# Section 1: Theoretical Questions

## PHP Basics

* + 1. **Difference between == and ===:**
       - Explain the difference between == and === in PHP with examples.

**Answer**

**==** checks if values are equal, ignoring type.   
 e.g. : "true" **==** true is true

"5" == 5 is true

**===** checks for both value and type equality.

e.g. : "true" **===** true is false

true **===** true is true

* + 1. **Purpose of isset() vs empty():**
       - Describe the use cases and differences between isset() and empty().

Answer

isset() checks if a variable exists and is not null.

empty() checks if a variable is empty (e.g., 0, "", null, etc.).

## Sessions vs Cookies:

* + - * Explain the purpose of PHP sessions and cookies and highlight their differences.

|  |  |
| --- | --- |
| Sessions | Cookies |
| Server-side | Client-side (browser) |
| Can store large amounts of sensitive or temporary information data | store small amounts of data on the client’s browser. |
| More secure, harder to tamper with | Less secure, accessible to client |

Answer

## Object-Oriented Programming (OOP)

1. **Abstract Class vs Interface:**
   * + - Describe the differences between an abstract class and an interface in PHP. Provide a scenario where each would be used.

Answer

**Abstract classes** are useful when creating a base class with some shared

implementation and requiring subclasses to complete specific functionality.

**Interfaces** are useful when ensuring that multiple unrelated classes follow the

same structure, often to support a common functionality across different

contexts.

## Inheritance:

* + - * Explain how inheritance works in PHP and its benefits for code reuse.

Answer

Reduce redundancy, simplifying maintenance, and keeping code organized and efficient.

## Security

1. **Preventing SQL Injection:**
   * + - Explain how to prevent SQL injection in PHP and provide an example using prepared statements.

Answer

Using prepared statements. This mechanism will separate SQL code from data, any injected SQL commands are not executed as part of the query. This method effectively prevents SQL injection vulnerabilities by ensuring that input data cannot alter the structure of the SQL query.

## Handling File Uploads Securely:

* + Describe the steps to handle file uploads securely in PHP.

Answer

- Validate file type.

- Limit file size.

- Generate unique filenames.

- Use a secure upload directory.

- Set proper directory permissions.

- Enable virus scanning (if possible).

- Avoid displaying detailed errors to users.

- Validate file content when necessary.

## Basic Coding

* + 1. **Reverse String Function:**
       - Write a PHP function reverseString($str) that reverses a string.

Answer

function reverseString($str) {

return strrev($str);

}

## Fibonacci Function:

* + - * Implement a PHP function fibonacci($n) to return the nth Fibonacci number using recursion.

Answer

<?php

function fibonacci($n) {

if ($n == 0) {

return 0;

} elseif ($n == 1) {

return 1;

}

return fibonacci($n - 1) + fibonacci($n - 2);

}

echo fibonacci(5);

?>

## Intermediate Coding

1. **Person Class:**
   * Create a PHP class Person with properties name, age, and email. Include a method displayInfo() to return a string with the person’s information.

Ensure email validation.

Answer

<?php

class Person {

public $name;

public $age;

public $email;

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

$this->name = $name;

$this->age = $age;

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

$this->email = $email;

} else {

throw new Exception("Invalid email format");

}

}

function displayInfo() {

return "Name: $this->name, Age: $this->age, Email: $this->email";

}

}

?>

## Read JSON File:

* + Write a PHP script to read a JSON file containing an array of users and display each user's name and email.

Answer

<?php

$json = file\_get\_contents(dummy\_data.json');

$users = json\_decode($json, true);

foreach ($users as $user) {

echo "Name: {$user['name']}, Email: {$user['email']}<br/>";

}

?>

# Section 3: Practical Task

## Build a Simple Contact Form

* + 1. **Task:**
       - Create a PHP script for a contact form with fields for name, email, and message. When submitted, validate the inputs and send an email using PHP’s mail() function. Implement basic validation to ensure the email is correctly formatted and all fields are filled out.

Answer

<?php

$respMsg = '';

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

$name = trim($\_POST['name']);

$email = trim($\_POST['email']);

$message = trim($\_POST['message']);

if (empty($name) || empty($email) || empty($message)) {

$respMsg = "All fields are required.";

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

$respMsg = "Please enter a valid email address.";

} else {

$to = "Zarifkim@yahoo.com";

$subject = "New Contact Form Submission";

$emailContent = "Name: $name\n";

$emailContent .= "Email: $email\n";

$emailContent .= "Message: $message\n";

$headers = "From: $email";

if (mail($to, $subject, $emailContent, $headers)) {

$respMsg = "Your message has been sent successfully!";

} else {

$respMsg = "There was an error sending your message. Please try again later.";

}

}

echo $respMsg;

}

?>

## Debugging

* + 1. **Fix Date and Time Display:**

o Given the following PHP code that displays the current date and time, ensure it works correctly:

<?php

echo "The current date and time is: " . date("Y-m-d H:i:s");

?>

Answer

<?php

date\_default\_timezone\_set("Asia/Kuala\_Lumpur");

echo "The current date and time is: " . date("Y-m-d H:i:s");

?>

## Optimization

* + 1. **Optimize Data Processing:**
       - Analyze and optimize the following PHP code snippet for better performance:

<?php

function getLargeDataSet() {

$data = array();

for ($i = 0; $i < 10000; $i++) {

$data[] = $i;

}

return $data;

}

function processData($data) { foreach ($data as $value) {

echo $value . "<br/>";

}

}

$data = getLargeDataSet(); processData($data);

?>

Answer

<?php

function getLargeDataSet() {

return range(0, 9999);

}

function processData($data) {

$output = '';

foreach ($data as $value) {

$output .= $value . "<br/>";

}

echo $output;

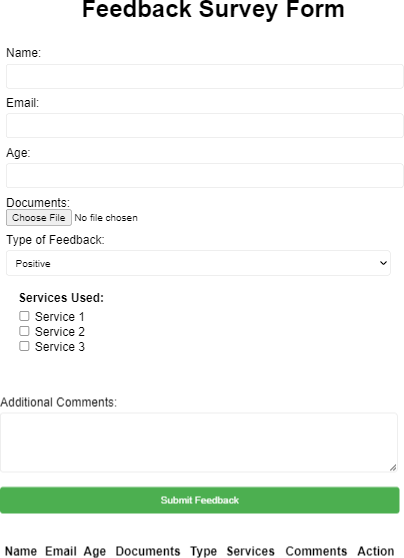
}

$data = getLargeDataSet();

processData($data);

?>

1. Creating a simple survey form.
2. Include CRUD operation – add, update and delete function.
3. Database – Postgres / MySQL
4. Language – PHP



# Section 6: Submission Requirement

1. Create a public or private GitHub repository for this project.
2. Commit all code, scripts and documentation to the repository.
3. Share the repository link as part of your submission.