# Everlytic Developer Assessment

|  |  |
| --- | --- |
| Name | Ruan van Zyl |

|  |  |
| --- | --- |
| Date | 27 May 2023 |

Please complete the answers to the questions below. The assessment should take roughly 30 minutes.

### What is the difference between public, protected and private in a class definition?

|  |
| --- |
| Public: When a member is declared as public, it is accessible from anywhere within the program. |
| Protected: The protected access modifier allows the member to be accessed within the class itself and its subclasses. |
| Private: Private members are accessible only within the class where they are defined. |

### Given this code: function doSomething(&$foo) { $bar = $foo; $foo += 1; return $foo; } $value = 3; $result = doSomething($value); echo "value: $value, result: $result"; What will be output to screen and why?

|  |
| --- |
| The given code defines a function doSomething that takes a reference to a variable $foo. |
| Inside the function, $foo is incremented by 1, and the modified value is returned. The code |
| then calls this function with $value set to 3. As $foo is passed by reference, both $value and |
| the returned value become 4. The final output to the screen is "value: 4, result: 4". |

### What is wrong with this query: "SELECT \* FROM table WHERE id = $\_POST[ 'id' ]"?

|  |
| --- |
| The issue arises from directly including user-supplied input ($\_POST['id']) into the query |
| string without proper sanitization or parameterization. This allows an attacker to modify |
| the query's structure or inject malicious SQL code. |

### What is the cause of this warning: 'Warning: Cannot modify header information - headers already sent', and what is a good practice to prevent it?

|  |
| --- |
| The warning occurs when there is an attempt to modify HTTP headers after some content |
| has already been sent to the Browser. |

### What is wrong with this code: class Foo { protected $bar; public function \_\_construct() { $this->bar = 1; } public static function doSomething() { return $this->bar; } }

|  |
| --- |
| The code has an error in the doSomething() method of the Foo class. Since doSomething() is |
| a static method, it cannot access non-static properties using $this. |
|  |

### Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".

<?php

for ($num = 1; $num <= 100; $num++) {

if ($num % 3 == 0 && $num % 5 == 0) {

echo "FizzBuzz" . PHP\_EOL;

} elseif ($num % 3 == 0) {

echo "Fizz" . PHP\_EOL;

} elseif ($num % 5 == 0) {

echo "Buzz" . PHP\_EOL;

} else {

echo $num . PHP\_EOL;

}

}

?>

### **What does the following code do? Explain what’s going on there.**

$date = '08/26/2003';

print preg\_replace('/([0-9]+)\/([0-9]+)\/([0-9]+)/'‚ '$2/$1/$3', $date);  
The given code uses regular expression pattern matching and replacement to

|  |
| --- |
| modify a date string from the format 'MM/DD/YYYY' to 'DD/MM/YYYY'. The modified date |
| is then printed using the print statement. |

### **Given a line of text $string, how would you write a regular expression to strip all the HTML tags from it?**

### To strip HTML tags from a line of text $string, a regular expression can be used with the

|  |
| --- |
| preg\_replace() function in PHP. |
| $strippedString = preg\_replace('/<[^>]\*>/', '', $string); |
|  |

### A palindrome is a word that reads the same backward or forward. Write a function that checks is a given word is a palindrome. Characters case should be ignored. EG. Deleveled is a palindrome and should return true as character case is ignored.

### <?php class Palindrome

### {

### public static function isPalindrome($word)

{

$word = strtolower($word);

$word = preg\_replace('/[^a-z]/', '', $word);

$reversed = strrev($word);

return $word === $reversed;

}

### } echo Palindrome::isPalindrome('Deleveled');

### Considering message text stores a combination of html and text. What security issue is prevalent in the code below and how would you fix it? <?php $messageStmt = $db->query('select message\_text from messages where message\_id = 1'); $messageStmt->execute(); $message = $messageStmt->fetch(PDO::FETCH\_OBJ); ?> <div><?php echo $message->message\_text; ?></div> The code provided is vulnerable to Cross-Site Scripting (XSS) attacks

|  |
| --- |
| To fix it I would change the div to the following: |
| <div><?php echo htmlspecialchars($message->message\_text, ENT\_QUOTES, 'UTF-8'); ?></div> |

### Write an inner join for the following tables

|  |
| --- |
| SELECT User.UserKey, User.FirstName, User.LastName, User.Email, Address.Address1, Address.Address2, Address.City, Address.State, Address.ZipCode  FROM User  INNER JOIN Address ON User.UserKey = Address.UsrKey; |

### Complete the JS function below that validates the conditions of a password: 1. The password must be greater then 7 characters 2. The first character must be a capital letter 3. The password must contain at least one number

function isPasswordValid(password) {

var isValid = false;

if (

password.length > 7 &&

/[A-Z]/.test(password[0]) &&

/\d/.test(password)

) {

isValid = true;

}

return isValid;

}