead to security vulnerabilities such as sensitive data exposure through logs, browser history, or referrer headers. On the other hand, $_POST data is not visible in the URL, reducing the risk of exposure.
- Cacheability: $_POST requests are not cached by default, which means sensitive data sent via $_POST is less likely to be stored in caches or proxies compared to $_GET requests, which can be cached by browsers and proxies.
- Request Method: While $_REQUEST can include data from both $_POST and $_GET, using $_POST directly is more explicit and makes it clear that the data is being submitted securely via the POST method.

Overall, $_POST provides better security for transmitting sensitive data over HTTP compared to $_GET and $_REQUEST. However, it's important to note that proper security measures should be implemented regardless of the method used, such as input validation, data sanitization, and protection against common web vulnerabilities like SQL injection and Cross-Site Scripting (XSS)

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

Ans-

From a developer's perspective, $_POST is often considered the most convenient to implement and maintain for several reasons:

Data Handling: $_POST is specifically designed for handling data submitted via HTML forms using the POST method. This makes it straightforward for developers to access form data directly via $_POST without having to parse the URL or handle query strings manually.

Security: As mentioned earlier, $_POST is more secure than $_GET because it doesn't expose data in the URL. This reduces the risk of sensitive information being leaked through browser history, server logs, or other means.

Consistency: By using $_POST, developers can ensure a consistent approach to handling form submissions across their applications. This can simplify code maintenance and reduce the likelihood of errors or inconsistencies in data handling.

Encapsulation: $_POST encapsulates data within the HTTP request body, which can make the code cleaner and more organized compared to extracting data from the URL with $_GET.

Compatibility: While $_REQUEST can combine data from both $_POST and $_GET, it may lead to ambiguity and unexpected behavior. Using $_POST exclusively ensures that developers are explicitly handling data submitted via the POST method.

Overall, $_POST is often the preferred choice for developers when implementing form submissions in web applications due to its simplicity, security benefits, and compatibility with HTML forms. However, it's essential to consider the specific requirements and constraints of each project when deciding which method to use.