WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Tulasiram Nakkanaboina

Email: nakkantm@mail.uc.edu

Figure 1: Tulasiram Nakkanaboina

Lab 2 - Front End Web Development

Overview: This lab deals with Front End Development, which gave an overview about basic HTML, Javascript , Ajax, CSS, JQuery library in JS, and web API integration. Part 1 of this lab is to design HTML web page with basic tags and forms. Then Javascript is integrated in 4 ways that is with inline JS, JS with script tag , JS with external file and JS code from a remote repository. This HTML page was then integrated with CSS . Inline CSS, internal CSS and External CSS have been used to make the webpage look elegant. Then Jquery is used to make AJAX get and post calls to echo.php. Lastly 2 web services one is for generating a random joke and the other one is to guess age are integrated into this HTML code using Jquery Ajax and fetch method respectively. Pandoc is used to generate the PDF file from the README.md

Link to the repository: https://github.com/nakkantm-uc/waph-nakkantm/blob/main/labs/lab2/README.md

Part 1 : Basic HTML with forms, and JavaScript

Task 1. HTML

A simple HTML webpage was developed as part of this task which includes basic tags such as `<h1>`,`<h2>`,`<h3>`,`<a>`,`<img>` , `<form>` etc. The file created was named waph-nakkantm.html

Included file `waph-nakkantm.html`:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>WAPH- TULASIRAM NAKKANABOINA</title>
</head>
<body>
<div >
    <div id="top">
        <h1>Web Application Programming and Hacking</h1>
        <h2>Front End Development Lab </h2>
        <h3>Instructor : Dr Phu Phung</h3>
    </div>
    <div >
        <div id="menubar">
        <h3>Student : Tulasiram Nakkanaboina</h3>
        <img src="images/headshot.jpg" alt="Tulasiram headshot" width="50">
        </div>
        <div id="main">
            <p>A Simple HTML Page</p>
            Using the <a href="https://www.w3schools.com/html">W3 Schools Template</a>
            <hr>
            <b>Interaction with HTTP Forms</b>
            <div>
                <i>Form with HTTP GET Request</i>
                <form action="/echo.php" method="GET">
                <lable for="data">Enter the input text</lable>
                <input type="text" name="data"
                onkeyup="console.log('you have clicked a Key')">
                <input type="submit" value="submit">
                </form>
            </div>
            <div>
                <i>Form with HTTP POST Request</i>
                <form action="/echo.php" method="POST">
                <lable for="data">Enter the input text</lable>
                <input type="text" name="data"
```

```
                onkeyup="console.log('you have clicked a Key')">
                <input type="submit" value="submit">
                </form>
            </div>
        </div>
    </div>
</div>
</body>
</html>
```

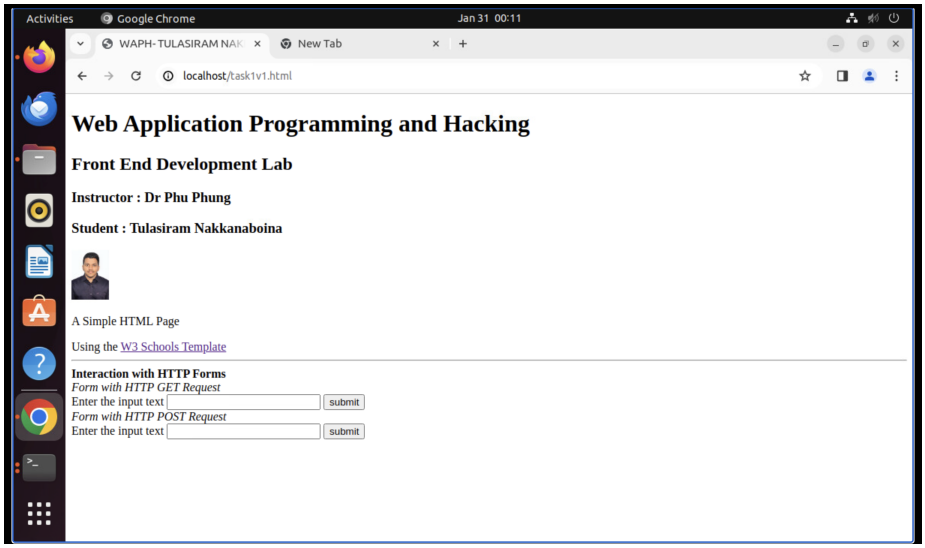

Figure 2: A simple HTML Page

Task 2. Simple JavaScript

This task has given a basic overview of JS syntax and different ways of integrating javaScript code in HTML file.

-Inline JS code was written to display current date and time when clicked ,as well as to log the on click event on the console.

```
<div>
    <b>Experiments with JavaScript code</b><br>
    <i>Inlined JavaScript</i>
    <div id="inlineDate"
        onClick="document.getElementById('inlineDate').innerHTML=Date();
        console.log('you have clicked a Key');">
        Click to display time and date</div>
</div>
```

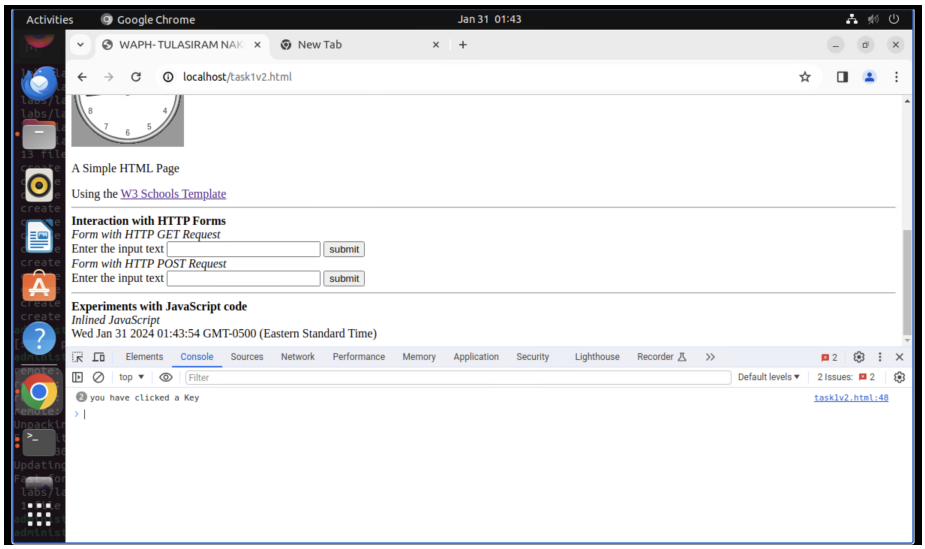

Figure 3: Console screen when clicked

-JavaScript code in a `<script>` tag to display a digital clock.

```
  <script>
        function displayTime() {
            document.getElementById('digital-clock').innerHTML=
                    " The current Time is : "+ Date();
        }
        setInterval(displayTime,500);
</script>
```

-JS code in JS file and and code in HTML page to show or hide email when clicked.

```
    var visible = false;
    function showOrHideEmail(){
     if (visible){
        document.getElementById('email').innerHTML=" Show my Email";
            visible=false;
         }
    else{
        var myEmail="<a href='mailto:nakkantm" +"@"+
                "mail.uc.edu'>nakkantm"+"@"+"mail.uc.edu</a>";
        document.getElementById('email').innerHTML=myEmail;
        visible= true;
        }
    }
```

```
    <div id="email" onclick="showOrHideEmail()">Display my Email</div>
    <script type="text/javascript" src="email.js"></script>
```

-Displaying an Analog clock with an external Javascript code and code in HTML page.

```
<canvas id="analog-clock" width="150" height="150"
         style="background-color:#999"></canvas>
<script src="https://waph-uc.github.io/clock.js"></script>
<script type=text/javascript>
            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>
```

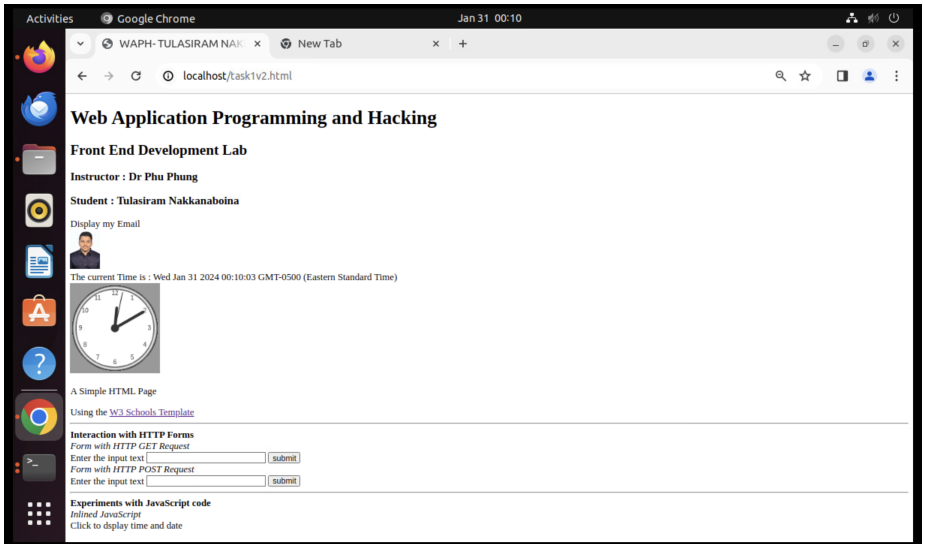

Figure 4: Webpage after adding JavaScript code

Part II - Ajax, CSS, jQuery, and Web API integration

Task 1: Ajax

HTML code is written to take the user input and make a GET call to echo.php using AJAX. The response recieved is then displayed within the div. As it is a GET call the input was sent as a path variable in the URL to the webserver.

```
<div>
    <i>AJAX Requests</i><br>
    <lable for="data">Enter the input text</lable>
    <input type="text" name="data" id="data">
    <input type="submit" value="Ajax Echo" onclick="getEcho()">
    <div id="response"></div>
</div>
<script>
    function getEcho(){
        var input = document.getElementById("data").value;
        if(input.length==0){
        return ;
        }
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function(){
    //alert("readyState "+ this.readyState +", status "+this.status+",
    //statusText= "+this.statusText);
        if(this.readyState==4 && this.status==200){
            console.log("Received data= "+xhttp.responseText);
            document.getElementById("response").innerHTML= xhttp.responseText;
        }
        }
        xhttp.open("GET", "echo.php?data="+input, true);
        xhttp.send();
        document.getElementById("data").value="";
        }
</script>
```

The response for the Ajax call was analyezed in the inspect view. The request method was GET and the status code is 200OK and the input data was passed within the URL.

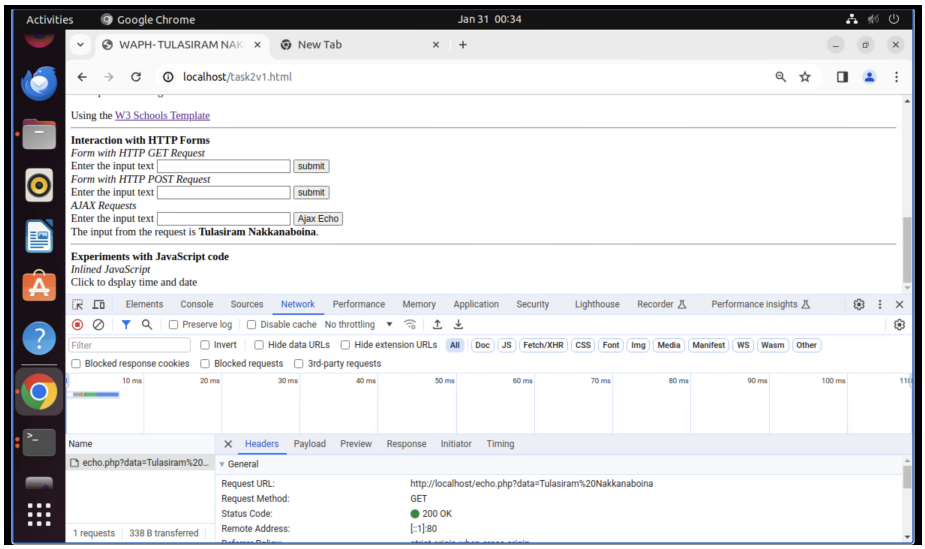

Figure 5: Making an Ajax get call

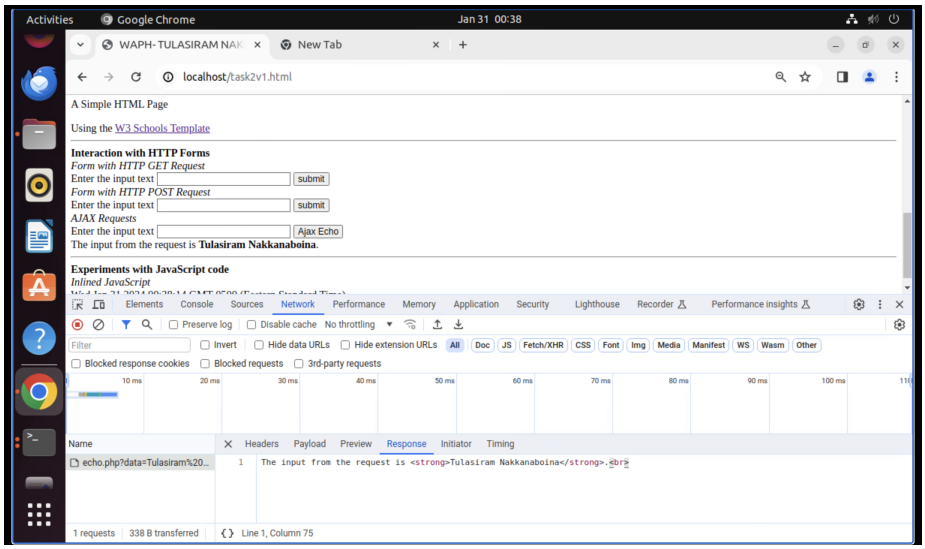

Figure 6: Inspecting the response of Ajax call

Task 2: CSS

a) Inline CSS

```
<body style="background-color: powderblue;">
<h1 style="color: blue;">Web Application Programming and Hacking</h1>
```

Figure 7: modifed webpage after adding inline CSS

b) Internal CSS.

```
<style>
    .button{
        background-color:#4CAF50;
        border:none;
        color:white;
        padding:5px;
        text-align:center;
        text-decoration:none;
        display:inline-block;
        font-size:12px;
        margin:4px2px;
        cursor:pointer;
    }
    .round{
        border-radius:8px;
    }
    #response{
        background-color:#ff9800;
```

```
    }
<!-- HTML code -->
</style>
<input class="button round" type="submit" value="Ajax Echo" onclick="getEcho()">
<input class="button round" type="submit"
    value="JQuery Ajax Echo" onclick="getJqueryAjax()">
<input class="button round" type="submit"
    value="JQuery Ajax Echo Post" onclick="getJqueryAjaxPost()">
<div id="response"></div>
```

c) External CSS from the remote repository provided in the lecture.https://waph-uc.github.io/style1.css.

```
<link rel="stylesheet" type="text/css" href="https://waph-uc.github.io/style1.css">
<!-- HTML code -->
<div class="container wrapper">
<!-- HTML code -->
    <div class="wrapper">
<!-- HTML code -->
    </div>
</div>
```

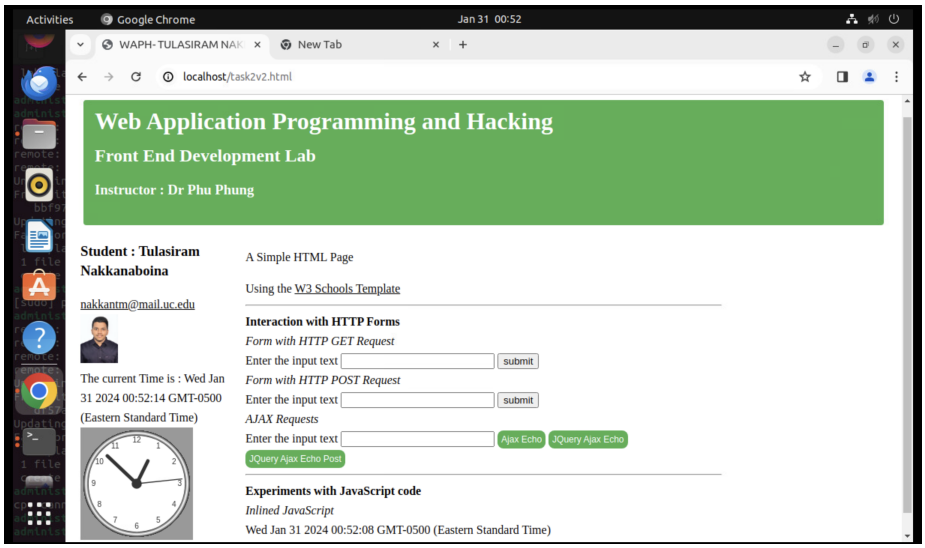

Figure 8: webpage after adding internal and external CSS

Task 3: JQuery

JQuery library has been added to the HTML code. 2 corresponding buttons i.e Jquery Ajax Get and Jquery Ajax Post have been added to make GET and POST calls respectively using Jquery to echo.php. **i.** Ajax GET request to echo.php , the response is analyzed in the inpect view. The call was GET and status code was 200OK.

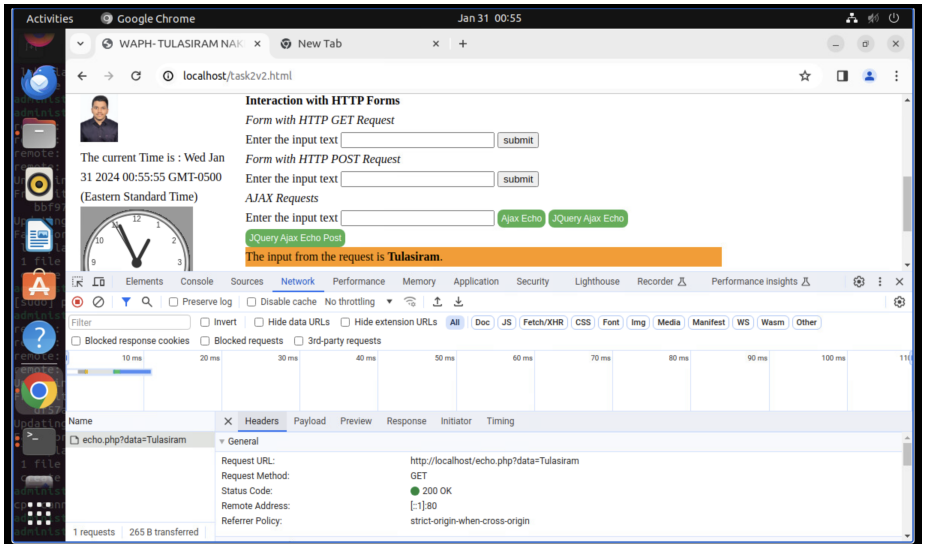

Figure 9: JQuery Ajax GET request to echo.php

```
<!-- HTML code -->
<input class="button round" type="submit" v
    alue="JQuery Ajax Echo" onclick="getJqueryAjax()">
<!-- HTML code -->
<script>
    function getJqueryAjax(){
        var input=$("#data").val();
            if(input.length==0)
                return;
        $.get("echo.php?data="+input,
                  function(result){
                      printResult(result);
                  });
        $("#data").val("");
        }
    function printResult(result){
        $("#response").html(result);
        }
```

```
</script>
```

ii. Ajax POST request to echo.php , the response is analyzed in the inpect view. The call was POST and status code was 200OK.

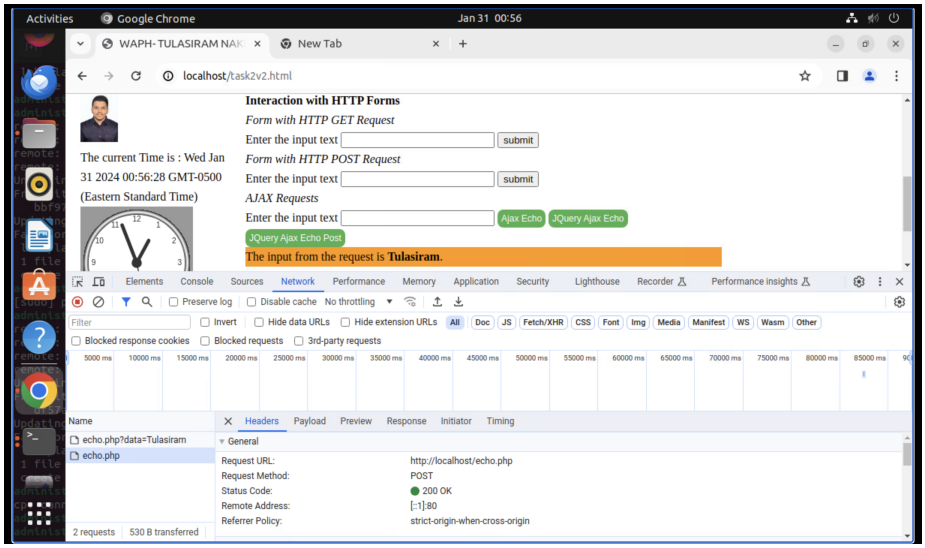

Figure 10: JQuery Ajax POST request to echo.php

```
<!-- HTML code -->
<input class="button round" type="submit"
    value="JQuery Ajax Echo Post" onclick="getJqueryAjaxPost()">
<!-- HTML code -->
<script>
    function getJqueryAjaxPost(){
        var input=$("#data").val();
        if(input.length==0)
            return;
        $.post("echo.php",{data:input},function(result){
                printResult(result);
                });
        $("#data").val("");
        }
    function printResult(result){
        $("#response").html(result);
        }
</script>
```

Task 4: WEB API Integration.

i. Using Ajax on https://v2.jokeapi.dev/joke/Programming?type=single

JavaScript code using JQuery Ajax has been written to make a GET call to the above web service. The response was in JSON , this response was converted to string using JSON.stringify() method and displayed in the console.

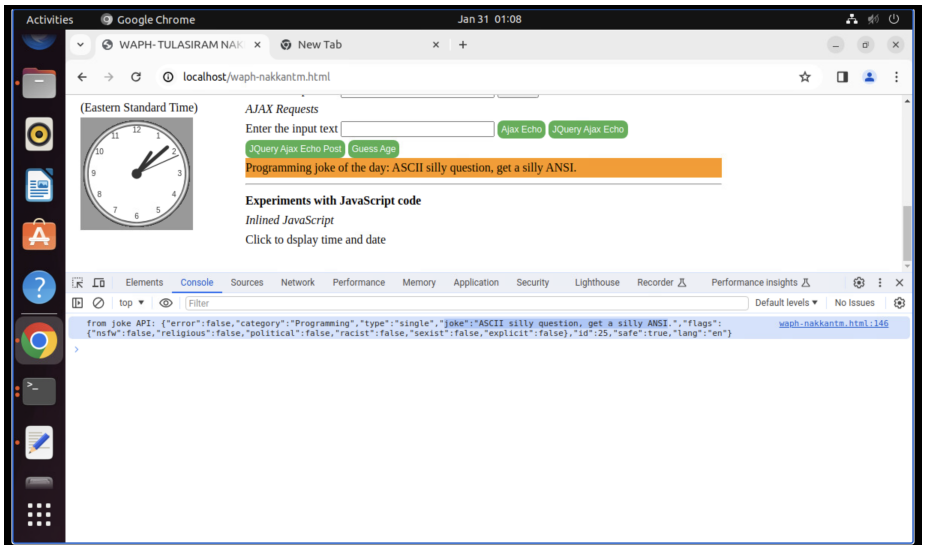

Figure 11: Random Joke displayed when the page is loaded

out of this JSON response, only the joke was filtered using result.joke , this service returns a random joke which is displayed when the webpage is loaded. Refreshing the webpage gives random joke each time.

```
<!-- HTML code -->
<script>
    $.get("https://v2.jokeapi.dev/joke/Programming?type=single",function(result){
        console.log("from joke API: "+ JSON.stringify(result));
    $("#response").html("Programming joke of the day: " +result.joke);
                });
</script>
<!-- HTML code -->
```

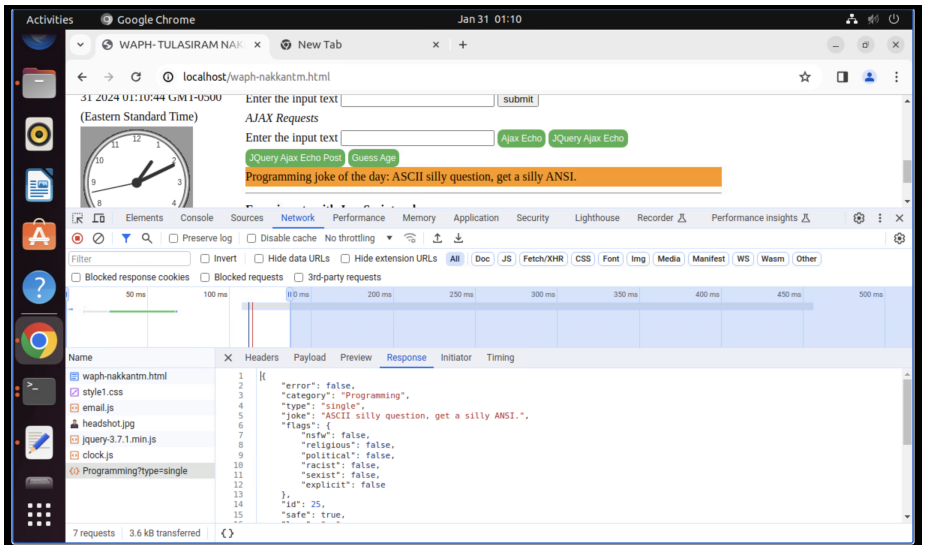

Figure 12: Response of the webservice in inspect view

ii. Using the `fetch` API on https://api.agify.io/?name=input fetch method in Javascript is used to make HTTP request to the above webservice. as it is an asynchronous call the function is defined with the async keyword and the await is used to synchronize the response. The HTTP request made is GET and the status code is 200OK.

```
<script>
async function guessAge(name){
            const response= await fetch("https://api.agify.io/?name="+name);
            const result= await response.json();
            $("#response").html("Hello "+name+" ,your age should be "+result.age);
        }
</script>
```

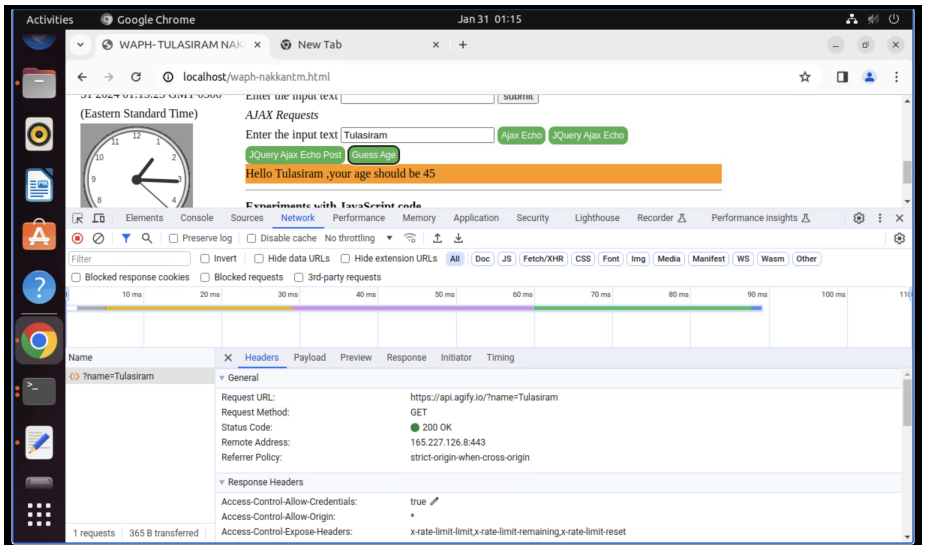

Figure 13: HTTP request to api.agify.io

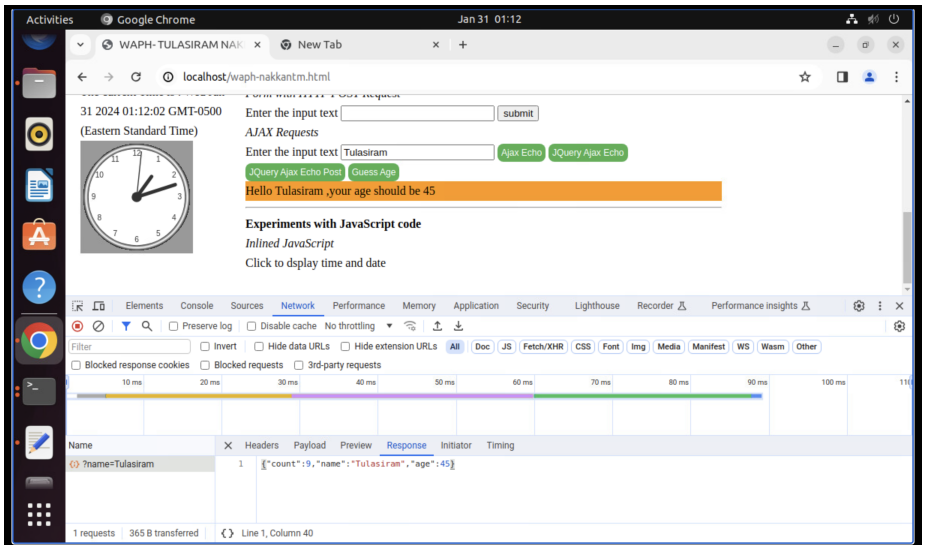

Figure 14: Response from api.agify.io

Below is the final webPage after completing all the tasks.

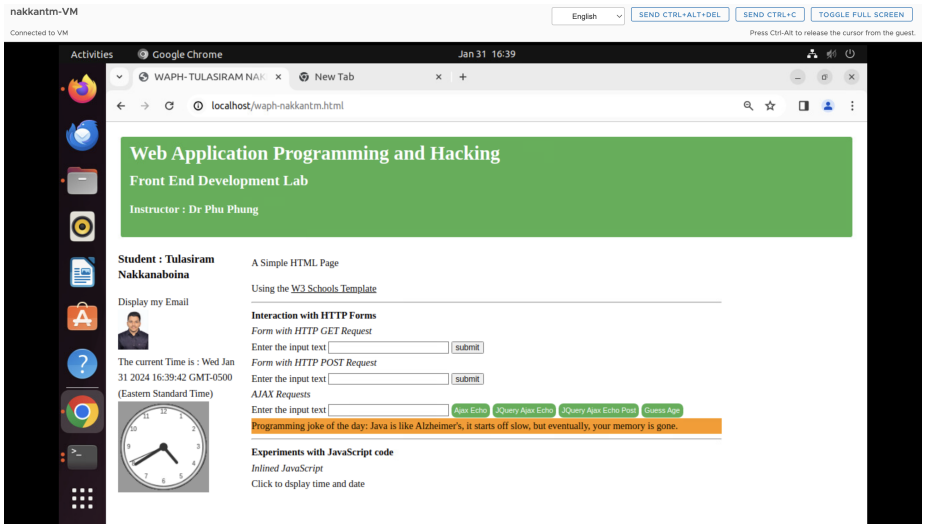

Figure 15: Lab 2 waph-nakkantm.html

Post this Labs/Lab2 folder was created to accomodate the project report and the changes were pushed. Pandoc tool was used to generate the project report from the README.md file