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

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

Experiment No. 06

TITLE: Cookies and Session in PHP

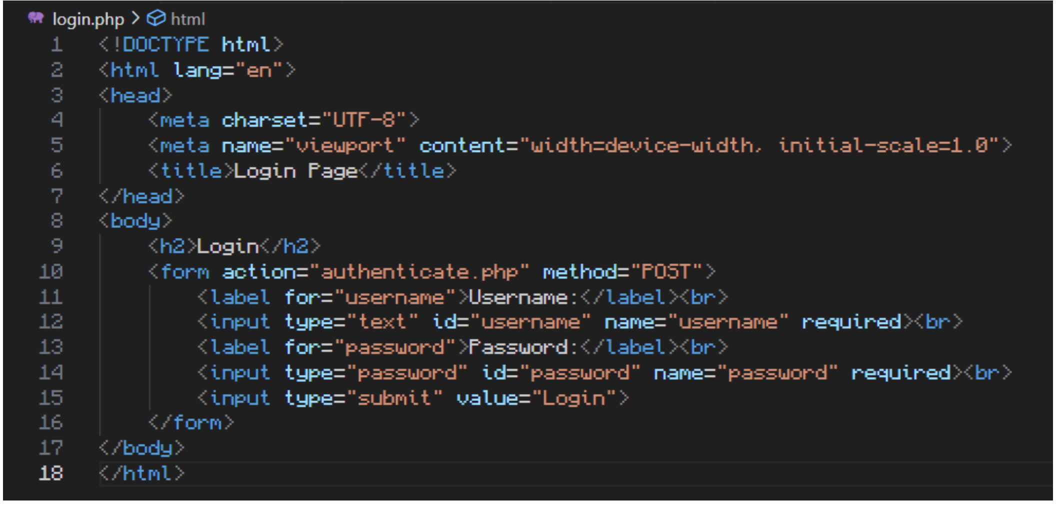
AIM: The aim of this experiment is to learn how to create and destroy cookies and sessions in PHP

Expected Outcome of Experiment:
The expected outcome of the experiment is understanding of how to effectively manage cookies and sessions in PHP.

Problem Statement: Description of the application implemented with output:

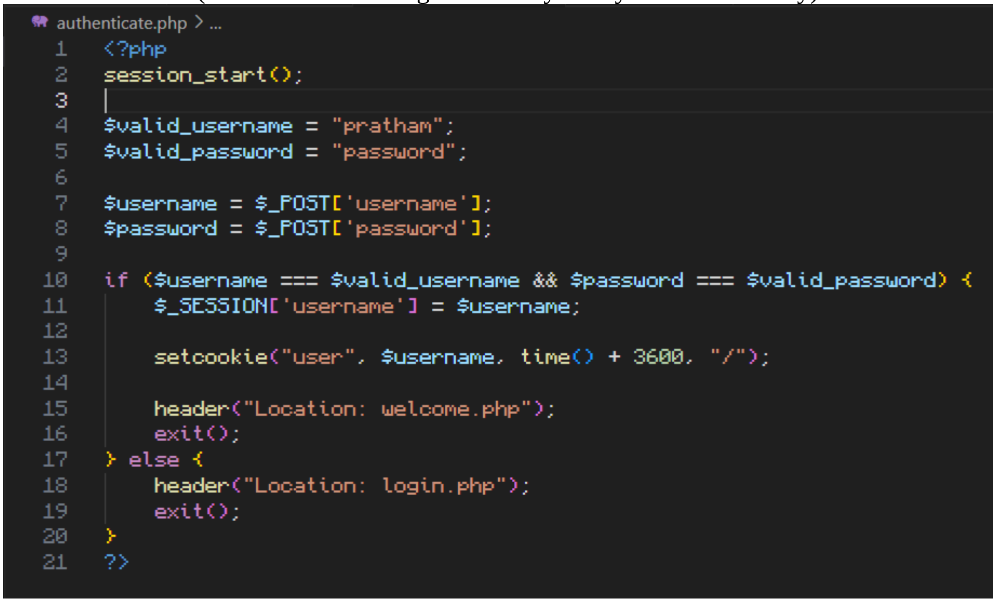
1. Create a secure login page with fields for username and password. Implement authentication of user credentials. Upon successful login, establish a session and set cookies, then redirect the user to a welcome page.
2. Upon refreshing the welcome page, ensure the session and cookies are properly destroyed.

Implementation Details
Login.php



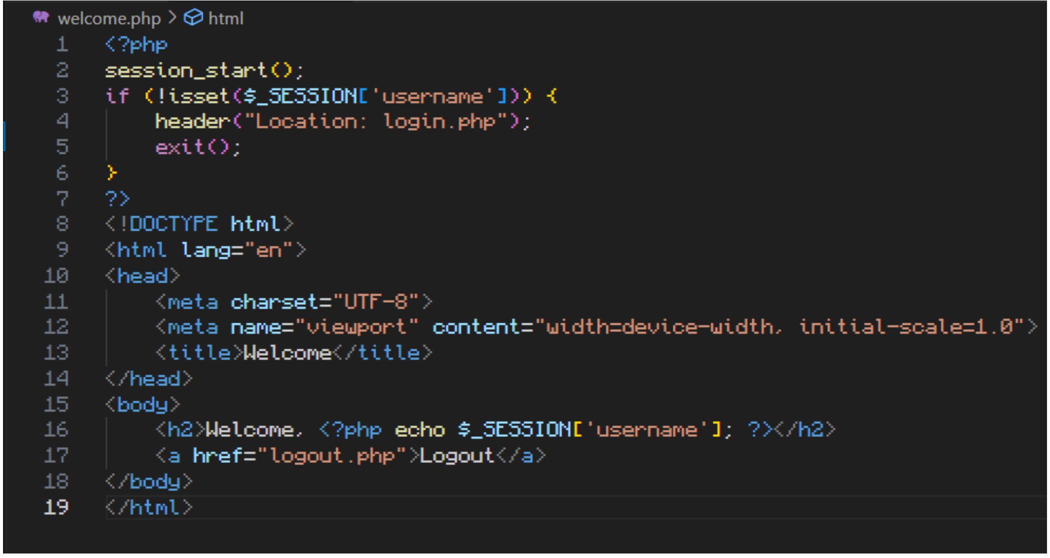
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Page</title>
</head>
<body>
    <h2>Login</h2>
    <form action="authenticate.php" method="POST">
        <label for="username">Username:</label><br>
        <input type="text" id="username" name="username" required><br>
        <label for="password">Password:</label><br>
        <input type="password" id="password" name="password" required><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

Authenticate.php



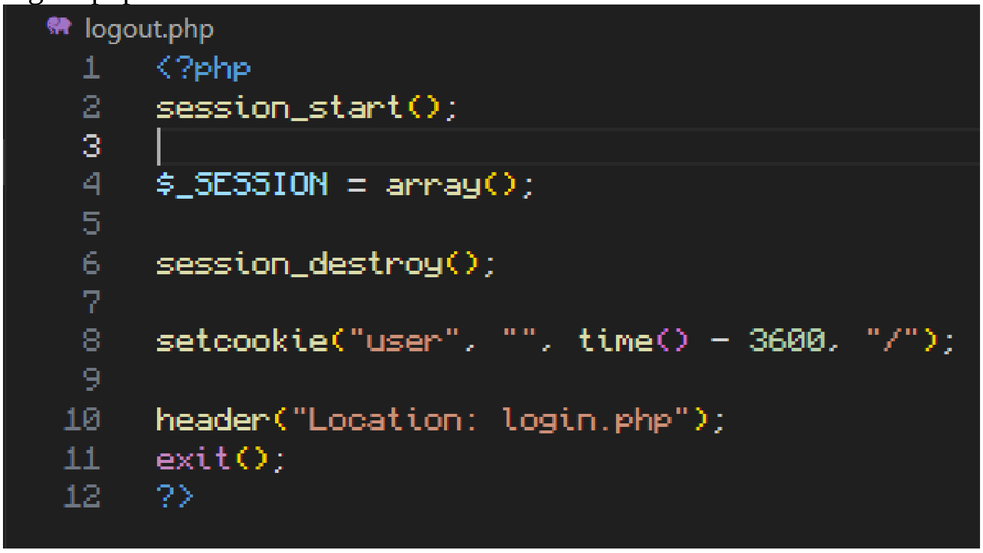
```php
<?php
session_start();

$valid_username = "pratham";
$valid_password = "password";

$username = $_POST['username'];
$password = $_POST['password'];

if ($username === $valid_username && $password === $valid_password) {
    $_SESSION['username'] = $username;

    setcookie("user", $username, time() + 3600, "/");

    header("Location: welcome.php");
    exit();
} else {
    header("Location: login.php");
    exit();
}
?>
```

welcome.php



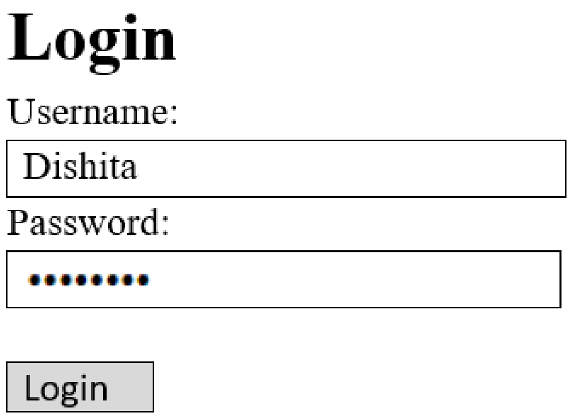
```php
<?php
session_start();
if (!isset($_SESSION['username'])) {
    header("Location: login.php");
    exit();
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome</title>
</head>
<body>
    <h2>Welcome, <?php echo $_SESSION['username']; ?></h2>
    <a href="logout.php">Logout</a>
</body>
</html>
```

logout.php



```php
<?php
session_start();

$_SESSION = array();

session_destroy();

setcookie("user", "", time() - 3600, "/");

header("Location: login.php");
exit();
?>
```

Output



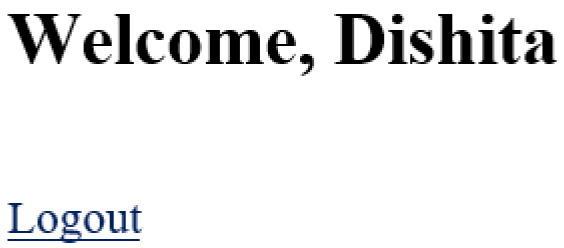
Login

Username:

Dishita

Password:

••••••••



Welcome, Dishita

Post Lab Objective with Ans :

1. Differentiate between sessions and cookies

Ans: Sessions and cookies are both used in web development for managing user state, but they differ in key aspects. Sessions store data on the server and are more secure, suitable for storing sensitive information. Cookies, on the other hand, are stored on the client-side and are typically used for small, non-sensitive data. While sessions expire after a period of inactivity or when the browser is closed, cookies can have different expiration times. Additionally, sessions are accessible only on the server-side, while cookies can be accessed on the client-side, including through JavaScript.

2. Describe the methods used to properly destroy sessions and cookies in a web application.

Ans: To properly destroy sessions in a web application, use `session_destroy()` in PHP to remove session data. For cookies, set their expiration time to a past value with `setcookie()` to delete them. These actions are typically performed when a user logs out or when the session or cookie is no longer needed.

Example:

```
setcookie("user", "", time() - 3600, "/");
```

3. Discuss strategies for implementing security measures in form sessions

Ans: To enhance security in form sessions:

1. Use HTTPS to encrypt data.

2. Store session IDs in secure, HttpOnly cookies.

3. Set session timeout for inactivity.

4. Regenerate session IDs after authentication.

5. Generate strong, unpredictable session IDs.

6. Validate and sanitize session data.

7. Optionally tie sessions to IP addresses.

8. Implement logging and monitoring.

9. Allow session revocation by users.