Experiment No. 4

Title: Cookies and Session handling using PHP

Batch: A3 **Roll No.: 16010421073** **Experiment No.:4**

Aim: Write PHP programs to demonstrate working of cookies and session handling with the help of basic programming constructs.

Resources needed: Windows OS, Web Browser, Editor, XAMPP Server

---

Pre Lab/ Prior Concepts:
Students should have prior knowledge of HTML/CSS/Basic Programming.

Theory:

Cookie:
A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it sends the cookie too with the request.

In PHP, to create/modify/delete a cookie, the setcookie() function is used. The syntax of setcookie() function is as follows:

setcookie (name, value, expire, path, domain, secure, httponly)
Where
name - The name of the cookie.
value - The value of the cookie. This value is stored on the clients computer; do not store sensitive information.
expire - The time the cookie expires. This is a Unix timestamp so is in number of seconds since the epoch. In other words, you'll most likely set this with the time() function plus the number of seconds before you want it to expire.
path - The path on the server in which the cookie will be available on. If set to '/', the cookie will be available within the entire domain. If set to '/foo/', the cookie will only be available within the /foo/ directory and all sub-directories such as /foo/bar/ of domain. The default value is the current directory that the cookie is being set in.
domain - The (sub)domain that the cookie is available to. Setting this to a subdomain (such as 'www.example.com') will make the cookie available to that subdomain and all other sub-domains of it (i.e. w2.www.example.com). To make the cookie available to the whole domain (including all subdomains of it), simply set the value to the domain name ('example.com', in this case). **secure-** Indicates that the cookie should only be transmitted over a secure HTTPS connection from the client. When set to TRUE, the cookie will only be set if a secure connection exists. **httponly** - When TRUE the cookie will be made accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript.

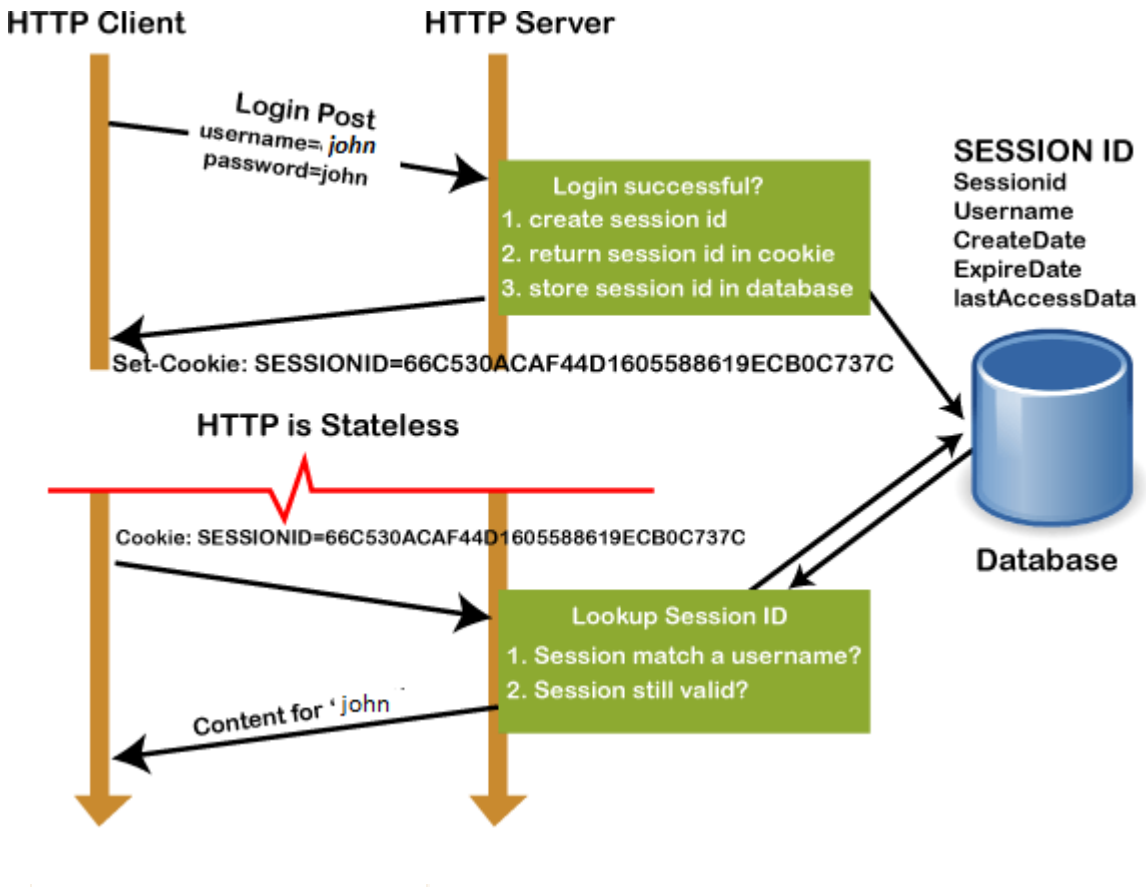
To retrieve a cookie value $_COOKIE superglobal is used.
For example: Assuming the cookie name is 'favcolor', this value is retrieved using $_COOKIE['favcolor']

Session Handling:

HTTP is a stateless protocol. So web server treates every request as a new request. So session variables solve this problem by storing user information to be used across multiple pages. By default, session variables last until the client closes the browser.

- Unlike a cookie, the information is not stored on the client's machine.

The working of a session can be understood with the help of the below diagram:



To create a session / resume a session ,session_start() function is used. session_start() creates a session or resumes the current one based on a session identifier passed via a GET or POST request, or passed via a cookie.

To set/modify/retrieve a session variable value, $_SESSION superglobal is used.

For example: Assuming the session variable name is 'favcolor',

- It's value is set as $_SESSION['favcolor'] ='green'
- It's value is modified as $_SESSION['favcolor'] = 'blue'
- It's value is retrieved as $_SESSION['favcolor']

Procedure:

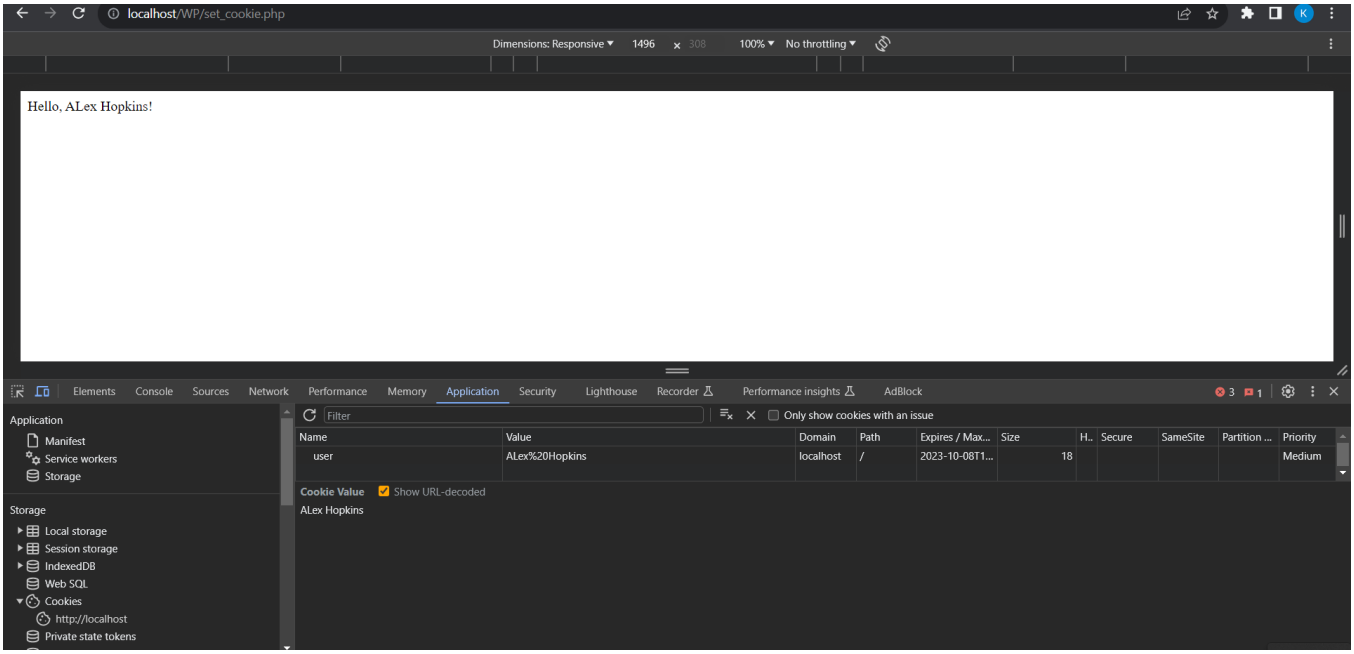
1. Read reference[1] to know more about cookies and sessions.
2. Write a simple program to set a cookie, and then read back the value.
3. Write a code to delete the cookie.
4. Start a session and check for timeout.
5. Delete the session.
6. Paste the code and output snapshots in the write-ups and submit.

Output(Code with result Snapshot)

1) reading a cookie and its value

Code(setting a cookie with name user in set_cookie.php)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
<?php
// Set a cookie with a name 'user' and value 'Alex Hopkins'
setcookie('user', 'ALex Hopkins', time() + 3600, '/');

// Read the value of the 'user' cookie
if (isset($_COOKIE['user']))
{
    $username = $_COOKIE['user'];
    echo "Hello, $username!";
}
else
{
    echo "Cookie not set.";
}
?>


</body>
</html>
```

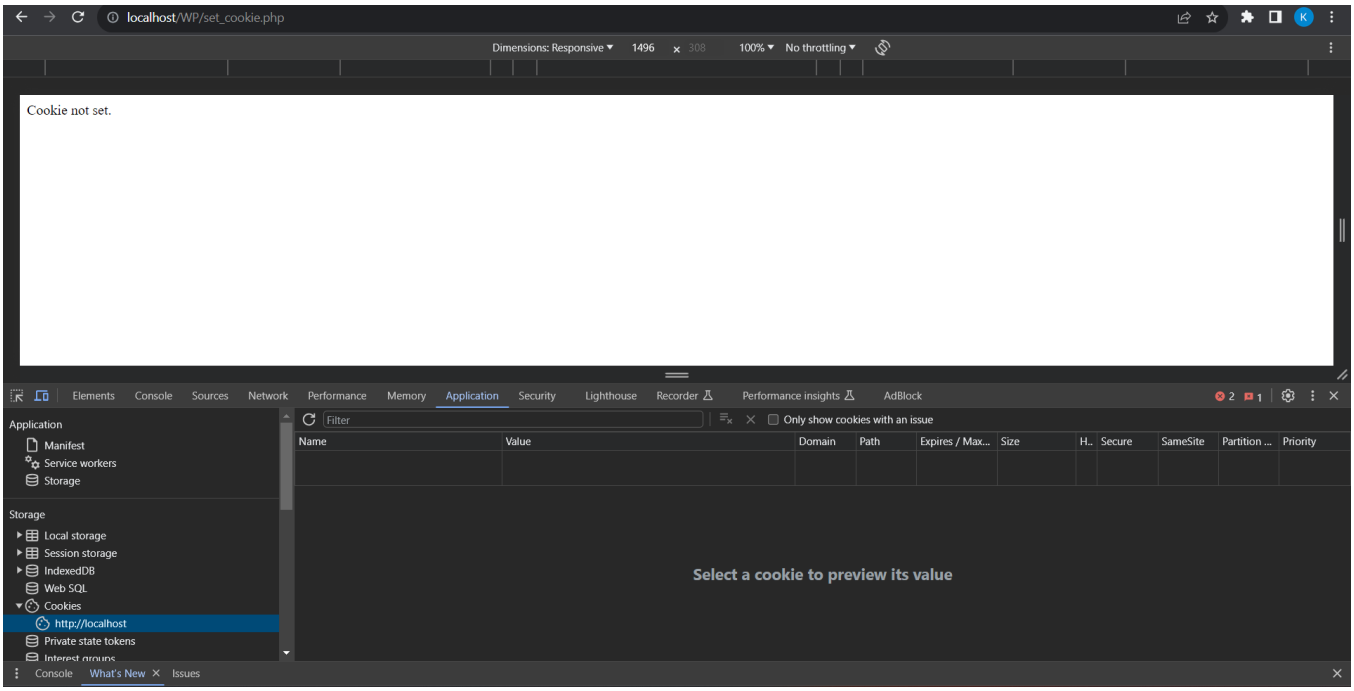


2) Write a code to delete the cookie.

Code:(Deleting the user cookie in set_cookie.php)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
<?php
// Set a cookie with a name 'user' and value 'Alex Hopkins'
setcookie('user', 'ALex Hopkins', time() + 3600, '/');

// Read the value of the 'user' cookie
if (isset($_COOKIE['user']))
{
    $username = $_COOKIE['user'];
    echo "Hello, $username!";
}
else
{
    echo "Cookie not set.";
}
?>
```

```php
<?php
// Delete the 'user' cookie
setcookie('user', '', time() - 3600, '/');
?>


</body>
</html>
```

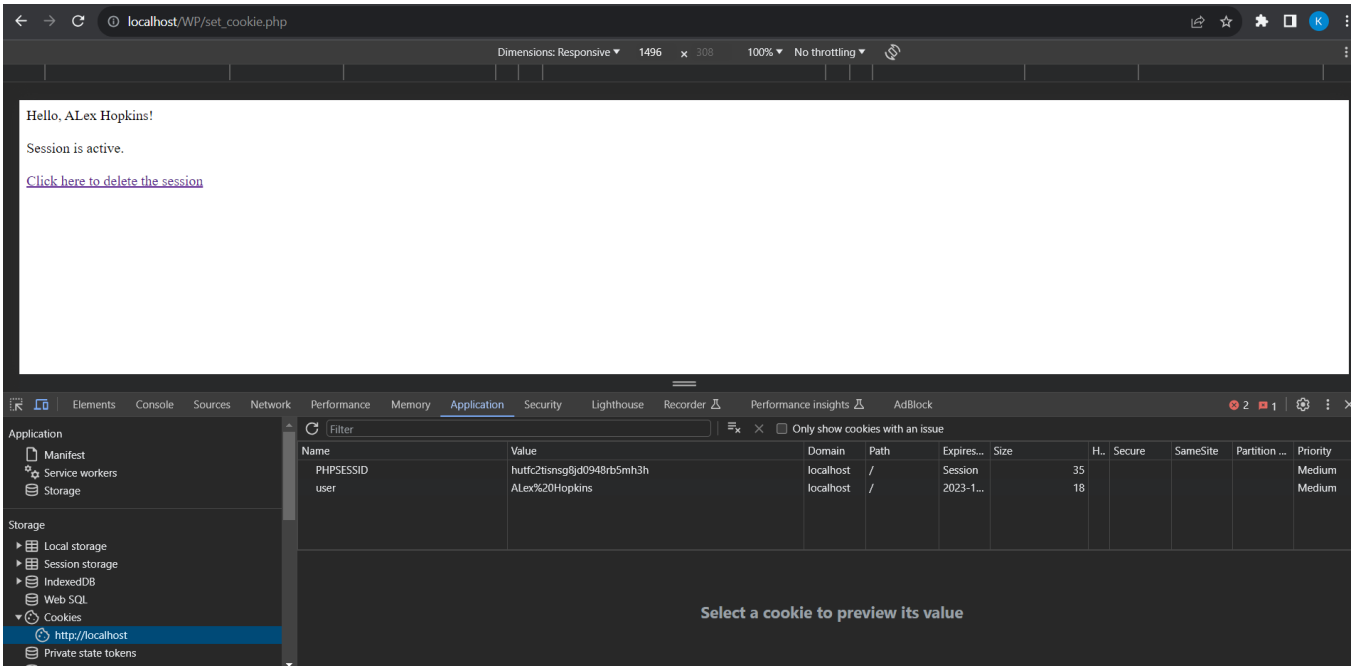


3) Start a session and check for timeout.

Code : Session Handling

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
<?php
// Set a cookie with a name 'user' and value 'Alex Hopkins'
setcookie('user', 'ALex Hopkins', time() + 3600, '/');

// Read the value of the 'user' cookie
if (isset($_COOKIE['user']))
{
    $username = $_COOKIE['user'];
```

```
    echo "Hello, $username!";
}
else
{
    echo "Cookie not set.";
}
echo "<br>";
echo "<br>";
?>
<?php
// Start a session
session_start();

// Set a session variable
$_SESSION['username'] = 'ALex Hopkins';

// Check for session timeout
$timeout = 3600;
if (isset($_SESSION['last_activity']) && (time() - $_SESSION['last_activity'] >
$timeout)) {
    // Session has expired, destroy it
    session_unset();
    session_destroy();

    echo "Session has timed out.";
} else {
    // Update last activity time
    $_SESSION['last_activity'] = time();
    echo "Session is active.";
}
?>



</body>
</html>
```



4) Delete the session.

Deleting the session

Code:

set_cookie.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
<?php
// Set a cookie with a name 'user' and value 'Alex Hopkins'
setcookie('user', 'ALex Hopkins', time() + 3600, '/');

// Read the value of the 'user' cookie
if (isset($_COOKIE['user']))
{
    $username = $_COOKIE['user'];
    echo "Hello, $username!";
}
else
{
    echo "Cookie not set.";
}
```

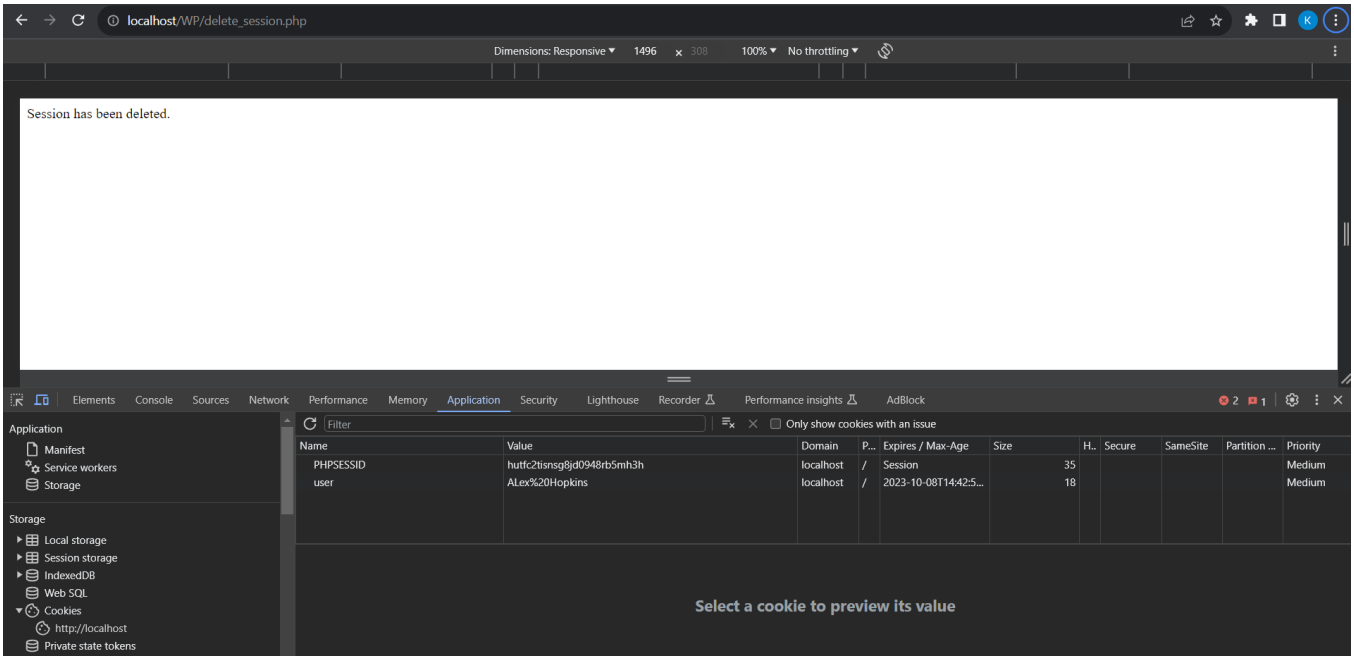
```
echo "<br>";
echo "<br>";
?>
<?php
// Start a session
session_start();

// Set a session variable
$_SESSION['username'] = 'ALex Hopkins';

// Check for session timeout
$timeout = 3600;
if (isset($_SESSION['last_activity']) && (time() - $_SESSION['last_activity'] >
$timeout)) {
    // Session has expired, destroy it
    session_unset();
    session_destroy();

    echo "Session has timed out.";
} else {
    // Update last activity time
    $_SESSION['last_activity'] = time();
    echo "Session is active.";
}
echo "<br>";
echo "<br>";
?>
<a href="delete_session.php">Click here to delete the session</a>




</body>
</html>
```

delete_session.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
<?php
// Set a cookie with a name 'user' and value 'Alex Hopkins'
setcookie('user', 'ALex Hopkins', time() + 3600, '/');
```

```php
// Read the value of the 'user' cookie
if (isset($_COOKIE['user']))
{
    $username = $_COOKIE['user'];
    echo "Hello, $username!";
}
else
{
    echo "Cookie not set.";
}
echo "<br>";
echo "<br>";
?>
<?php
// Start a session
session_start();

// Set a session variable
$_SESSION['username'] = 'ALex Hopkins';

// Check for session timeout
$timeout = 3600;
if (isset($_SESSION['last_activity']) && (time() - $_SESSION['last_activity'] >
$timeout)) {
    // Session has expired, destroy it
    session_unset();
    session_destroy();

    echo "Session has timed out.";
} else {
    // Update last activity time
    $_SESSION['last_activity'] = time();
    echo "Session is active.";
}
echo "<br>";
echo "<br>";
?>
<a href="delete_session.php">Click here to delete the session</a>




</body>
</html>
```

Session has been deleted after clicking the hyperlink.



Post Lab Questions:-

1. **Explain the process of destroying a single session variable and the process of destroying theentire session with suitable examples of each.**
   Ans:

In web development, sessions are used to store and manage user-specific data across multiple page requests. Session variables are used to store data that should persist throughout a user's session on a website. PHP, for example, provides a way to create and manage session variables. Let's discuss how to destroy a single session variable and the entire session in PHP with suitable examples for each.

- **Destroying a Single Session Variable:** To destroy a single session variable in PHP, you can use the unset() function. Here's an example:

```
<?php
// Start the session
session_start();

// Set a session variable
$_SESSION['username'] = 'john_doe';

// Destroy a single session variable
unset($_SESSION['username']);

// The 'username' session variable is now destroyed
    ?>
```

In this example, we first start the session using session_start(), then set a session variable named 'username'. We can destroy the 'username' session variable using unset($_SESSION['username']).

- **Destroying the Entire Session:** To destroy the entire session in PHP, you can use the session_destroy() function. Here's an example:

```
<?php
// Start the session
session_start();

// Set some session variables
$_SESSION['username'] = 'john_doe';
$_SESSION['user_id'] = 123;

// Destroy the entire session
session_destroy();

// All session variables are now destroyed
?>
```

In this example, we first start the session, set some session variables, and then use session_destroy() to completely destroy the session. This will remove all session variables and invalidate the session ID, making the session unavailable.

2. What are the different types of Cookies? What is the difference between them?
Ans:

1. Session Cookies:
- Session cookies are temporary cookies that are stored in the browser's memory and are deleted when the user closes their browser.

2. Persistent Cookies:
- Persistent cookies have an expiration date and are stored on the user's device even after the browser is closed.

3. Secure Cookies:
- Secure cookies are transmitted over secure, encrypted connections (HTTPS).
- They are often used for sensitive data like login credentials or session IDs to enhance security.

4. HttpOnly Cookies:
- HttpOnly cookies cannot be accessed via JavaScript, which adds a layer of security by protecting them from cross-site scripting (XSS) attacks.

5. Same-Site Cookies:
- Same-Site cookies specify whether a cookie should be sent with cross-origin requests.
- They help protect against cross-site request forgery (CSRF) attacks by controlling when cookies are sent to a different site.

6. Third-party Cookies:
- Third-party cookies are set by a domain other than the one the user is currently visiting.
- They are often used by advertisers and tracking services to collect user data across multiple websites.
- Many web browsers have implemented restrictions on third-party cookies due to privacy concerns.

7. First-party Cookies:

- First-party cookies are set by the domain the user is currently visiting.
- They are typically used to store user-specific information or preferences for that particular website.

Difference between Cookies:

- The main differences between these types of cookies lie in their lifespan, security, and usage. Session cookies are temporary, while persistent cookies have a specified expiration date. Secure and HttpOnly cookies enhance security. Same-Site cookies control cross-origin behavior, and first-party and third-party cookies depend on the domain that sets them.

3. How website cookies can be made to comply with Data Privacy Regulations?

Ans:

Compliance with data privacy regulations, such as the General Data Protection Regulation (GDPR) in Europe or the California Consumer Privacy Act (CCPA) in the United States, is essential when using website cookies. To ensure that website cookies are compliant with data privacy regulations, consider the following best practices:

1. Transparency and Consent:

- Clearly inform users about the use of cookies on your website through a cookie policy or a cookie banner.
- Obtain explicit consent from users before setting non-essential cookies. Users should have the option to accept or reject these cookies.
- Allow users to easily change their cookie preferences, including withdrawing consent.

2. Cookie Classification:

- Classify cookies into categories (e.g., essential, functional, analytics, advertising) to help users understand their purpose.
- Ensure that essential cookies, which are necessary for the website's core functionality, are set without requiring user consent.

3. Data Minimization:

- Only collect and store the data that is necessary for the specified purpose. Avoid collecting excessive or unnecessary user data.

4. Anonymization and Pseudonymization:

- Anonymize or pseudonymize user data whenever possible to reduce the risk of identifying individuals.

Outcomes: CO 2: Design forms and use session handling mechanism with web applications

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

Thus we successfully implemented Cookie and session handling using PHP and understood the same.

Signature of faculty in-charge with date

References:

1. https://www.guru99.com/cookies-and-sessions.html

2. https://www.massey.ac.nz/~nhreyes/MASSEY/159339/Lectures/Lecture%2011%20-%20PHP%20-%20Part%205%20-%20CookiesSessions.**pdf**

3. Thomson PHP and MySQL Web Development Addison-Wesley Professional , 5th Edition 2016.