**QUESTION BANK ON PHP PA SECOND**

**PART A**

1. **State the use of Cookies (any four)**

**Remembering User Preferences**: Cookies can be used to remember user-specific settings, such as language preferences, theme choices, or layout preferences.

**User Authentication and Sessions**: Cookies are often used to maintain user sessions. When a user logs in, a session cookie is usually created to identify the user for subsequent requests.

**Personalization:** Cookies can be used to personalize content for users based on their preferences or previous interactions with the website.

**Shopping Carts**: In e-commerce websites, cookies can be used to store information about the items a user has added to their shopping cart.

**Remembering User Login**: Cookies can be used to remember a user's login credentials, allowing them to stay logged in even after closing the browser.

**Tracking User Behavior**: Cookies can be used to track user behavior on a website, such as which pages they visit, how long they stay on a page, etc. This information can be used for analytics and marketing purposes.

**Managing Advertisements**: Cookies can be used to track user interactions with advertisements, allowing for targeted advertising campaigns.

1. **State AND EXLPLAIN role of GET and POST methods.**

Get and Post Methods in PHP

PHP provides these two methods through which a client (browser) can send information to the server.

Get and Post methods are the HTTP request methods used inside the <form> tag to send form data to the server.

HTTP protocol enables the communication between the client and the server.

The GET method is used to submit the HTML form data. This data is collected by the predefined $\_GET variable for processing.

POST method is also used to submit the HTML form data. But the data submitted by this method is collected by the predefined superglobal variable $\_POST

The POST method is useful for sending any sensitive information because the information sent using the POST method is not visible to anyone.

There is no limitation on size of data to be sent using the POST Method. You can send a large amount of information using this method.

Binary and ASCII data can also be sent using the POST method.

Data security depends on the HTTP protocol because the information sent using the POST method goes through the HTTP header. By using secure HTTP, you can ensure that your data is safe

The GET method should not be used while sending any sensitive information.

A limited amount of data can be sent using method = "get". This limit should not exceed 2048 characters.

For security reasons, never use the GET method to send highly sensitive information like username and password, because it shows them in the URL.

The GET method cannot be used to send binary data (such as images or word documents) to the server.

1. **Define Introspection. Explain it with suitable example.**

Introspection refers to the ability of a program to examine or analyze its own structure, properties, and behavior at runtime. It allows a program to gain insight into its own state and capabilities. In computer science, introspection is a powerful tool for dynamic languages and runtime environments.

1. Getting the Type of a Variable gettype():

$variable = 42;

echo gettype($variable); // Output: integer

2. Checking if a Variable is of a Certain Type is\_\*():

$array = [1, 2, 3];

$is\_array = is\_array($array); // $is\_array will be true

$string = "Hello";

$is\_string = is\_string($string); // $is\_string will be true

3. Getting the Class Name of an Object geet\_class():

class MyClass {

// Class definition

}

$object = new MyClass();

echo get\_class($object); // Output: MyClass

4. Checking if a Method Exists in a Class method\_exists():

class MyClass {

public function myMethod() {

// Method implementation

}

}

$object = new MyClass();

$has\_method = method\_exists($object, 'myMethod'); // $has\_method will be true

5. Checking if a Property Exists in a Class property\_exists():

class MyClass {

public $myProperty;

}

$object = new MyClass();

$has\_property = property\_exists($object, 'myProperty'); // $has\_property will be true

6. Getting the Methods of a Class get\_class\_methods():

class MyClass {

public function method1() {}

public function method2() {}

}

$methods = get\_class\_methods('MyClass');

print\_r($methods); // Output: Array ( [0] => method1 [1] => method2 )

7.Getting the Properties of an Object get\_object\_vars():

class MyClass {

public $property1 = 'Value 1';

public $property2 = 'Value 2';

}

$object = new MyClass();

$properties = get\_object\_vars($object);

print\_r($properties);

// Output: Array ( [property1] => Value 1 [property2] => Value 2 )

These examples demonstrate how introspection functions in PHP can be used to examine variables, classes, methods, and properties at runtime

1. **Describe : (i) Start session (ii) Get session variables**

In web development, sessions are used to maintain stateful information between multiple HTTP requests from a user. PHP provides built-in functions to work with sessions.

(i) Start Session:

Starting a session in PHP involves using the session\_start() function. This function initializes a session or resumes an existing one.

<?php

session\_start();

?>

This line should be placed at the beginning of every PHP script that wants to use session variables. It's typically placed at the very top of the PHP file, before any HTML or other output.

(ii) Get Session Variables:

Once a session is started, you can store and retrieve data in the session using the $\_SESSION superglobal array. This array is available throughout the entire session and allows you to store information that persists between different pages or requests.

<?php

session\_start();

// Set session variables

$\_SESSION['username'] = 'JohnDoe';

$\_SESSION['email'] = 'john@example.com';

?>

To retrieve these variables on subsequent requests, you can use the $\_SESSION array:

<?php

session\_start();

// Get session variables

$username = $\_SESSION['username'];

$email = $\_SESSION['email'];

echo "Welcome back, $username! Your email is $email.";

?>

Keep in mind that sessions rely on cookies to maintain the session ID, so the client's browser must have cookies enabled for sessions to work. Additionally, sessions should be started at the beginning of every PHP script that needs to access session data.Also, remember to call session\_start() before any output is sent to the browser, as it sets HTTP headers and must come before any content is sent.

1. **State and explain any four form controls to get user’s input in PHP**

In PHP, you can use HTML form controls to get user input. Here are four common form controls that are used to collect user input:

Text Input:

This control allows the user to enter a single line of text.

<input type="text" name="username">

In PHP, you can access the value entered by the user using $\_POST or $\_GET (depending on the form's method attribute):

$username = $\_POST ['username']; // Assuming the form uses POST method

Textarea:

This control allows the user to enter multiple lines of text.

<textarea name="comments" rows="4" cols="50"></textarea>

$comments = $\_POST['comments']; // Assuming the form uses POST method

Checkbox:

This control allows the user to select one or more options from a set of choices.

<input type="checkbox" name="interests[]" value="sports"> Sports

<input type="checkbox" name="interests[]" value="music"> Music

$selected\_interests = $\_POST['interests'];

Radio Buttons:

This control allows the user to select only one option from a set of choices.

<input type="radio" name="gender" value="male"> Male

<input type="radio" name="gender" value="female"> Female

$gender = $\_POST['gender'];

1. **Write steps to create database using PHP.**

To create a database using PHP, you'll need to interact with a database management system (DBMS) like MySQL or SQLite. Here are the steps to create a database using PHP and MySQL as an example:

Set Up Database Connection:

Start by establishing a connection to your MySQL server using the mysqli\_connect() function. You'll need to provide the hostname, username, password, and optionally the port number.

$servername = "localhost";

$username = "username";

$password = "password";

$port = 3306; // Default MySQL port

$conn = mysqli\_connect($servername, $username, $password, "", $port);

if (!$conn) {

die("Connection failed: " . mysqli\_connect\_error());

}

Create a Database:

Once you have a connection, you can use SQL queries to create a database. Use the mysqli\_query() function to execute the SQL query.

$sql = "CREATE DATABASE my\_database";

if (mysqli\_query($conn, $sql)) {

echo "Database created successfully";

} else {

echo "Error creating database: " . mysqli\_error($conn);

}

Close Connection:

After the database is created, you can close the connection to the MySQL server.

mysqli\_close($conn);

1. **Write a program for cloning of an object.**

class Person {

public $name;

public $age;

public function \_\_construct($name, $age) {

$this->name = $name;

$this->age = $age;

}

}

// Create an instance of the Person class

$person1 = new Person("John Doe", 30);

// Clone the object

$person2 = clone $person1;

// Modify properties of the cloned object

$person2->name = "Jane Doe";

$person2->age = 25;

// Print information of both objects

echo "Person 1: {$person1->name}, {$person1->age} years old\n";

echo "Person 2: {$person2->name}, {$person2->age} years old\n";

In this example, we have a Person class with name and age properties. We create an instance of this class called $person1. We then clone it using clone $person1, creating a new object called $person2.

After cloning, $person2 is a separate instance with the same properties and values as $person1. We can then modify the properties of $person2 without affecting $person1.

1. **EXPLAIN queries to update and delete data in the database.**

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
  die("Connection failed: " . mysqli\_connect\_error());  
}  
  
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";  
  
if (mysqli\_query($conn, $sql)) {  
  echo "Record updated successfully";  
} else {  
  echo "Error updating record: " . mysqli\_error($conn);  
}  
  
mysqli\_close($conn);  
?>

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
  die("Connection failed: " . mysqli\_connect\_error());  
}  
  
// sql to delete a record  
$sql = "DELETE FROM MyGuests WHERE id=3";  
  
if (mysqli\_query($conn, $sql)) {  
  echo "Record deleted successfully";  
} else {  
  echo "Error deleting record: " . mysqli\_error($conn);  
}  
  
mysqli\_close($conn);  
?>

1. **Write a program to connect PHP with MySQL.**

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password);  
  
// Check connection  
if (!$conn) {  
  die("Connection failed: " . mysqli\_connect\_error());  
}  
echo "Connected successfully";  
?>

1. **Write a PHP program to set and modify cookies.**

<?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>

Modify a Cookie Value

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

<?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>

1. **State AND EXPLAIN the use of serialization**

Serialization is the process of converting a data structure or object into a format that can be easily stored or transmitted, and later reconstructed. This is useful in various scenarios, such as:

**Data Persistence**: Serialization allows you to save the state of an object or data structure to a file or a database. This is commonly used in applications where you want to save user preferences, game progress, or any kind of application state.

**Network Communication**: When you need to send complex data structures or objects over a network (e.g., in a client-server architecture), you can serialize the data on the sender's side, transmit it, and then deserialize it on the receiver's side.

**Caching**: Serialization is used in caching mechanisms to store objects in memory or on disk, so they can be quickly retrieved without the need for expensive operations to recreate them.

**Session Management**: In web applications, session data (e.g., user authentication information) can be serialized and stored on the server to maintain user state across multiple HTTP requests.

**Cloning or Deep Copying**: When you want to create a duplicate of a complex object, you can serialize it and then deserialize it. This process creates a deep copy of the object, meaning that changes to the original object won't affect the copy.

**Cross-platform Communication**: Serialization allows data to be converted to a platform-independent format, making it possible to exchange data between systems with different architectures or programming languages.

**Object Persistence in Databases**: Some databases support serialization of objects, allowing complex data structures or objects to be stored directly in a database field.

1. **State the query to insert data in the database.**

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = mysqli\_connect($servername, $username, $password, $dbname);  
// Check connection  
if (!$conn) {  
  die("Connection failed: " . mysqli\_connect\_error());  
}  
  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
VALUES ('John', 'Doe', 'john@example.com')";  
  
if (mysqli\_query($conn, $sql)) {  
  echo "New record created successfully";  
} else {  
  echo "Error: " . $sql . "<br>" . mysqli\_error($conn);  
}  
  
mysqli\_close($conn);  
?>

1. **How can we destroy cookies?**

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

<?php  
// set the expiration date to one hour ago  
setcookie("user", "", time() - 3600);  
?>  
<html>  
<body>  
  
<?php  
echo "Cookie 'user' is deleted.";  
?>  
</body>  
</html>

1. How to create session variable in PHP?

A session is started with the session\_start() function.Session variables are set with the PHP global variable: $\_SESSION.

<?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>

1. Write difference between get ( ) & post ( ) method of form (Any four points).

Parameters in URL: When you use the GET method, the data is appended to the URL as query parameters. This means that the data is visible in the URL bar of the browser.

Limited Data Size: The amount of data that can be sent using GET is limited by the maximum length of a URL. This varies depending on the browser and server configuration, but it's typically around 2,048 characters.

Bookmarkable and Shareable: Since the data is in the URL, GET requests can be bookmarked, shared, and even cached by the browser.

Idempotent: GET requests are considered "idempotent," meaning they can be repeated multiple times with the same result. They should not have any side effects on the server.

Not Suitable for Sensitive Data: Because the data is visible in the URL, GET requests are not suitable for sensitive information like passwords or credit card numbers.

POST Method:

Parameters in Request Body: In contrast, with the POST method, the data is sent in the body of the HTTP request, rather than in the URL. This means the data is not visible in the URL bar.

No Limit on Data Size: POST requests have no specific limit on the amount of data that can be sent, making them suitable for sending large amounts of data.

Not Bookmarkable or Shareable: Since the data is not in the URL, POST requests are not easily bookmarked, shared, or cached.

Not Idempotent: POST requests are considered "non-idempotent," meaning they can have side effects on the server. For example, submitting a form that creates a new record in a database.

Suitable for Sensitive Data: Because the data is not visible in the URL, POST requests are more suitable for sending sensitive information

1. **Create customer form like customer name, address, mobile no, date of birth using different form of input elements & display user inserted values in new PHP form.**

Create the HTML Form (customer\_form.html):

<!DOCTYPE html>

<HTML>

<head>

<title>Customer Form</title>

</head>

<body>

<h2>Customer Information Form</h2>

<form action="process\_form.php" method="post">

<label for="name">Name:</label>

<input type="text" id="name" name="name"><br><br>

<label for="address">Address:</label>

<input type="text" id="address" name="address"><br><br>

<label for="mobile">Mobile No:</label>

<input type="text" id="mobile" name="mobile"><br><br>

<label for="dob">Date of Birth:</label>

<input type="date" id="dob" name="dob"><br><br>

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

</form>

</body>

</html>

Create the PHP Form to Display User-Inserted Values (process\_form.php):

<!DOCTYPE html>

<html >

<head>

<title>Display Customer Info</title>

</head>

<body>

<h2>Customer Information</h2>

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$name = $\_POST["name"];

$address = $\_POST["address"];

$mobile = $\_POST["mobile"];

$dob = $\_POST["dob"];

echo "Name: $name<br>";

echo "Address: $address<br>";

echo "Mobile No: $mobile<br>";

echo "Date of Birth: $dob";

}

?>

<br><br>

<a href="customer\_form.html">Go Back</a>

</body>

</html>

1. How do you validate user inputs in PHP?

Validating user inputs in PHP is crucial to ensure the security and integrity of your application. Here are some common techniques and best practices for validating user inputs in PHP:

**Use filter\_var() Function:**

PHP provides the filter\_var() function to validate and sanitize various types of data, such as email addresses, URLs, integers, and more. It returns the filtered data or false on failure.

$email = $\_POST['email'];

if (filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

echo "Valid email address.";

} else {

echo "Invalid email address.";

}

**Check Required Fields:**

Ensure that mandatory fields are filled out by checking if they are set and not empty.

if (isset($\_POST['username']) && !empty($\_POST['username'])) {

// Process username

} else {

echo "Username is required.";

}

**Validate Numbers or Numeric Values:**

Use functions like is\_numeric() or ctype\_digit() to verify if a value is a valid number.

$age = $\_POST['age'];

if (is\_numeric($age) && $age > 0) {

echo "Valid age.";

} else {

echo "Invalid age.";

}

**Regular Expressions (Regex):**

Regular expressions can be used to match specific patterns in strings. They're powerful for validating complex data like phone numbers or passwords.

$phone = $\_POST['phone'];

if (preg\_match("/^[0-9]{10}$/", $phone)) {

echo "Valid phone number.";

} else {

echo "Invalid phone number.";

}

**Sanitize User Input:**

Use functions like htmlspecialchars()to sanitize user input before using it in a SQL query to prevent SQL injection attacks.

$input = $\_POST['user\_input'];

$sanitized\_input = htmlspecialchars($input);

**Limit Input Length:**

Set maximum character limits on input fields to prevent overly long entries**.**

$comment = $\_POST['comment'];

if (strlen($comment) <= 200) {

// Process comment

} else {

echo "Comment is too long.";

}

1. **STATE the purpose of $\_PHP\_SELF?**

$\_SERVER['PHP\_SELF'] is a PHP superglobal variable that holds the filename of the currently executing script, relative to the document root. It provides a way to refer to the current script in a self-referencing manner. The main purpose of $\_SERVER['PHP\_SELF'] is to help in processing form submissions or generating dynamic links within a PHP script.

1. **Explain the use of preg\_match () functions in regular expression?**

The preg\_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise. This function is used for performing a regular expression match

<?php  
$str = "Visit W3Schools";  
$pattern = "/w3schools/i";  
echo preg\_match($pattern, $str);  
?>

$name = test\_input($\_POST["name"]);  
if (!preg\_match("/^[a-zA-Z-' ]\*$/",$name)) {  
  $nameErr = "Only letters and white space allowed";  
}

t's used to check if the $name variable matches the specified regular expression pattern

^: Matches the start of a string.

[a-zA-Z-' ]:This is a character class that matches any letter (both uppercase and lowercase), hyphens, apostrophes and spaces.

\*:Matches zero or more occurrences of the preceding pattern.

$:Matches the end of a string.

1. **LIST super global variable IN PHP?**

The PHP superglobal variables are:

1. $GLOBALS
2. $\_SERVER
3. $\_REQUEST
4. $\_POST
5. $\_GET
6. $\_FILES
7. $\_ENV
8. $\_COOKIE
9. $\_SESSION
10. **Why we use $\_REQUEST variable?**

The $\_REQUEST variable in PHP is a superglobal array that combines the content of $\_GET, $\_POST, and $\_COOKIE arrays into a single array. It allows you to access data sent via both HTTP GET and POST methods, as well as cookies, all in one place.

1. **Write the PHP code for fetching the data from a database to a webpage?**

?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect\_error) {  
  die("Connection failed: " . $conn->connect\_error);  
}  
  
$sql = "SELECT id, firstname, lastname FROM MyGuests";  
$result = $conn->query($sql);  
  
if ($result->num\_rows > 0) {  
  // output data of each row  
  while($row = $result->fetch\_assoc()) {  
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";  
  }  
} else {  
  echo "0 results";  
}  
$conn->close();  
?>

1. **Differentiate between Session and Cookies**

Cookies are client-side files on a local computer that hold user information.

Sessions are server-side files that contain user data.

Cookies end on the lifetime set by the user.

When the user quits the browser or logs out of the programmed, the session is over.

It can only store a certain amount of info.

It can hold an indefinite quantity of data.

The browser’s cookies have a maximum capacity of 4 KB.

We can keep as much data as we like within a session, however there is a maximum memory restriction of 128 MB that a script may consume at one time.

Because cookies are kept on the local computer, we don’t need to run a function to start them.

To begin the session, we must use the session start() method.

Cookies are not secured.

Session are more secured compare than cookies.

Cookies stored data in text file.

Session save data in encrypted form.

Cookies stored on a limited data.

Session stored a unlimited data.

In PHP, to get the data from Cookies , $\_COOKIES the global variable is used

In PHP , to get the data from Session, $\_SESSION the global variable is used

We can set an expiration date to delete the cookie’s data. It will automatically delete the data at that specific time.

In PHP, to destroy or remove the data stored within a session, we can use the session\_destroy() function, and to unset a specific variable, we can use the unset() function.

1. **Elaborate the mysqli\_connect().**

The mysqli\_connect() function in PHP is used to establish a connection to a MySQL database. It is part of the MySQLi (MySQL Improved) extension, which provides an improved set of functions for interacting with MySQL databases compared to the older MySQL extension.

Syntax:

mysqli\_connect(server, username, password, dbname, port, socket);

Parameters:

server: Specifies the MySQL server host or IP address. It can be a string representing either a hostname (e.g., "localhost") or an IP address (e.g., "127.0.0.1").

username: The MySQL database username used to connect to the database server.

password: The password associated with the username.

dbname (optional): The name of the database you want to connect to. If provided, the connection will automatically select this database.

port (optional): The port number to use when connecting to the MySQL server. The default is 3306.

socket (optional): The socket or named pipe used for local MySQL connections. This parameter is typically used on Unix-like systems.

Return Value:

If the connection is successful, mysqli\_connect() returns a MySQLi object that represents the connection. If it fails, it returns false.

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "my\_database";

$conn = mysqli\_connect($servername, $username, $password, $dbname);

if (!$conn) {

die("Connection failed: " . mysqli\_connect\_error());

} else {

echo "Connected successfully";

}

1. **Explain web page validation with example.(U CAN WRITE CODE FOR ANY TWO CONTROLS )**

PHP - Validate Name

The preg\_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise. This function is used for performing a regular expression match

<?php  
$str = "Visit W3Schools";  
$pattern = "/w3schools/i";  
echo preg\_match($pattern, $str);  
?>

$name = test\_input($\_POST["name"]);  
if (!preg\_match("/^[a-zA-Z-' ]\*$/",$name)) {  
  $nameErr = "Only letters and white space allowed";  
}

t's used to check if the $name variable matches the specified regular expression pattern

^: Matches the start of a string.

[a-zA-Z-' ]:This is a character class that matches any letter (both uppercase and lowercase), hyphens, apostrophes and spaces.

\*:Matches zero or more occurrences of the preceding pattern.

$:Matches the end of a string.

The easiest and safest way to check whether an email address is well-formed is to use PHP's filter\_var() function.

filter\_var() is a function in PHP used to filter a variable with a specified filter

FILTER\_VALIDATE\_EMAIL is a predefined constant specifically designed to validate email addresses. It only verifies the syntax of the email address.

$email = test\_input($\_POST["email"]);  
if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {  
  $emailErr = "Invalid email format";  
}

PHP - Validate URL

$website = test\_input($\_POST["website"]);  
if (!preg\_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~\_|!:,.;]\*[-a-z0-9+&@#\/%=~\_|]/i",$website)) {  
  $websiteErr = "Invalid URL";  
}

/\b:This is a word boundary anchor. It matches the position between a word character (as \w) and a non-word character. It ensures that the URL is a separate word or surrounded by non-word characters.

(?:(?:https?|ftp):\/\/|www\.) :This part of the pattern matches the protocol part of the URL. It looks for either http,htttps,ftp followed by :// OR it looks for www. The ?:is used for non-capturing groups, meaning that the matched text won't be captured as a separate group.

[-a-z0-9+&@#\/%?=~\_|!:,.;]\* :This part matches the path and query string of the URL. It allows for various characters that are commonly used in URLs.

[-a-z0-9+&@#\/%=~\_|] :This part matches the last character in the URL (e.g., a filename or an anchor).

/I :flag makes the pattern case-insensitive, so it will match both upper and lower case letters

<html>

<head>

<style>

.error {color: #FF0000;}

</style>

</head>

<body>

<?php

// define variables and set to empty values

$nameErr = $emailErr = $genderErr = $websiteErr = "";

$name = $email = $gender = $comment = $website = "";

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

if (empty($\_POST["name"])) {

$nameErr = "Name is required";

} else {

$name = test\_input($\_POST["name"]);

// check if name only contains letters and whitespace

if (!preg\_match("/^[a-zA-Z-' ]\*$/",$name)) {

$nameErr = "Only letters and white space allowed";

}

}

if (empty($\_POST["email"])) {

$emailErr = "Email is required";

} else {

$email = test\_input($\_POST["email"]);

// check if e-mail address is well-formed

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$emailErr = "Invalid email format";

}

}

if (empty($\_POST["website"])) {

$website = "";

} else {

$website = test\_input($\_POST["website"]);

// check if URL address syntax is valid (this regular expression also allows dashes in the URL)

if (!preg\_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~\_|!:,.;]\*[-a-z0-9+&@#\/%=~\_|]/i",$website)) {

$websiteErr = "Invalid URL";

}

}

if (empty($\_POST["comment"])) {

$comment = "";

} else {

$comment = test\_input($\_POST["comment"]);

}

if (empty($\_POST["gender"])) {

$genderErr = "Gender is required";

} else {

$gender = test\_input($\_POST["gender"]);

}

} function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

?>

<h2>PHP Form Validation Example</h2>  
<p><span class="error">\* required field</span></p>  
<form method="post" 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>

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

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

<br><br>

Website: <input type="text" name="website" value="<?php echo $website;?>">

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

<br><br>

Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?></textarea>

<br><br>

Gender:

<input type="radio" name="gender" <?php if (isset($gender) && $gender=="female") echo "checked";?> value="female">Female

<input type="radio" name="gender" <?php if (isset($gender) && $gender=="male") echo "checked";?> value="male">Male

<input type="radio" name="gender" <?php if (isset($gender) && $gender=="other") echo "checked";?> value="other">Other

<span class="error">\* <?php echo $genderErr;?></span>

<br><br>

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

</form>

<?php

echo "<h2>Your Input:</h2>";

echo $name;

echo "<br>";

echo $email;

echo "<br>";

echo $website;

echo "<br>";

echo $comment;

echo "<br>";

echo $gender;

?>

1. **Define the uses of serialize() and unserialize().**
2. **State the use of cloning of an object.**

Cloning can be useful when you want to create a copy of an object to work with, without affecting the original object.

1. **List out the database operations.**