WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Tejeswar Reddy Nalijeni

Email: nalijety@ucmail.uc.edu

Short-bio: Currently I am pursuing Master's in Information Technology and I am interested in Cybersecurity

Lab 2 - Front-end Web Development

The lab's overview

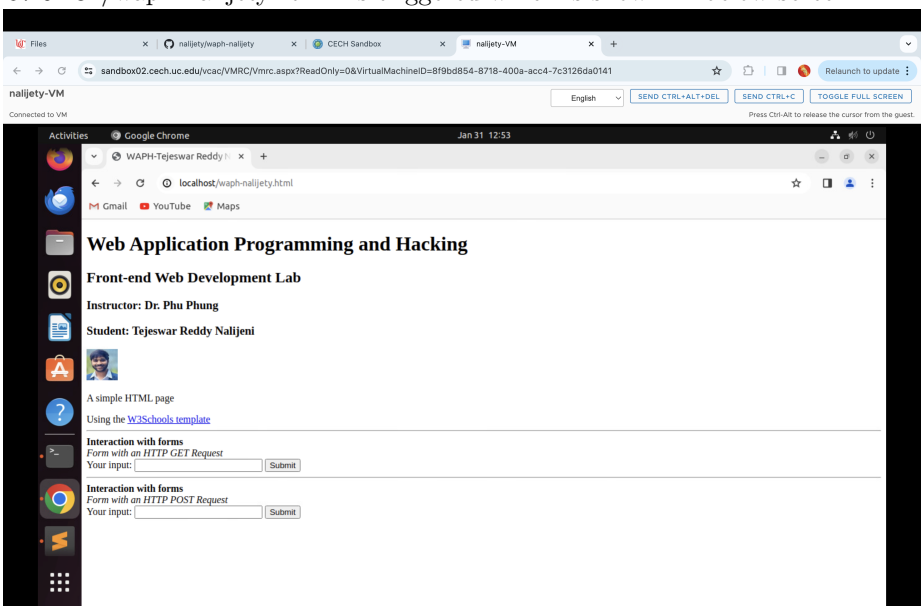
In this lab2, I created an HTML webpage named "waph-nalijety.html," featuring various elements and functionalities. I implemented two input forms for HTTP GET and POST requests, included a headshot image, and added inline JavaScript to display the current date and log keypress events. The page also incorporates a digit clock and an analog clock with JavaScript. For Ajax integration, I created forms with buttons triggering Ajax GET and POST requests to "/echo.php," capturing and displaying server responses dynamically. CSS styles were applied to enhance the webpage's appearance, and jQuery was integrated for simplified Ajax operations. Additionally, I utilized Web APIs, fetching a programming joke from Joke API and predicting age using the Agify API. The webpage URL - "192.167.0.181/waph-nalijety.html" provides an interactive and visually appealing experience, showcasing elements like clocks, forms, and API responses.

https://github.com/nalijety/waph-nalijety/tree/main/labs/lab2.

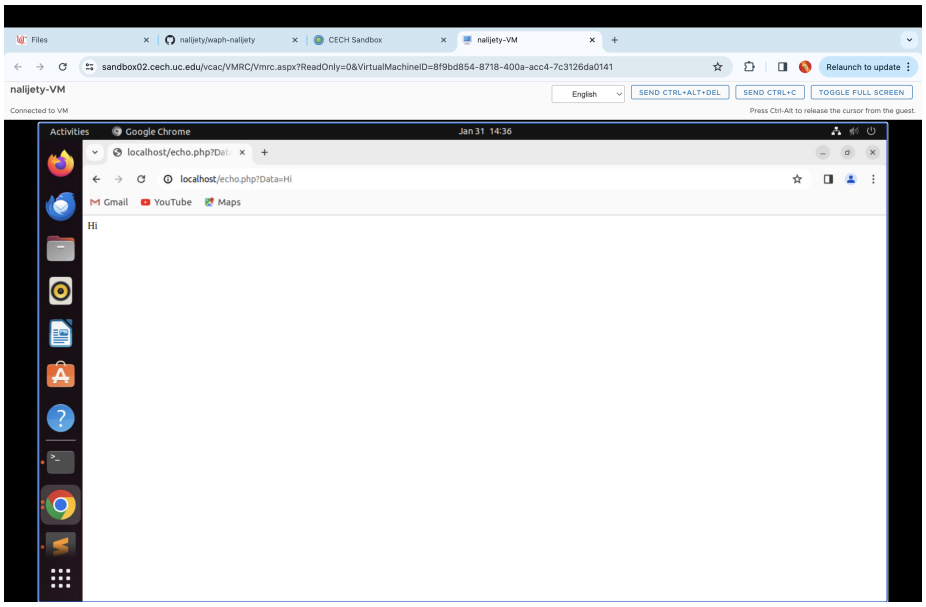
Task 1: Basic HTML with forms, and JavaScript

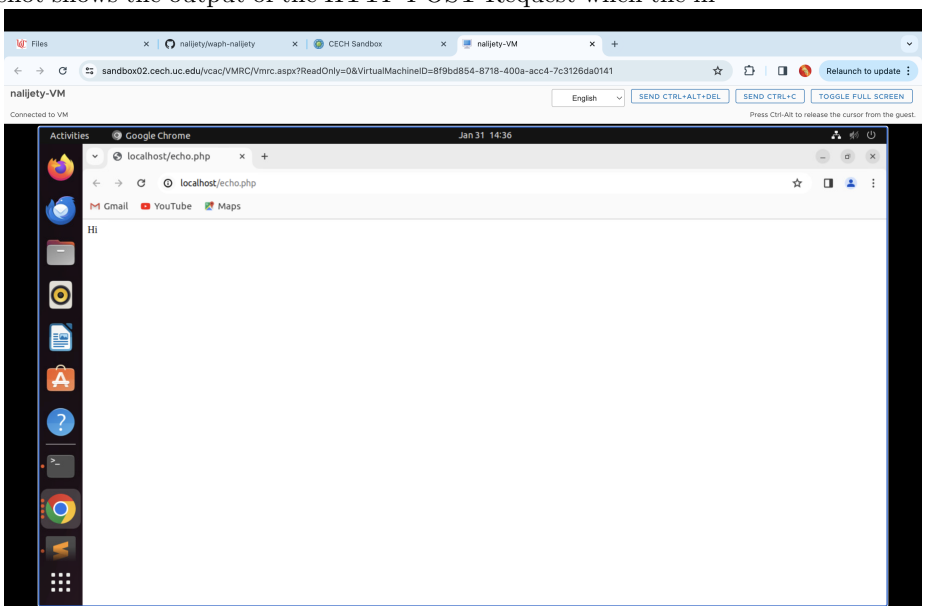
a. HTML In this, I have created the HTML file and named "waph-nalijety.html". Then written the html code for creating two input interation with forms and included the image headshot. The first input is form with the an HTTP GET Request and the second input is form with the an HTTP POST Request. The provided code creates two forms in HTML. Each form has a text input field where users can type something. Both forms have a "Submit" button that users can click to send the form data to a server script located at "/echo.php."

Then given the submit buttons for both of the inputs. After that the html page "192.167.0.181/waph-nalijety.html" is triggered which is shown in below screen-shot.



The below screenshot shows the output of the HTTP GET Request when the in-

put is submitted. 

The below screenshot shows the output of the HTTP POST Request when the input is submitted. 

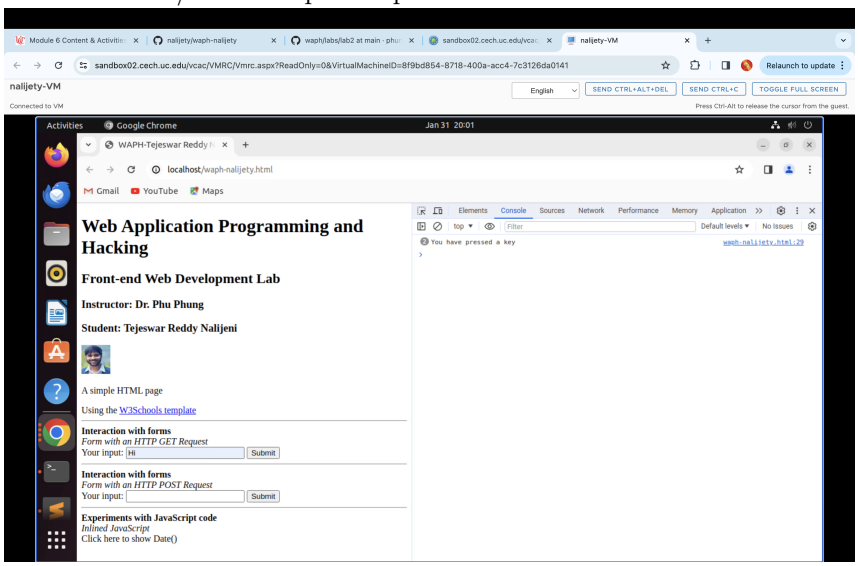
b. Simple JavaScript Write the JavaScript code in your HTML page:

- In this, I have given the inline JavaScript code in html which displays the current date and time. I have written the code which adds a line in the HTML page and introduces a section for JavaScript experiments. Within this section I have provided a clickable text. When clicked on that, it shows

the current date and time. This is done using inline JavaScript, where the code directly changes the content of a specific part of the webpage without reloading the page.

```
<hr>
<b>Experiments with JavaScript code</b><br>
<i>Inlined JavaScript</i>
<div id="date" onclick="document.getElementById('date').innerHTML= Date()">Click here to sho
</div>
```

- After that given a code to log when a key is pressed. In console of the inspect, we will get a "You have pressed a key" when we give input to either GET/POST request inputs which shown in below screenshot.



- I have written a code using simple tags

  that creates digit-clock on webpage. It uses JavaScript to constantly update the current time. The time is refreshed providing a dynamic display of the current time on the webpage.

```
function displayTime() {
    document.getElementById('digit-clock').innerHTML = "Current time: " + new Date();
}
setInterval(displayTime, 500);
```

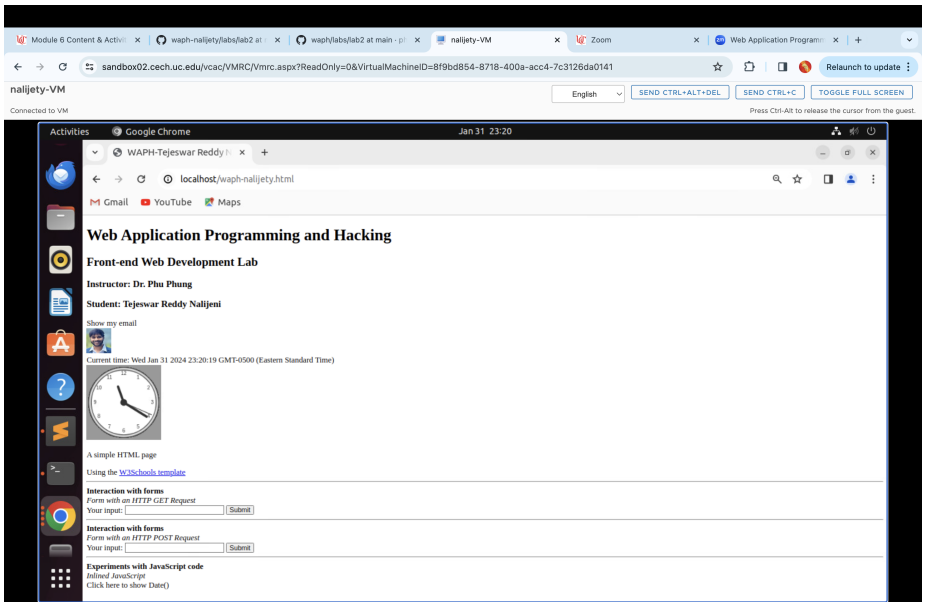
- In this I have given a code which adds a new section "email" to my webpage. Within this section, I'm implementing a specific task using the "showhideEmail()" function. To enable the functionality of this function, I'm bringing in an external JavaScript file named "email.js" and linking it to my webpage. To ensure seamless integration, I've placed both my HTML file and the "email.js" file in the root directory of my web server.

This ensures that my webpage can have features provided by the external JavaScript file and whenever I click on “show my email”, it will reflect my mail id.

```
<div id="email" onclick="showhideEmail()">Show my email</div>
```

- In this I have added a new feature an analog clock through the addition of a "
  " element with the identifier “analog-clock”. Now by using JavaScript, I have retrieved this canvas element, set up the drawing context, and define the clock’s size based on a specified radius. I have implemented a timer to refresh the clock’s display every second which is done by invoking the “drawClock” function. This function, in turn, calls other functions such as “drawFace”, “drawNumbers”, and “drawTime” to handle different aspects of rendering the clock’s face, numerical indicators, and current time. So, this code introduces a dynamic analog clock to my webpage “192.167.0.181/waph-nalijety.html” which gives an interactive and visual element. The below attached screenshot shows the complete web page with all elements related to task 1.

```
<canvas id="analog-clock" width="150" height="150" style="background-color:#999"></canvas>
  <script type="text/javascript" src="https://waph-uc.github.io/clock.js"></script>
<script>
var canvas = document.getElementById("analog-clock");
var ctx = canvas.getContext("2d");
var radius = canvas.height/2;
ctx.translate(radius, radius);
radius = radius*0.90
setInterval(drawClock, 1000);

function drawClock(){
    drawFace(ctx,radius);
    drawNumbers(ctx, radius);
    drawTime(ctx, radius);
}
</script>
```



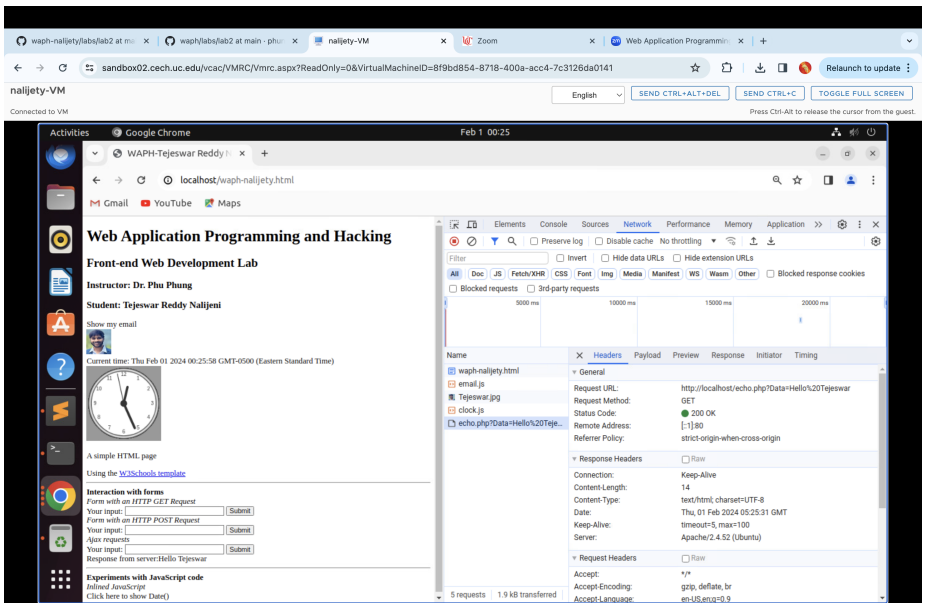
Task 2: Ajax, CSS, jQuery, and Web API integration

a. Ajax

- I've added a input section to my webpage, allowing to type something in a text box "Your input." I have given a code for a "Submit" button that, when clicked, activates a function called "getEcho()". Within this section, users can give input using a text field, and a "Submit" button is provided to trigger an action. The button is associated with a function "getEcho()".
- I've give a code for initiating an Ajax GET request to "echo.php". This function captures input from my webpage and employs an XMLHttpRequest object to get communication with the server. It handles the configuration and sending of the GET request, with the user input as a parameter in the URL.
- The "getEcho()" allows my webpage with the ability to interact with the server, allowing me to seamlessly fetch and display data based on input. The webpage awaits the HTTP response from "echo.php" and displays the result. When I open the inspect and viewed the network, I could see the Request URL, status code, Request method. The below is the screenshot of the inspection of network for Ajax request and response.

```
<script>
function getEcho(){
        var input = document.getElementById("Data").value;
        if      (input.length == 0){
            return;
```

```
        }
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function(){
            if (this.readyState ==4 &&
                this.status == 200){
                console.log("Received data="+xhttp.responseText);
                document.getElementById("response").innerText="Response from server:" + xhtt
                //code to show the data
            }
        }
xhttp.open("GET", "echo.php?Data="+input, true);
//code to create an Ajax request
xhttp.send(); //code to send the request
document.getElementById("Data").value="";
}
</script>
<i>Ajax requests</i><br>
    Your input:
    <input name="Data"
    onkeypress="console.log('You have pressed a key')" id="Data">
    <input class="button round" type="button" value="Ajax Echo" onclick="getEcho()">
```

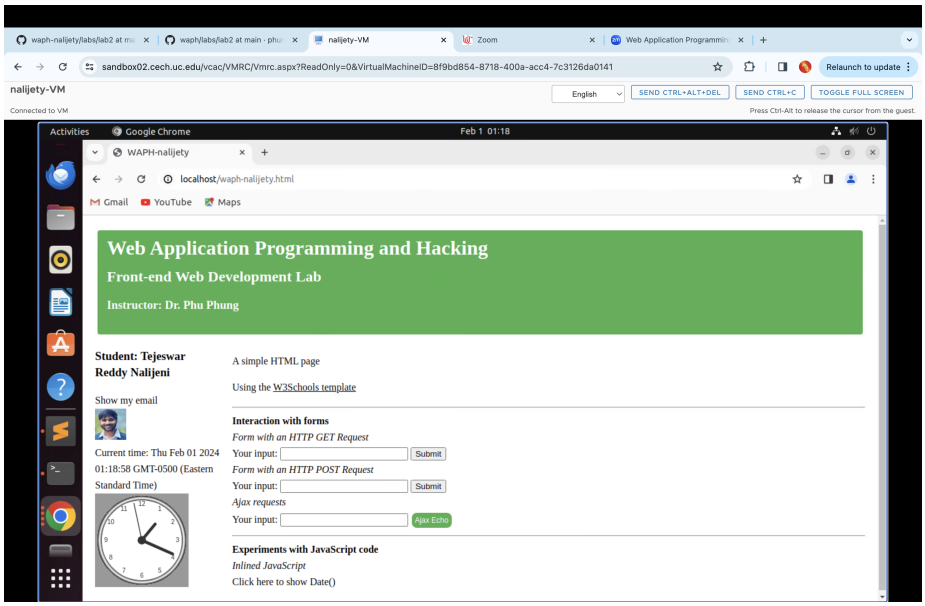


b. CSS Inline CSS: - I have included a specific CSS style to improve the overall appearance. I've enhanced the webpage's structure by giving new containers. I have used a "container wrapper" to establish a well-organized layout, featuring sections like a menu bar, and the main content. These adjustments

aim to improve the overall visual and organizational structure of my webpage.

Internal CSS: - The provided code includes internal CSS styles designed to create a visually appealing button and response. I have given a HTML input element which have these styles to generate a button "Ajax Echo." The functionality is associated with the "getEcho()" JavaScript function, indicating its used for Ajax operations.

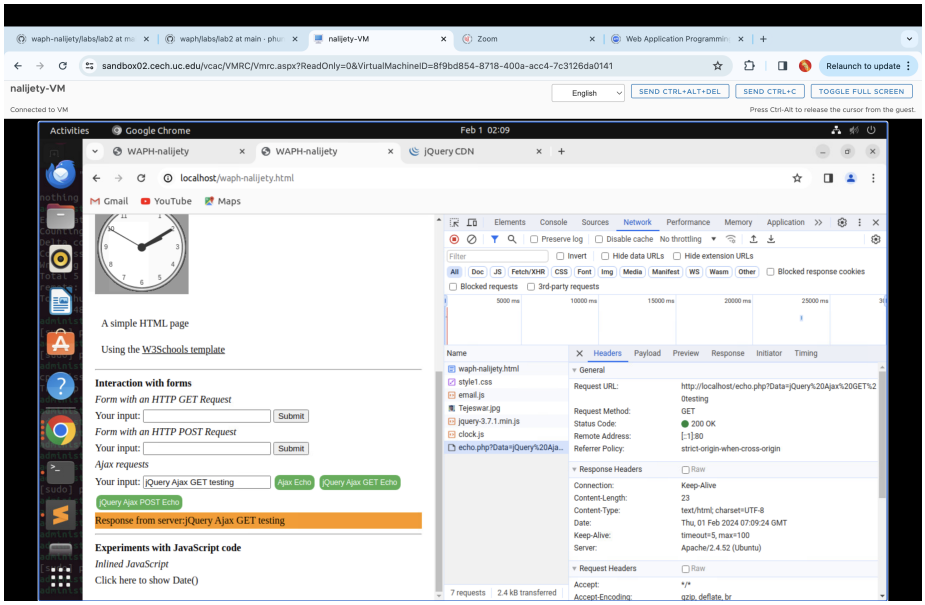
External CSS: - The external CSS file linked in the code hosts styling rules for the chosen visual style. By linking an external CSS file, the styling rules are maintained separately. The styles contribute to the overall look and feel of the webpage, ensuring consistency and providing a visually appealing and cohesive design. The below is the screenshot of overall web page after giving the CSS styles.



c. jQuery I have opened the jQuery website provided and included the CDN in the HTML code. We can either download the library in a .js file and include it locally or we can use a Content Delivery Network (CDN) method.

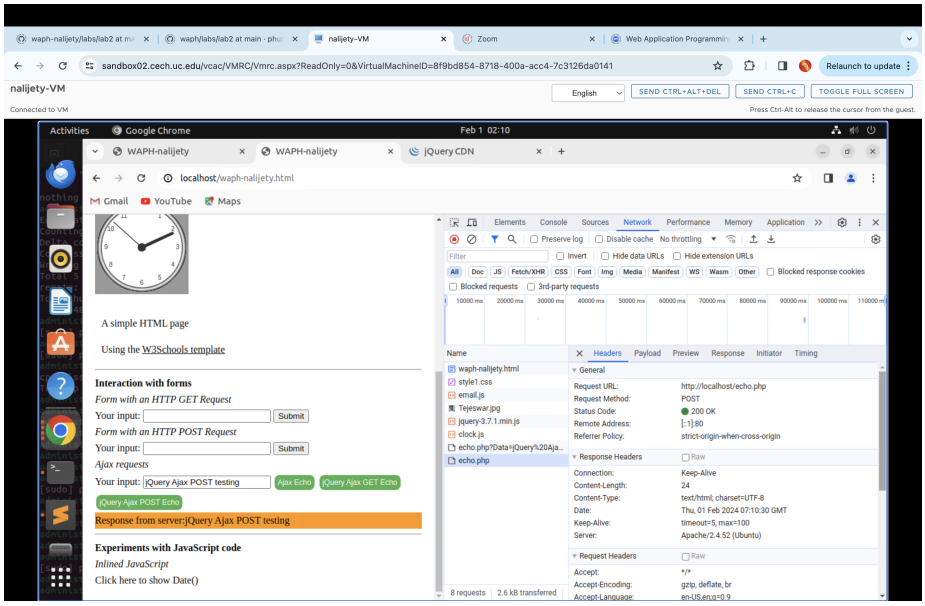
i.jQueryAjax GET:

I have created the jQueryAjax GET Echo button with the function for Ajax GET requests. I have given the input value for testing which is retrieved from an element with the ID "Data" checks and then initiates a GET request to the "echo.php" server with the user input as a query parameter. Upon receiving the server's response, it updates the content of an element with the ID "response" to display the result. I have opened the inspect to view the network where the response is displayed. The below screenshot is for Ajax GET request and response.



ii.jQueryAjax POST:

Similarly, I used a jQueryAjaxPost function to create jQueryAjax POST Echo button for Ajax POST requests. It also works similar to GET like retrieving input, checking for emptiness, and then sending a POST request to the server with the input in the request body. The server's response is processed, updating the content of an element with the ID "response". I have opened the inspect to view the network where the response is displayed. The below screenshot is for Ajax POST request and response.



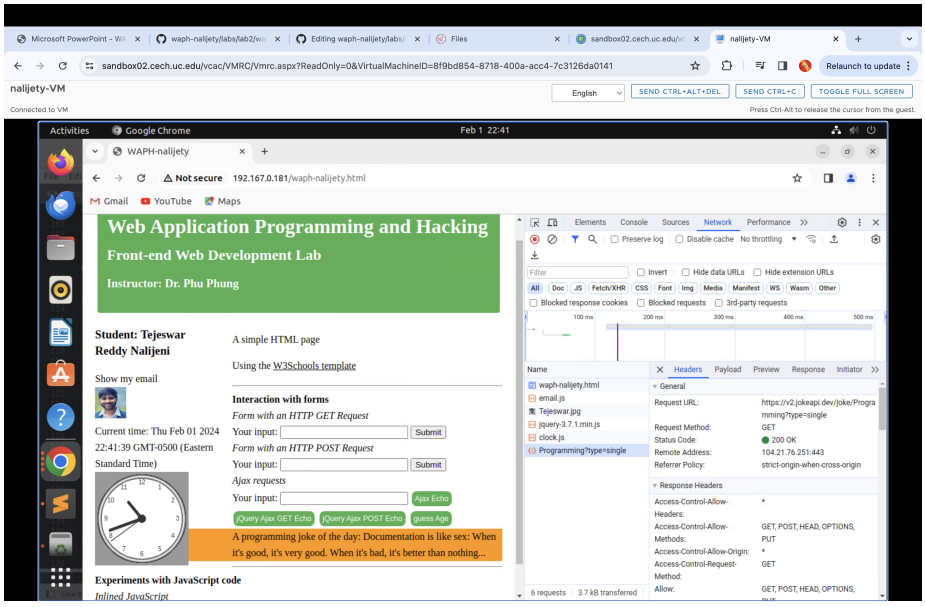

```
<script src="https://code.jquery.com/jquery-3.7.1.min.js" integrity="sha256-/JqT3SQfawRcv/BI
<script>
function jQueryAjax(){
    var input = $("#Data").val();
    if (input.length == 0) return;
    $.get("echo.php?Data="+input, function(result){
        $("#response").html("Response from server:" + result);
    }
);
$("Data").val("");
}

function jQueryAjaxPost(){
    var input = $("#Data").val();
    if (input.length == 0) return;
    $.post("echo.php",{Data: input}, function(result){
        $("#response").html("Response from server:" + result);
    }
);
$("Data").val("");
}
</script>
<input class="button round" type="button" value="jQuery Ajax GET Echo" onclick="jQueryAjax
    <input class="button round" type="button" value="jQuery Ajax POST Echo" onclick="jQueryA
    <div id="response"></div>
```

d. Web API integration

i.Web-API(Joke-generation):

In this JavaScript code, I'm personally using jQuery to fetch a programming-related joke from the "https://v2.jokeapi.dev" API. Through an Ajax GET request to the specific API endpoint, "https://v2.jokeapi.dev/joke/Programming?type=single", I retrieve a single-line programming joke. Once I get a successful retrieval, a callback function is triggered. Inside this function, I log the detailed result to the browser console using "console.log". This allows me to inspect the received data for debugging purposes. Moreover, I dynamically update my webpage by using jQuery to modify the content of an HTML element identified as "response." I display a formatted message, "A programming joke of the day:" followed by the actual programming joke obtained from the API. This script adds a touch of humor and interactivity to my webpage "192.167.0.181/waph-nalijety.html", shows a new programming joke each time the page is loaded. The below is the screenshot which shows the response in the network of the inspect.

```
<script type="text/javascript">
$.get("https://v2.jokeapi.dev/joke/Programming?type=single",
    function(result){
        console.log("From jokeAPI: " + JSON.stringify(result));
        $("#response").html("A programming joke of the day: " + result.joke)
    })
</script>
```



ii.Web-API(Age-display):

I've given a HTML code for button "guess Age". I've set it to invoke a JavaScript function, "guessAge()", passing the value retrieved from an input field with

the ID “Data” using jQuery. The logic for the “guessAge()” function would be defined in JavaScript code. I’ve written a JavaScript function called “guessAge” that uses the Agify API on “https://api.agify.io/?name=input” to predict age. When invoked with a name as an argument, it initiates an asynchronous request to the Agify API using the “fetch” function, retrieves the response, and parses it as JSON. After extracting the predicted age from the data, I construct a message. Finally, using jQuery, I dynamically update the content of an HTML element with the ID “response”, presenting a message that includes the person’s name and the estimated age. This code an engaging and interactive feature to estimate age based on provided names and display the results on a webpage “192.167.0.181/waph-nalijety.html”. The below is the screenshot which shows the response in the network of the inspect. Beacuse of many requests from the users, the output is getting undefined.

```
<script>
async function guessAge(name){
    const response = await fetch("https://api.agify.io/?name="+name);
    const result = await response.json();
    $("#response" ).html("Hi " + name + ", your age should be" + result.age);
}
</script>
<input class="button round" type="button" value="guess Age" onclick="guessAge($('#Data').val
    <div id="response"></div>
```