FEU DILIMAN

DEPARTMENT OF COMPUTER STUDIES

(Applications Development and Emerging Technologies)

PRE-SUMMATIVE ASSESSMENT

5

USER INTERFACES, SESSION AND COOKIES

Student Name	Jan Andrei Reyes
Section:	TN22
Professor:	Ms. Kim Giselle Bautista

I. PROGRAM OUTCOME/S (PO) ADDRESSED BY THE LABORATORY EXERCISE

- Design, implement and evaluate computer-based systems or applications to meet desired needs and requirements.

II. COURSE LEARNING OUTCOME/S (CLO) ADDRESSED BY THE LABORATORY EXERCISE

- Understand and apply best practices and standards in the development of website.

III. INTENDED LEARNING OUTCOME/S (ILO) OF THE LABORATORY EXERCISE

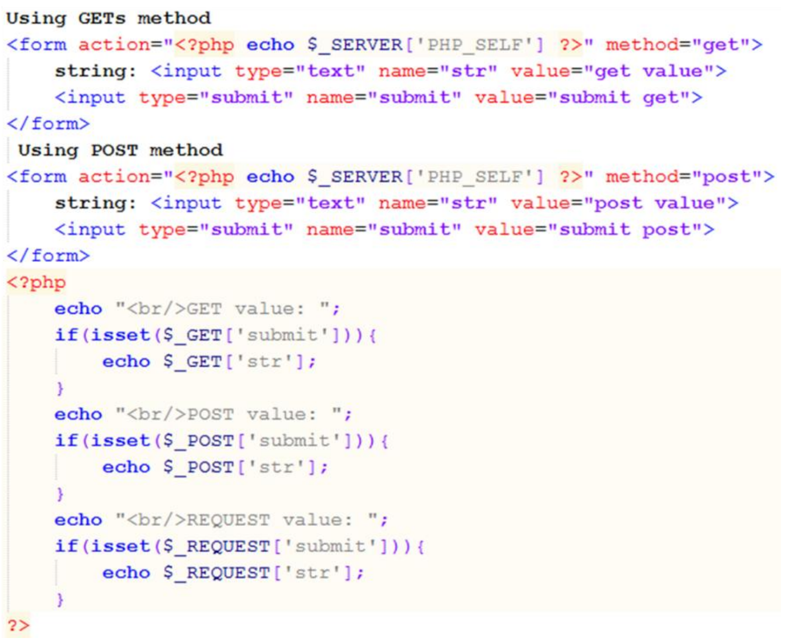
At the end of this exercise, students must be able to:

- To apply the available super global variables for form processing and validation.
- To differentiate the use of $_GET, $_POST, and $_REQUEST super global variable in form processing and know when to use it.
- To differentiate the use of Session and Cookies for form security of a Web Site.
- To know the proper syntax for validating user inputs using Regular Expression.

IV. BACKGROUND INFORMATION

Superglobal variables

Example:



```
Using GETs method
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="get">
    string: <input type="text" name="str" value="get value">
    <input type="submit" name="submit" value="submit get">
</form>
 Using POST method
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="post">
    string: <input type="text" name="str" value="post value">
    <input type="submit" name="submit" value="submit post">
</form>
<?php
    echo "<br/>GET value: ";
    if(isset($_GET['submit'])){
        echo $_GET['str'];
    }
    echo "<br/>POST value: ";
    if(isset($_POST['submit'])){
        echo $_POST['str'];
    }
    echo "<br/>REQUEST value: ";
    if(isset($_REQUEST['submit'])){
        echo $_REQUEST['str'];
    }
?>
```

PHP Superglobal - $GLOBALS

Super global variables are built-in variables that are always available in all scopes.

PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).
PHP stores all global variables in an array called $GLOBALS[*index*].
The *index* holds the name of the variable.
The example below shows how to use the super global variable $GLOBALS:

Example

```
<?php
$x = 75;
$y = 25;

function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

PHP Superglobal - $_POST

Super global variables are built-in variables that are always available in all scopes.

PHP $_POST

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:

Example

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_POST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```

PHP Superglobal - $_GET

Super global variables are built-in variables that are always available in all scopes.

PHP $_GET

PHP $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$_GET can also collect data sent in the URL.

Assume we have an HTML page that contains a hyperlink with parameters:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>
```

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

When a user clicks on the link "Test $GET", the parameters "subject" and "web" are sent to "test_get.php", and you can then access their values in "test_get.php" with $_GET.

The example below shows the code in "test_get.php":

Example

```
<html>
<body>

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>

</body>
</html>
```

PHP Superglobal - $_REQUEST

Super global variables are built-in variables that are always available in all scopes.

PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:

Example

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
```

```
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_REQUEST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```

PHP Cookies

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 will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the `setcookie()` function.

Syntax

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the `isset()` function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); //
86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Note: The `setcookie()` function must appear BEFORE the <html> tag.

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use `setrawcookie()` instead).

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the `setcookie()` function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
```

```
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Delete a Cookie

To delete a cookie, use the `setcookie()` function with an expiration date in the past:

Example

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

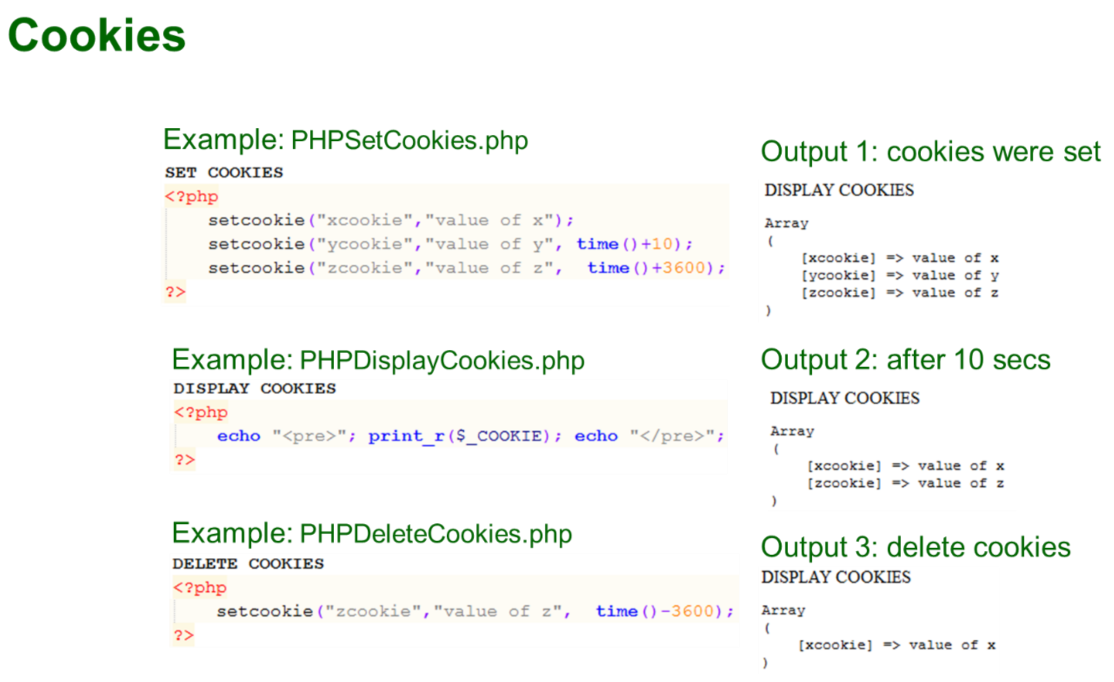
The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the `setcookie()` function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
  echo "Cookies are enabled.";
} else {
  echo "Cookies are disabled.";
}
?>

</body>
</html>
```



Cookies

Example: PHPSetCookies.php

SET COOKIES

```
<?php
    setcookie("xcookie","value of x");
    setcookie("ycookie","value of y",  time()+10);
    setcookie("zcookie","value of z",   time()+3600);
?>
```

Output 1: cookies were set

DISPLAY COOKIES

```
Array
(
    [xcookie] => value of x
    [ycookie] => value of y
    [zcookie] => value of z
)
```

Example: PHPDisplayCookies.php

DISPLAY COOKIES

```
<?php
    echo "<pre>"; print_r($_COOKIE); echo "</pre>";
?>
```

Output 2: after 10 secs

DISPLAY COOKIES

```
Array
(
    [xcookie] => value of x
    [zcookie] => value of z
)
```

Example: PHPDeleteCookies.php

DELETE COOKIES

```
<?php
    setcookie("zcookie","value of z",   time()-3600);
?>
```

Output 3: delete cookies

DISPLAY COOKIES

```
Array
(
    [xcookie] => value of x
)
```

PHP Sessions

A session is a way to store information (in variables) to be used across multiple pages.

Unlike a cookie, the information is not stored on the users computer.

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

Tip: If you need a permanent storage, you may want to store the data in a database.

Start a PHP Session

A session is started with the `session_start()` function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
```

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

Note: The `session_start()` function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (`session_start()`).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

Another way to show all the session variable values for a user session is to run the following code:

Example

```
<?php
session_start();
?>
```

```
<!DOCTYPE html>
<html>
<body>
<?php
print_r($_SESSION);
?>
</body>
</html>
```

Modify a PHP Session Variable

To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use `session_unset()` and `session_destroy()`:

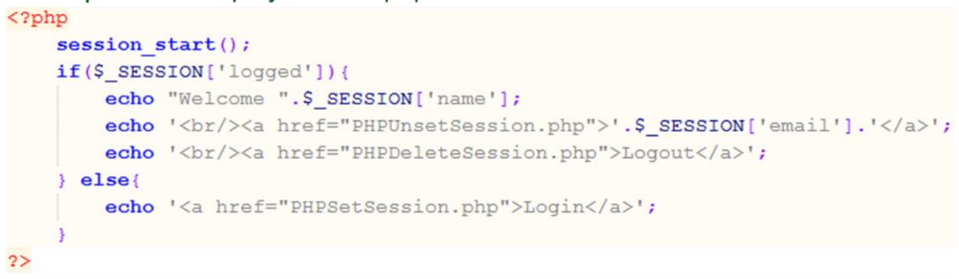
Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
```

```
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?>
</body>
</html>
```

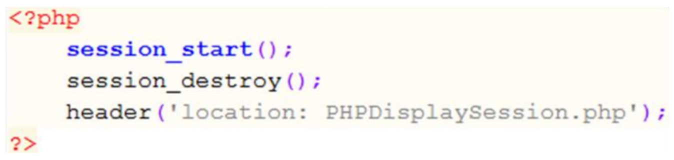
Session

Example: PHPDisplaySession.php



```
<?php
    session_start();
    if($_SESSION['logged']){
        echo "Welcome ".$_SESSION['name'];
        echo '<br/><a href="PHPUnsetSession.php">'.$_SESSION['email'].'</a>';
        echo '<br/><a href="PHPDeleteSession.php">Logout</a>';
    } else{
        echo '<a href="PHPSetSession.php">Login</a>';
    }
?>
```

Example: PHPDeleteSession.php



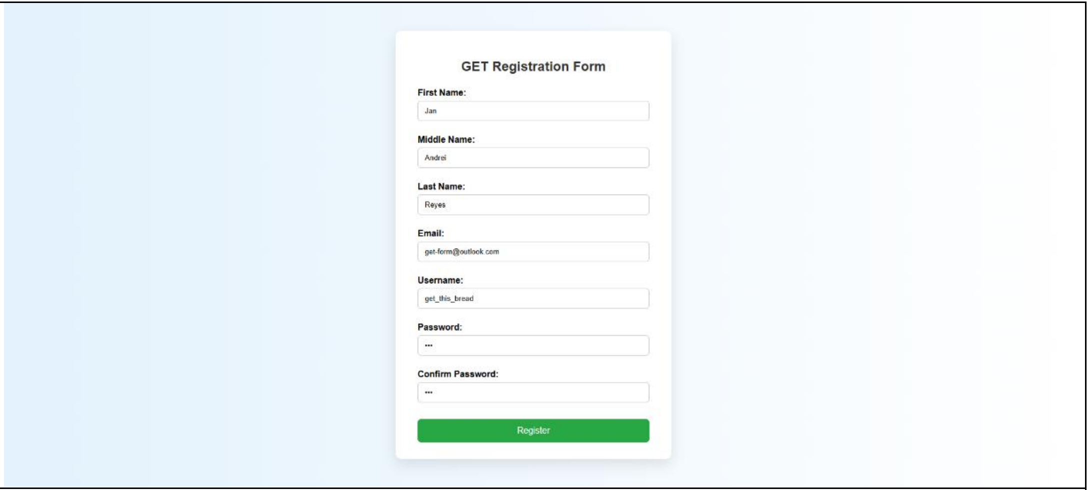
```
<?php
    session_start();
    session_destroy();
    header('location: PHPDisplaySession.php');
?>
```

V. GRADING SYSTEM / RUBRIC (please see separate sheet)

VI. LABORATORY ACTIVITY

1. Create a personal information webpage. Using $_GET and $_POST get the data (firstname, middlename, lastname, date of birth and address) from the forms and display below.

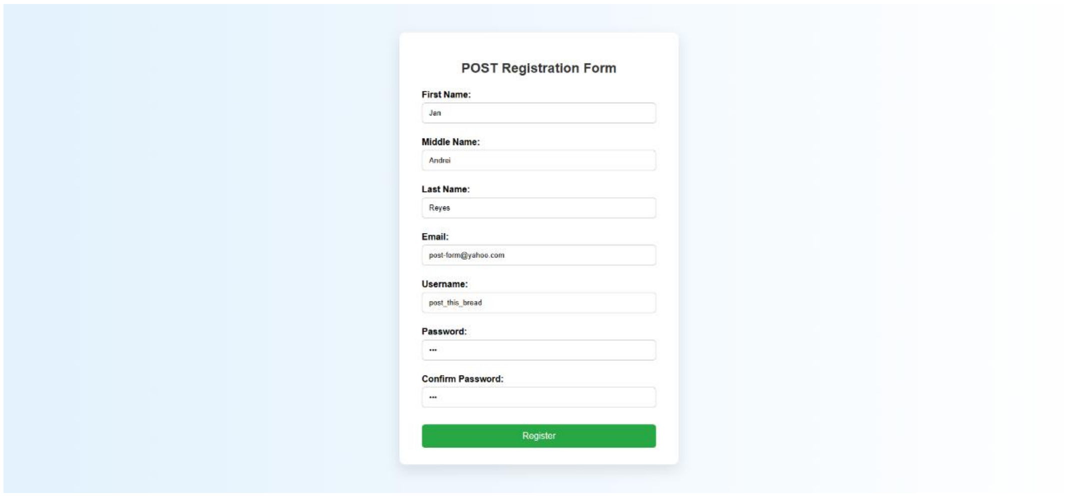
GET



GET Registration Form

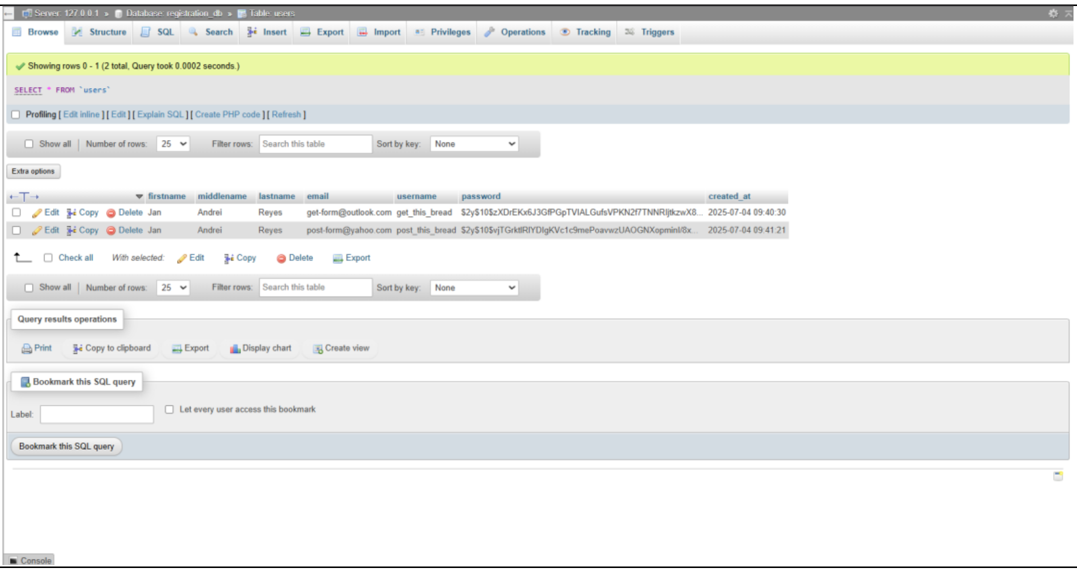
First Name:	Jan
Middle Name:	Andrei
Last Name:	Reyes
Email:	get-form@outlook.com
Username:	get_this_bread
Password:	•••
Confirm Password:	•••

POST



POST Registration Form

First Name:	Jan
Middle Name:	Andrei
Last Name:	Reyes
Email:	post-form@yahoo.com
Username:	post_this_bread
Password:	•••
Confirm Password:	•••



Server: 127.0.0.1 » Database: registration_db » Table: users
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
Triggers

✓ Showing rows 0 - 1 (2 total, Query took 0.0002 seconds.)

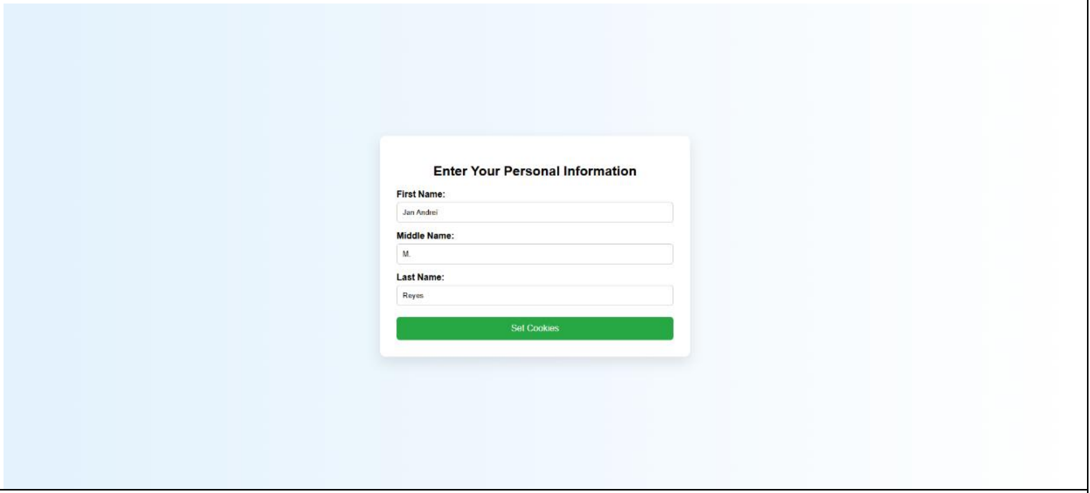
```
SELECT * FROM `users`
```

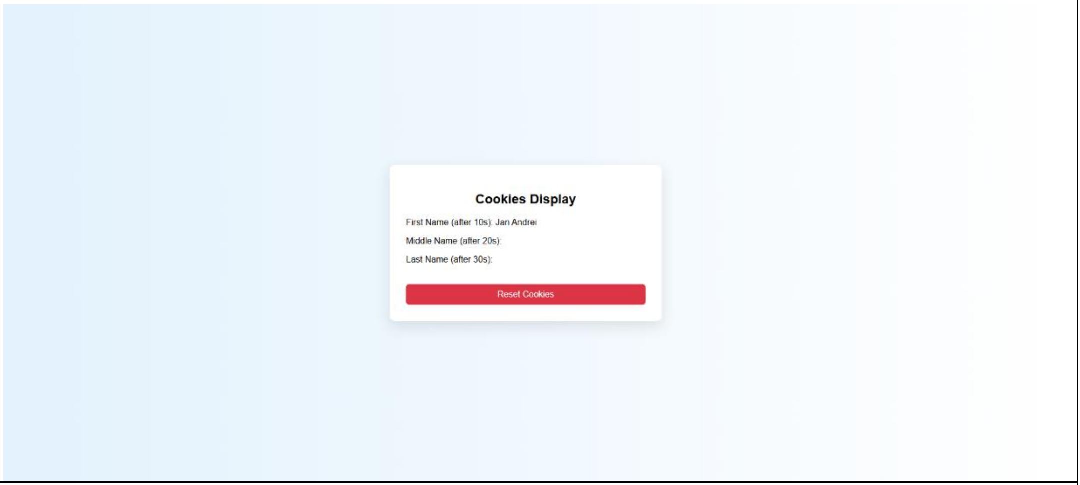
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

	firstname	middlename	lastname	email	username	password	created_at
Edit Copy Delete	Jan	Andrei	Reyes	get-form@outlook.com	get_this_bread	$2y$10$zXDrEKx6J3GfPGpTVIALGufsVPKN2f7TNNRIjtkzwX8...	2025-07-04 09:40:30
Edit Copy Delete	Jan	Andrei	Reyes	post-form@yahoo.com	post_this_bread	$2y$10$vjTGrktlRIYDlgKVc1c9mePoavwzUAOGNXopminl/8x...	2025-07-04 09:41:21

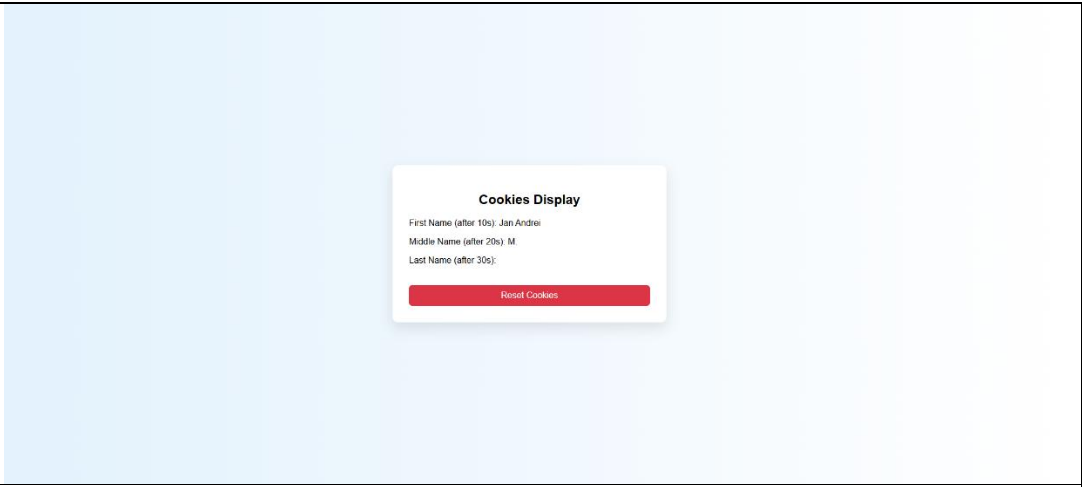
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Print
Copy to clipboard
Export
Display chart
Create view
Bookmark this SQL query
Label:
Let every user access this bookmark
Bookmark this SQL query
Console

2. Create a personal information webpage. Using setcookie() function, create cookies for the firstname, middle name and lastname. It must be show the cookies after 10, 20 and 30 seconds.

SET Cookie
 Enter Your Personal Information First Name: Jan Andrei Middle Name: M. Last Name: Reyes
AFTER 10 SECONDS

 Cookies Display
First Name (after 10s): Jan Andrei
Middle Name (after 20s):
Last Name (after 30s):

AFTER 20 SECONDS



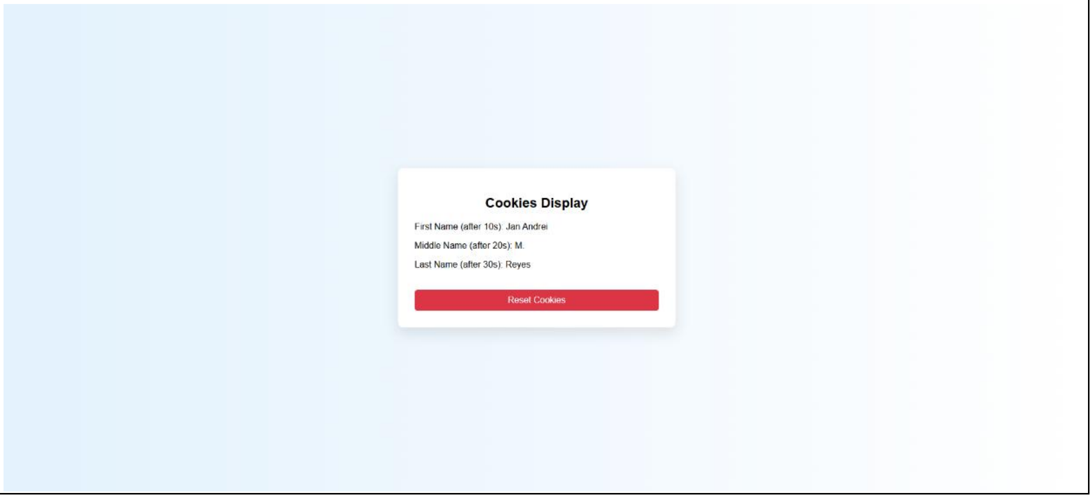
Cookies Display

First Name (after 10s): Jan Andrei

Middle Name (after 20s): M.

Last Name (after 30s):

AFTER 30 SECONDS



Cookies Display

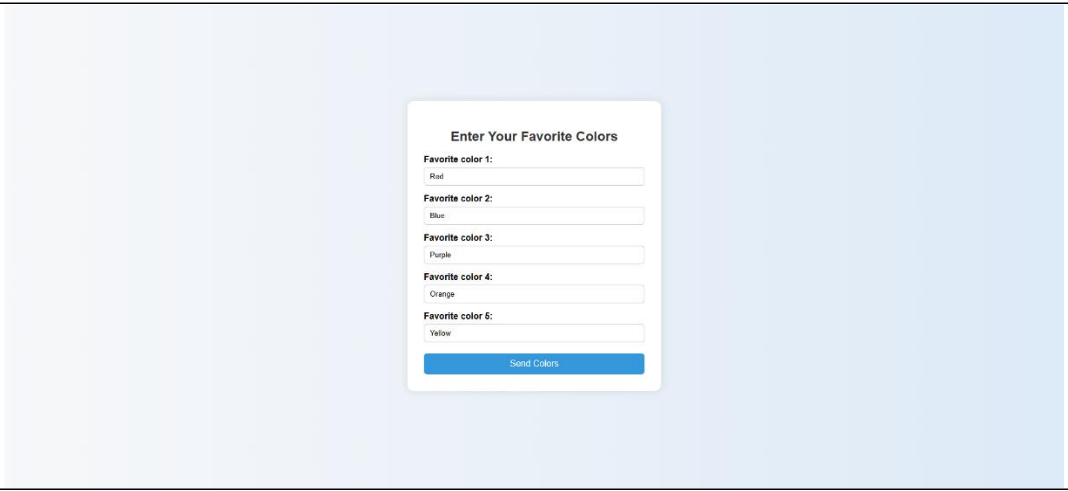
First Name (after 10s): Jan Andrei

Middle Name (after 20s): M.

Last Name (after 30s): Reyes

3. Create a webpage that will set a session for your 5 favorite colors and use it to the other webpage. Integrate HTML and CSS, you can create your own design and include images. Please see the example below:

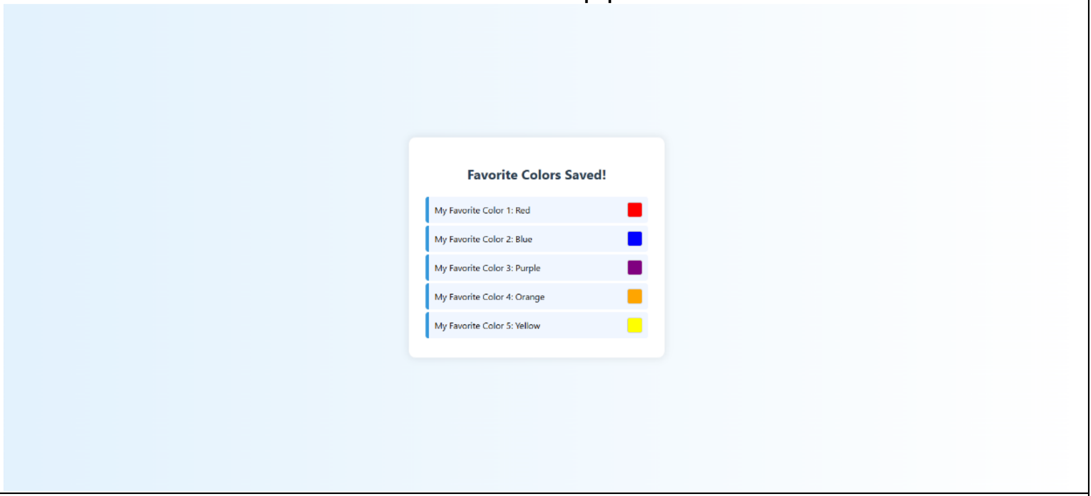
FavoriteColors.php



Enter Your Favorite Colors

Favorite color 1:	Red
Favorite color 2:	Blue
Favorite color 3:	Purple
Favorite color 4:	Orange
Favorite color 5:	Yellow

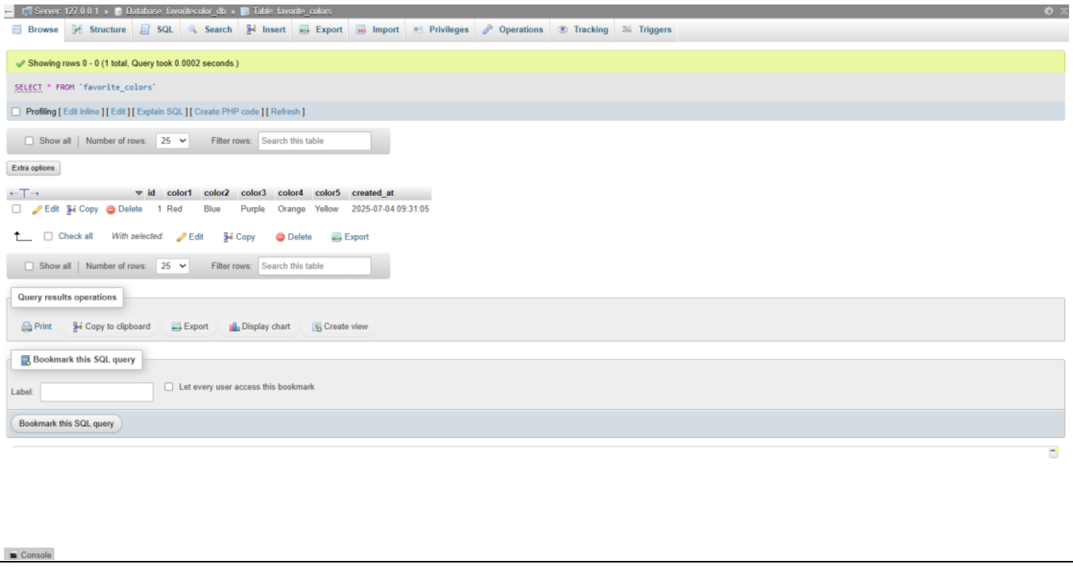
ShowColors.php



Favorite Colors Saved!

My Favorite Color 1: Red	🟥
My Favorite Color 2: Blue	🟦
My Favorite Color 3: Purple	🟪
My Favorite Color 4: Orange	🟧
My Favorite Color 5: Yellow	🟨

Database Result



Server: 127.0.0.1 » Database: favoritecolor_db » Table: favorite_colors
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

✓ Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

```
SELECT * FROM `favorite_colors`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options

	id	color1	color2	color3	color4	color5	created_at
Edit Copy Delete	1	Red	Blue	Purple	Orange	Yellow	2025-07-04 09:31:05

Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table
Query results operations
Print Copy to clipboard Export Display chart Create view
Bookmark this SQL query
Label:
Let every user access this bookmark
Bookmark this SQL query
Console

Snip and paste your source codes here. Snip it directly from the IDE so that colors of the codes are preserved for readability. Include additional pages if necessary.

VII. QUESTION AND ANSWER

1. **What is the usage of super global?**
   Super globals are built-in PHP variables used to access data like form inputs, session info, and server details from anywhere in the script.

2. **What are the differences between $_POST and $_GET?**
   $_GET sends data through the URL and is visible, while $_POST sends data through the request body and is hidden, making it more secure for sensitive information.

3. **What is the importance of cookies on a webpage or website?**
   Cookies store small data in the browser to remember users, preferences, or track activity.

4. **What is a session?**
   A session stores user data on the server and keeps it across multiple pages.

5. **What is the importance of session?**
   Sessions help track users, store secure data, and personalize the website experience.

VIII. REFERENCES

1. https://www.w3schools.com/css/
2. https://www.w3schools.com/html/
3. https://www.w3schools.com/php/php_variables.asp
4. https://www.w3schools.com/php/php_superglobals_globals.asp
5. https://www.w3schools.com/php/php_superglobals_post.asp
6. https://www.w3schools.com/php/php_superglobals_get.asp
7. https://www.w3schools.com/php/php_superglobals_request.asp
8. https://www.w3schools.com/php/php_cookies.asp
9. https://www.w3schools.com/php/php_sessions.asp

Note: The following rubrics/metrics will be used to grade students' output.

Program (100 pts.)	**(Excellent)**	**(Good)**	**(Fair)**	**(Poor)**
Program execution (20pts)	Program executes correctly with no syntax or runtime errors **(18-20pts)**	Program executes with less than 3 errors **(15-17pts)**	Program executes with more than 3 errors **(12-14pts)**	Program does not execute **(10-11pts)**
Correct output (20pts)	Program displays correct output with no errors **(18-20pts)**	Output has minor errors **(15-17pts)**	Output has multiple errors **(12-14pts)**	Output is incorrect **(10-11pts)**
Design of output (10pts)	Program displays more than expected **(10pts)**	Program displays minimally expected output **(8-9pts)**	Program does not display the required output **(6-7pts)**	Output is poorly designed **(5pts)**
Design of logic (20pts)	Program is logically well designed **(18-20pts)**	Program has slight logic errors that do no significantly affect the results **(15-17pts)**	Program has significant logic errors **(3-5pts)**	Program is incorrect **(10-11pts)**
Standards (20pts)	Program code is stylistically well designed **(18-20pts)**	Few inappropriate design choices (i.e. poor variable names, improper indentation) **(15-17pts)**	Several inappropriate design choices (i.e. poor variable names, improper indentation) **(12-14pts)**	Program is poorly written **(10-11pts)**
Delivery (10pts)	The program was delivered on time. **(10pts)**	The program was delivered a day after the deadline. **(8-9pts)**	The program was delivered two days after the deadline. **(6-7pts)**	The program was delivered more than two days after the deadline. **(5pts)**