

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)

Batch:C2 Roll No.: 16010122257

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. <https://tutorialsclass.com/exercises/php/php-all-exercises-assignments/>

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.

CODE:

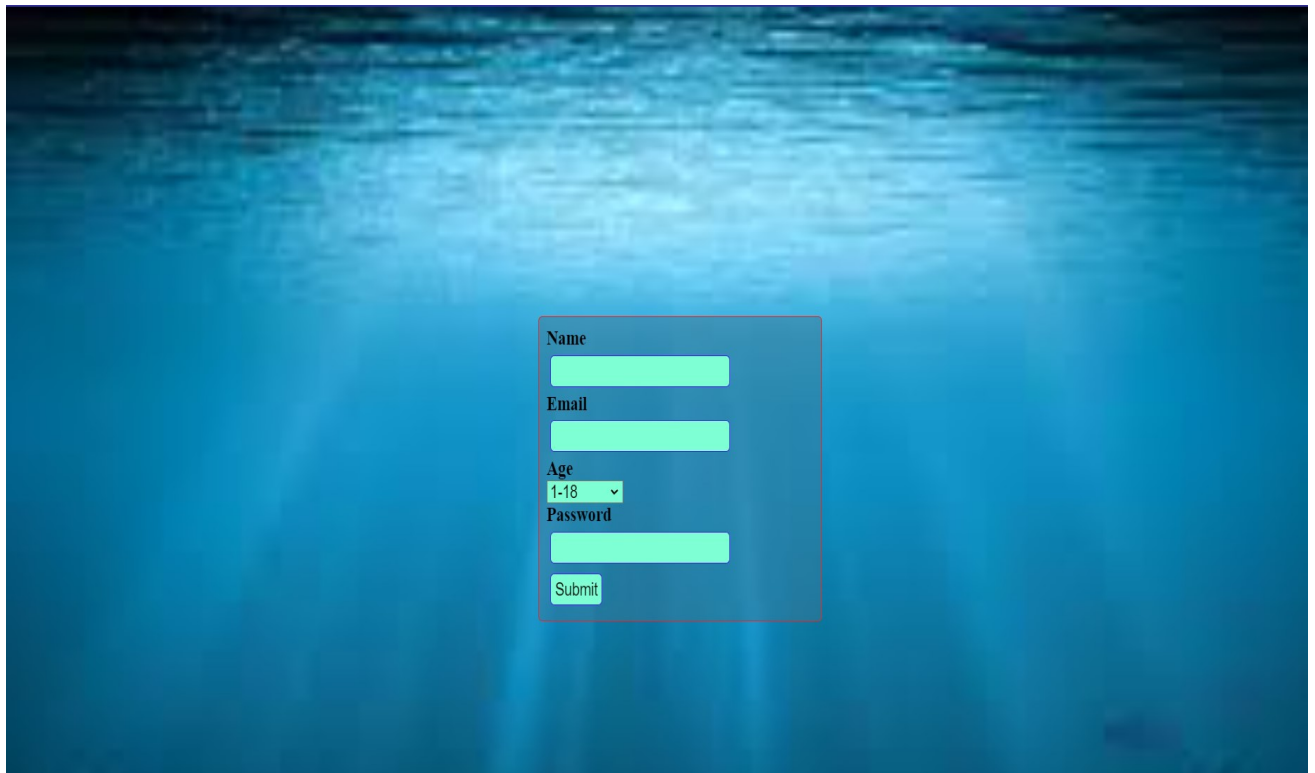
```
<!DOCTYPE html>
<html>
<body>
  <style>
    body{
      background-image:url("images.jpg");
      background-repeat: no-repeat;
      background-size: cover;
    }
    form{
      margin:auto;
      margin-top:310px;
      width: 20%;
      padding: 10px;
      border: 1px solid #e22b2b;
      border-radius: 5px;
      background-color: rgba(58, 119, 150, 0.5);
      backdrop-filter: blur(2px);
      backdrop-filter: brightness(60%);
      backdrop-filter: contrast(40%);
      backdrop-filter: drop-shadow(4px 4px 10px rgb(129, 129, 215));
    }
    form input{
```

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)

```
margin: 5px;
padding: 5px;
border-radius: 5px;
border: 1px solid #432be2;
background-color:aquamarine;
}
</style>
<form action="work.php" method="POST">
  <label for="n"><strong>Name</strong></label><br>
  <input type="text" name="n"><br>
  <label for="e"><strong>Email</strong></label><br>
  <input type="text" name="e"><br>
  <label for="age"><strong>Age</strong></label><br>
  <select name="age" style="background-color:aquamarine ;">
    <option value="1-18">1-18</option>
    <option value="19-45">19-45</option>
    <option value="45-60">45-60</option>
    <option value="above 60">above 60</option>
  </select>
  <br>
  <label for="p"><strong>Password</strong></label><br>
  <input type="password" name="p"><br>
  <input type="submit">
</form>
```

```
</body>
</html>
```

OUTPUT:



Name

Email

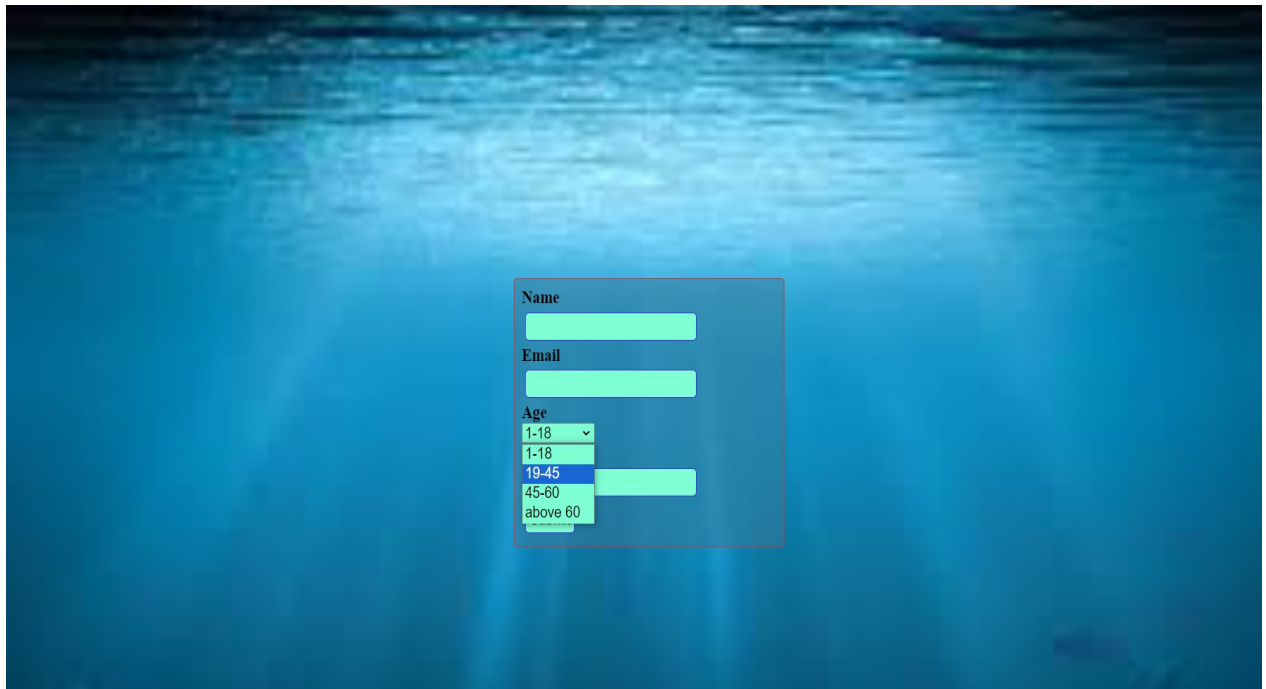
Age

1-18

Password

Submit

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)



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

CODE:

```
<?php
    $n = $_POST['n'];
    $e = $_POST['e'];
    $Age = $_POST['age'];
    $p = $_POST['p'];

    $conn = new mysqli('localhost','root','','newtest');
    if($conn->connect_error){
        die('Connection Failed : '.$conn->connect_error);
    }else{
        $stmt = $conn->prepare("insert into registration(Name, Email, Age, Password)
values(?, ?, ?, ?)");
        $stmt->bind_param("ssis", $Name, $Email, $Age, $Password);
        $stmt->execute();
        echo "Registration Successfully...";
        $stmt->close();
        $conn->close();
    }
    echo "PHP up and running!";
    if( strpos($e,'.')==null){
        echo "!! email invalid";
    }
    $pattern = "[0-9].";

    if( preg_match($pattern,$p)==0){
        echo "!! password must contain numbers";
    }
    if (strlen($p) < "7") {
        echo "!! Password too short";
    }
}
```

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)

}

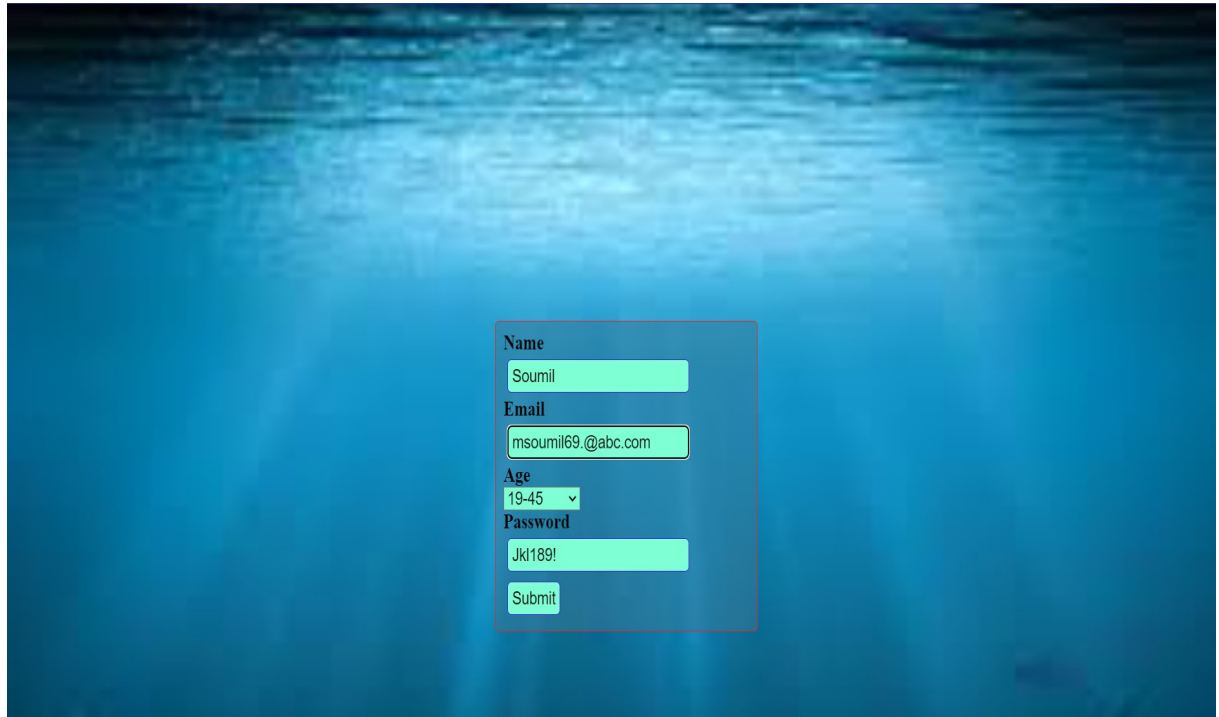
}

OUTPUT:

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:

Case-1:



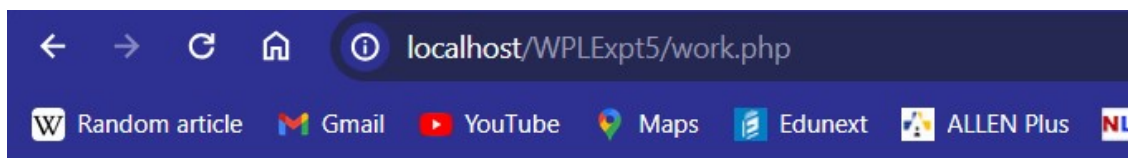
Registration

Name

Email

Age
19-45 ▾

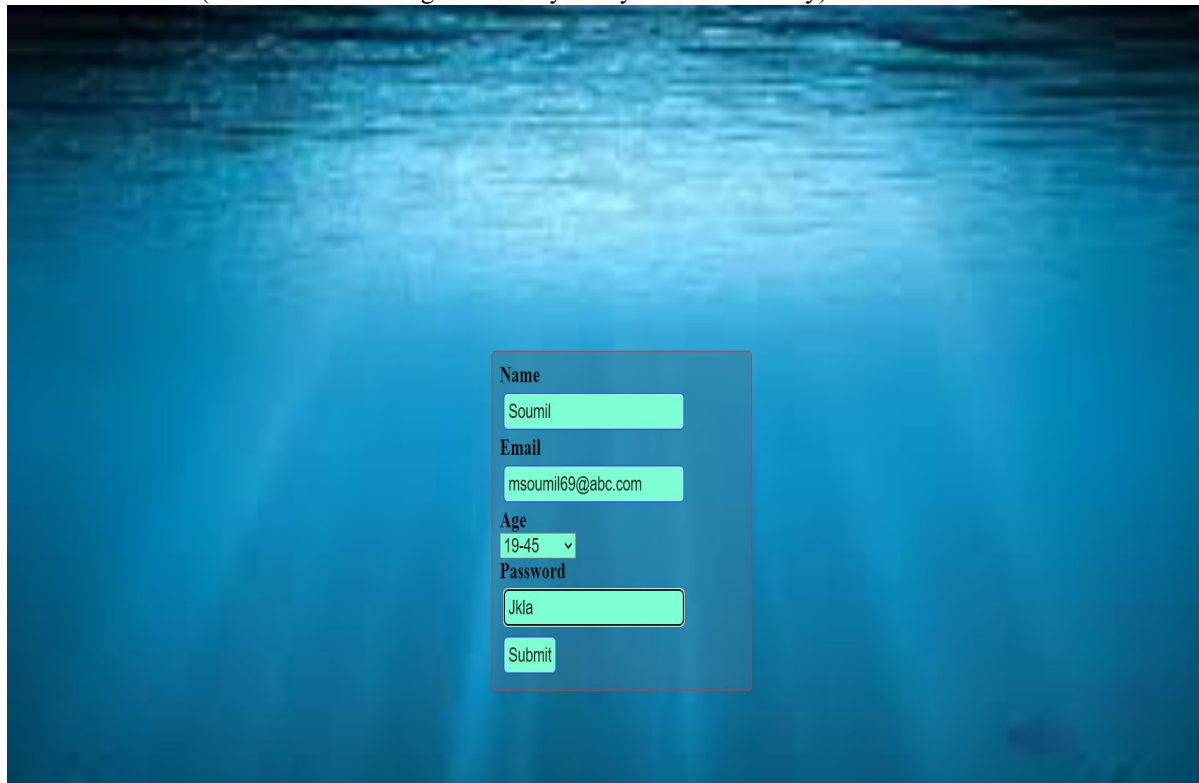
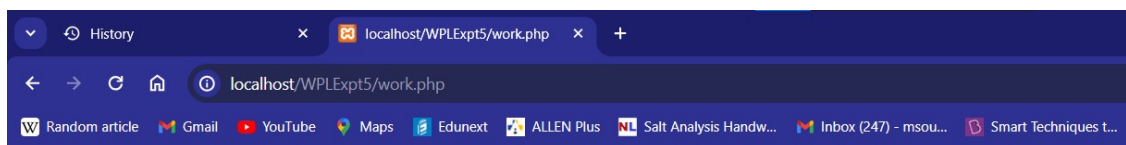
Password



Registration Successfully...PHP up and running!

Case-2:

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)

A registration form is displayed on a blue background. The form fields are: Name (Soumil), Email (msoumil69@abc.com), Age (19-45 dropdown), Password (Jkla), and a Submit button.

Registration Successfully...PHP up and running!!! email invalid!! password must contain numbers!! Password too short

:

Post Lab Objective with Ans :

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

In PHP, `\$_POST` and `\$_GET` are both used to collect form data after submitting an HTML form with method="post" or method="get" respectively. The `\$_REQUEST` variable is a combination of `\$_GET`, `\$_POST`, and `\$_COOKIE` variables.

K. J. Somaiya College of Engineering, Mumbai-77

(A Constituent College of Somaiya Vidyavihar University)

In terms of security, `\$_POST` is generally considered more secure than `\$_GET`. This is because data sent via the POST method is not visible in the URL, unlike with the GET method where data is appended to the URL. This makes it less likely for sensitive information to be exposed in logs, browser history, or through shoulder surfing.

Moreover, using `\$_POST` is recommended for sensitive data like passwords or personal information because it's not as easily tampered with by the user. Users can manipulate URL parameters with `\$_GET`, potentially leading to security vulnerabilities such as injection attacks (e.g., SQL injection) or cross-site scripting (XSS) attacks.

Therefore, for the highest level of security, it's generally best to use `\$_POST` when handling sensitive data or performing actions that modify server state, while still taking appropriate security measures such as input validation, sanitization, and prepared statements to prevent common security vulnerabilities.

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

From a developer's perspective, the convenience of implementing and maintaining a method depends on various factors such as the specific requirements of the project, the nature of the data being handled, and the overall architecture of the application. However, I can provide some general considerations:

1. \$_POST:

- Convenience: Using `\$_POST` is straightforward for submitting form data and handling it on the server side. It's commonly used for submitting sensitive information like passwords or large amounts of data.

- Maintenance: Since POST data is not appended to the URL, it can make URLs cleaner and more readable, which can aid in maintaining the application. Additionally, because POST data is typically submitted through forms, developers may find it easier to organize and manage form submissions.

2. \$_GET:

- Convenience: `\$_GET` is convenient for passing data through URLs, such as search queries or pagination parameters. It's particularly useful for bookmarkable or shareable URLs.

- Maintenance: While `\$_GET` can be convenient for simple data passing, it may lead to longer and less readable URLs, especially when passing large amounts of data. This could potentially affect the maintainability of the application, especially if URLs need to be managed carefully for SEO or other purposes.

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)

3. `$_REQUEST`:

- Convenience: `$_REQUEST` combines data from `$_GET`, `$_POST`, and `$_COOKIE`, providing a unified interface for accessing request data. It can be convenient if the developer needs to handle data from multiple sources without explicitly checking each one.

- Maintenance: While `$_REQUEST` may seem convenient, it can make code less transparent and harder to debug, especially if the source of the data isn't clear. Relying heavily on `$_REQUEST` could lead to potential security issues if not used carefully, as it may inadvertently expose sensitive data stored in cookies.

Overall, the choice of which method to use depends on the specific requirements and considerations of the project. In many cases, developers may prefer to use `$_POST` for handling form submissions and sensitive data, while `$_GET` can be convenient for passing non-sensitive data via URLs. `$_REQUEST` may be suitable for certain situations where data needs to be accessed from multiple sources, but its use should be carefully considered to maintain code clarity and security.