WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Ruthvik Suvarnakanti

Email: suvarnrk@mail.uc.edu

Figure 1: Ruthvik Suvarnakanti

Lab 2 - Front End Web Development

Overview: This lab deals with Front End Development. This lab gave 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 https://github.com/suvarnrk/waph-suvarnrk/blob/main/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-suvarnrk.html

Included file `waph-suvarnrk.html`:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>WAPH- Ruthvik Suvarnakanti</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 : Ruthvik Suvarnakanti</h3>
        <img src="images/headshot.jpg" alt="Ruthvik Headshot image" 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 forms</b>
        <div>
            <i> Form with an HTTP GET request</i>
            <form action="/echo.php" method="GET">
                Your Input: <input name="input">
                <input type="submit" value="Submit">
            </form>

        </div>
        <div>
            <i> Form with an HTTP POST request</i>
            <form action="/echo.php" method="GET" name="echo_post">
                Your Input: <input name="input" onkeypress="console.log('You pressed a key')
                <input type="submit" value="Submit">
```

```
            </form>
            </div>
        </div>
    </div>
</div>
</body>
</html>
```

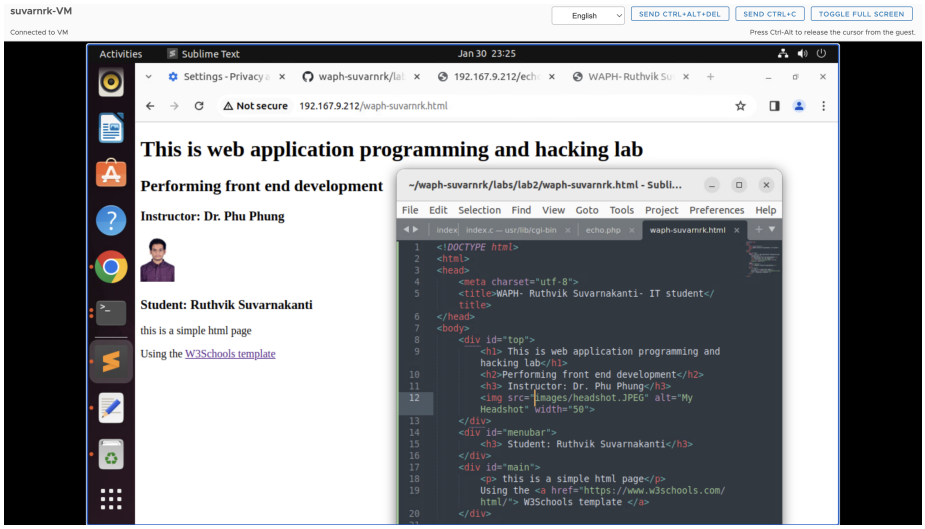

Figure 2: A simple HTML Page

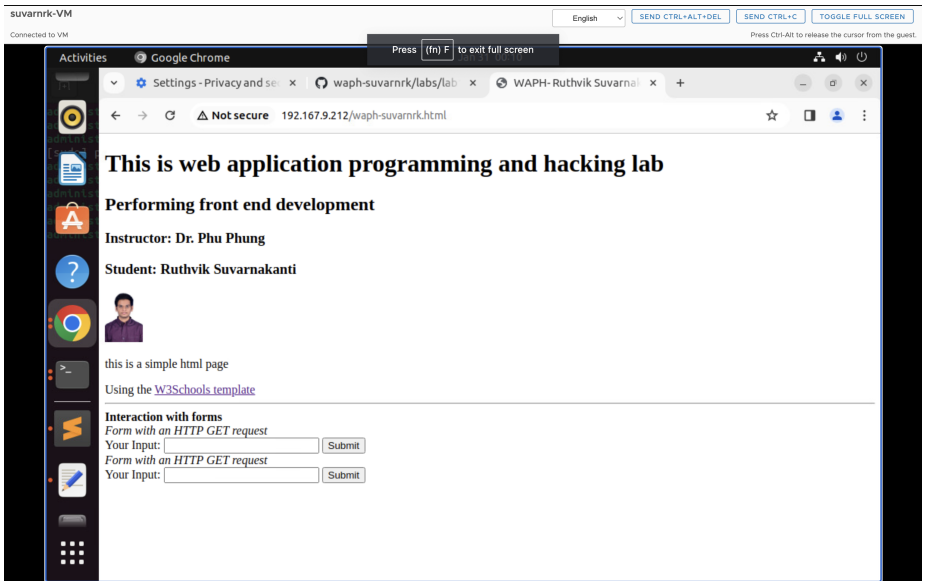

Figure 3: A simple HTML Page

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

-JavaScript code in a

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

```JavaScript
    var shown=false;
function showhideEmail(){
    if(shown){
        document.getElementById('email').innerHTML="Show ny email";
        shown =false;
    }else{
```

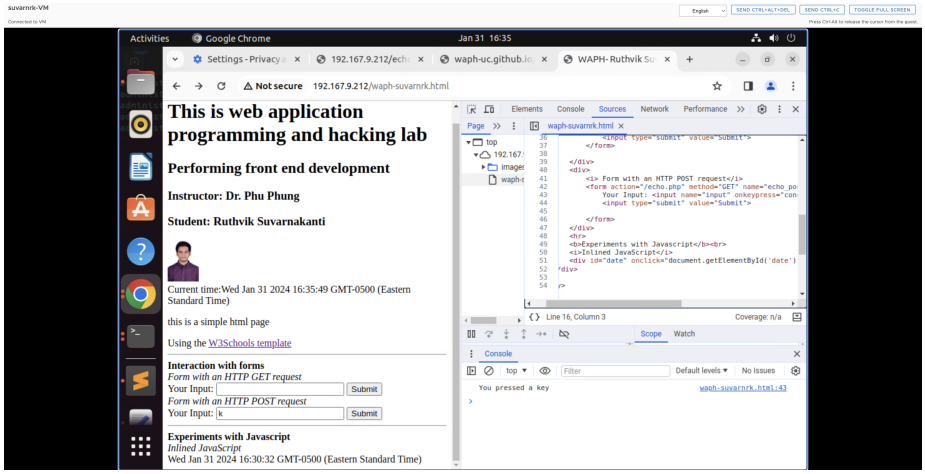

Figure 4: Display date/time when clicked

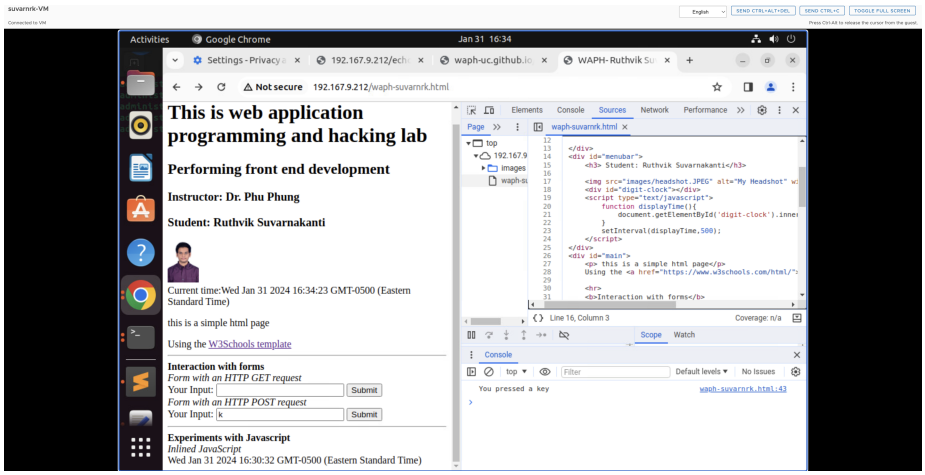

Figure 5: Display digital clock

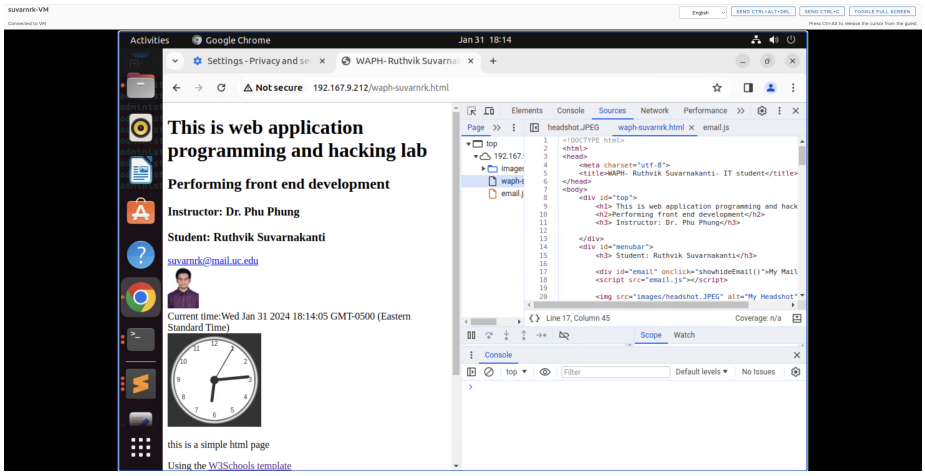

Figure 6: show email when clicked

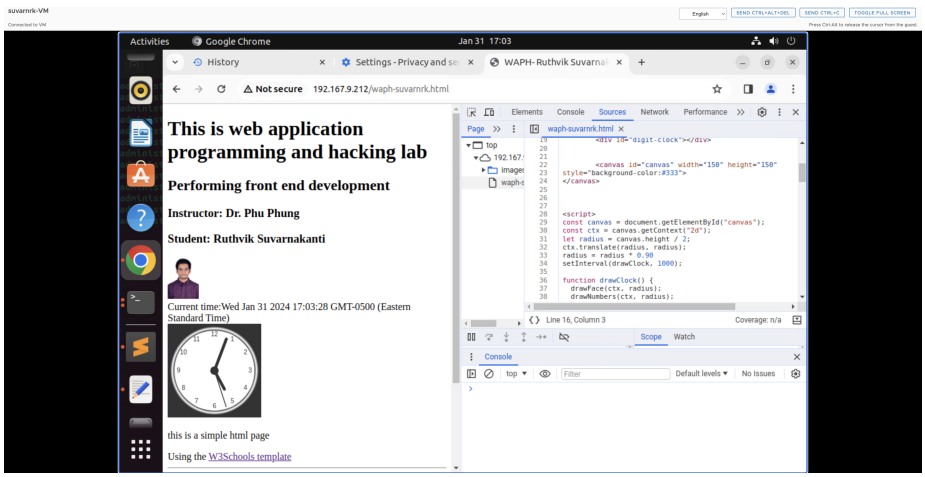

Figure 7: Display analog clock

```
        var myemail="<a href='mailto:suvarnrk"+"@"+"mail.uc.edu'>suvarnrk"+"@"+"mail.uc.edu<
        document.getElementById('email').innerHTML=myemail;
        shown=true;
    }
}
```

```
    <div id="email" onclick="showOrHideEmail()">Show 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>
            const canvas = document.getElementById("canvas");
const ctx = canvas.getContext("2d");
let 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>
```

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.

```
<div>
    <i> Ajax Requests</i><br>
    Your Input:
    <input name="data"
    onkeypress="console.log('You have pressed a key ')" id="data">
    <input type="button" class="button round" value="Ajax Echo" onclick="getEcho()">
    <div id="response"></div>
        <input class="button round" type="submit" value="JQuery Ajax Echo" onclick="getJ
            <input class="button round" type="submit" value="JQuery Ajax Echo Post" oncl
            <input class="button round" type="submit" value="Guess Age" onclick="guessAg
            <div id="response"></div>
</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.

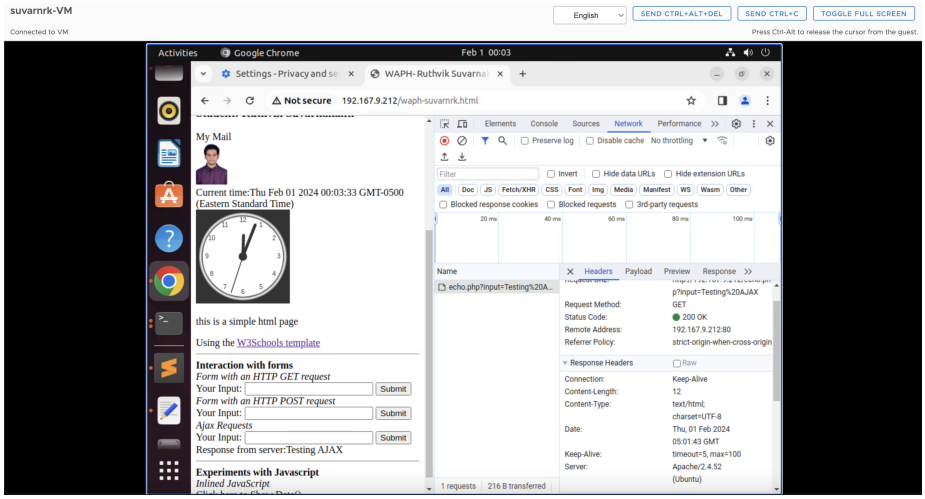

Figure 8: Making an Ajax get call and inspecting respponse

Task 2: CSS

a) Inline CSS

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

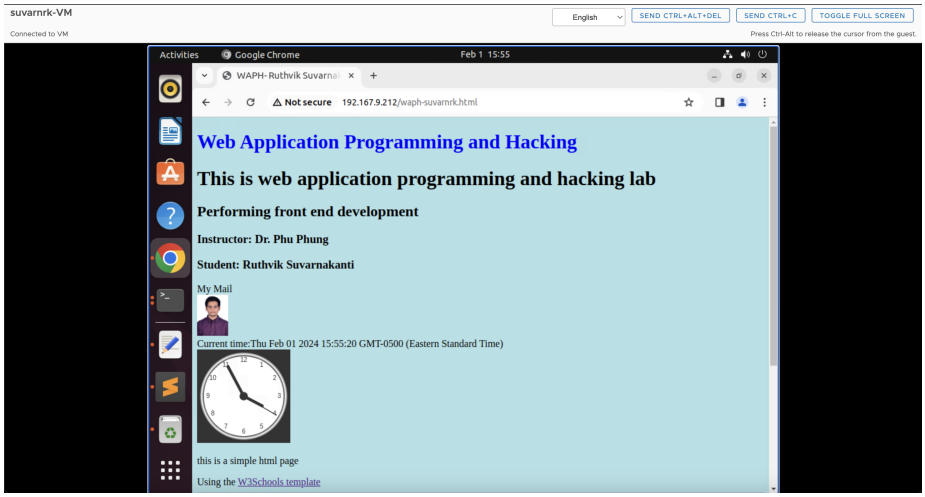

Figure 9: webpage after adding inline CSS

b) Internal CSS.

```
    <style>
        .button{
            background-color:green;
            border: none;
            color: white;
            padding: 5px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 12px;
            margin: 4px 2px;
            cursor: pointer;
        }
        .round{border-radius: 8px;}
        #response{background-color: orange;}
        }
    <!-- HTML code -->
    </style>
    input class="button round" type="submit" value="JQuery Ajax Echo" onclick="getJqueryAjax
                <input class="button round" type="submit" value="JQuery Ajax Echo Post" oncl
```

```
                <input class="button round" type="submit" value="Guess Age" onclick="guessAg
<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>
```

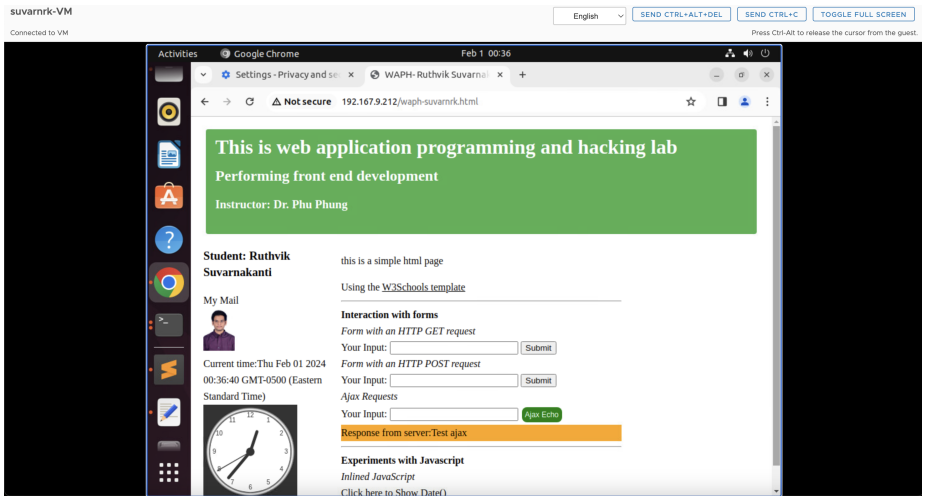

Figure 10: web page after adding internal CSS and external CSS

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.

```
<!-- HTML code -->
<input class="button round" type="submit" value="JQuery Ajax Echo" onclick="getJqueryAja
<!-- 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>
```

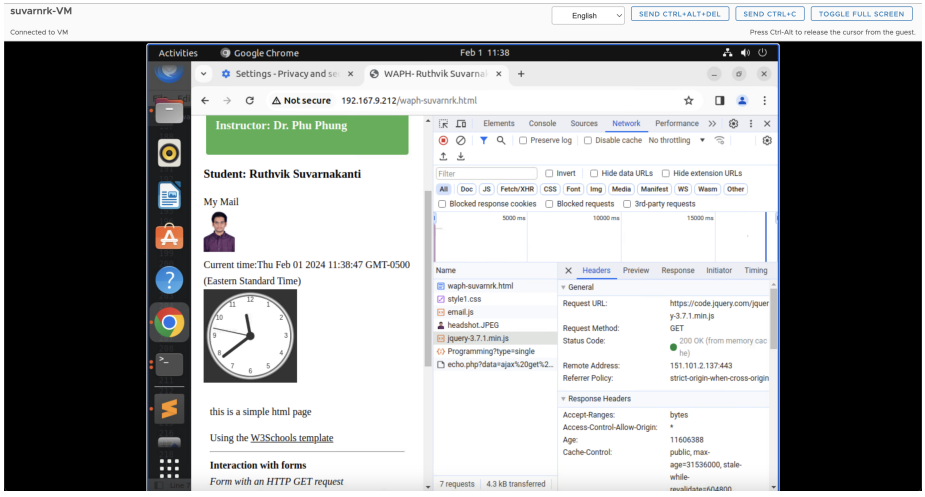

Figure 11: JQuery Ajax GET request to echo.php

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

```
<!-- 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>
```

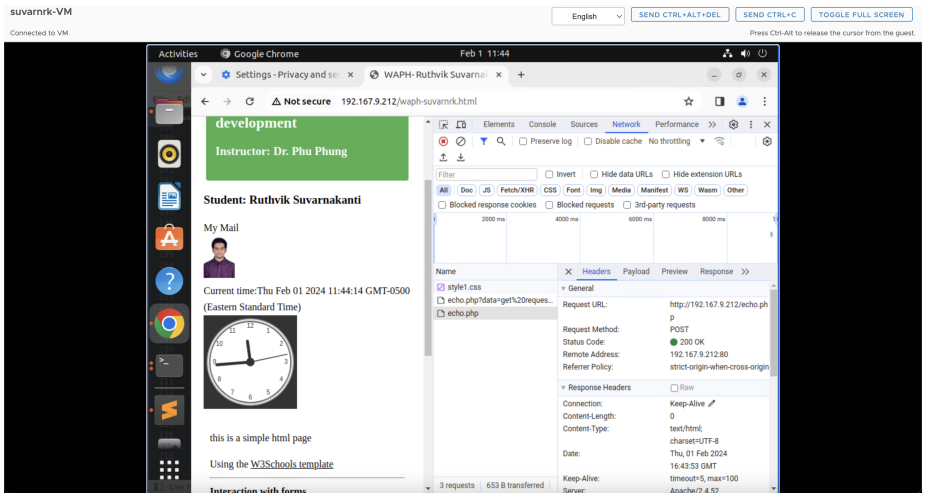

Figure 12: JQuery Ajax POST request to echo.php

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. out of this response 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.

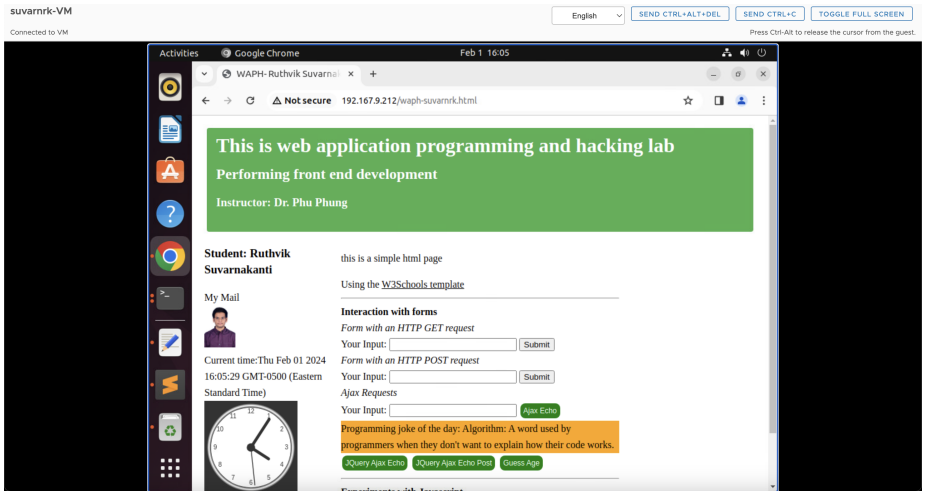

Figure 13: Random Joke displayed when the page is loaded

The below picture represents image of randomly created joke when page is loaded.

```
<!-- 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 -->
```

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.

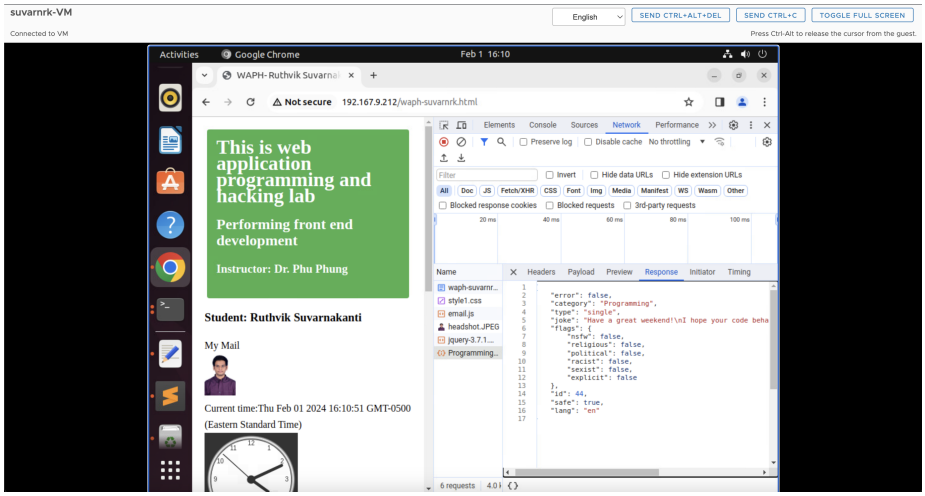

Figure 14: Response of the webservice in inspect view

```
<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>
```

The pictures of API calls are not responsive as there are too many request to API and gettting a status of 429 which is meant for request limit reached. I am attaching the pictures of the same here.

Below is the response of the API call.

Below is the final webPage after completing all the tasks and following the lectures.

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

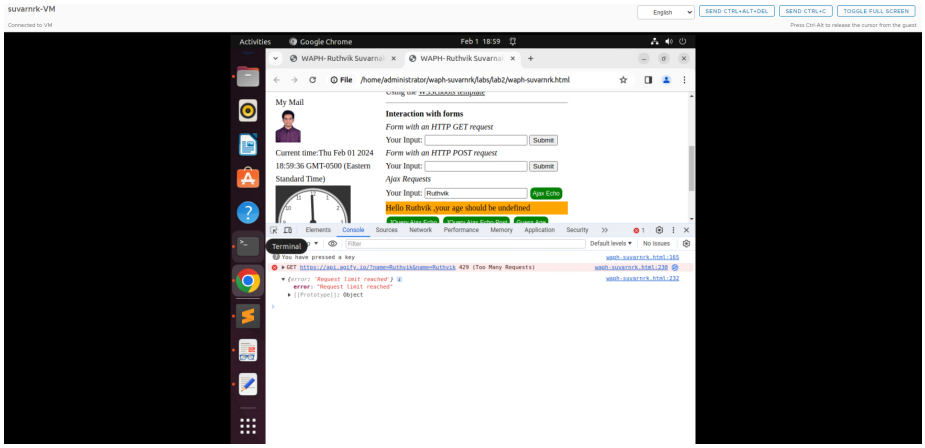

Figure 15: HTTP request to API

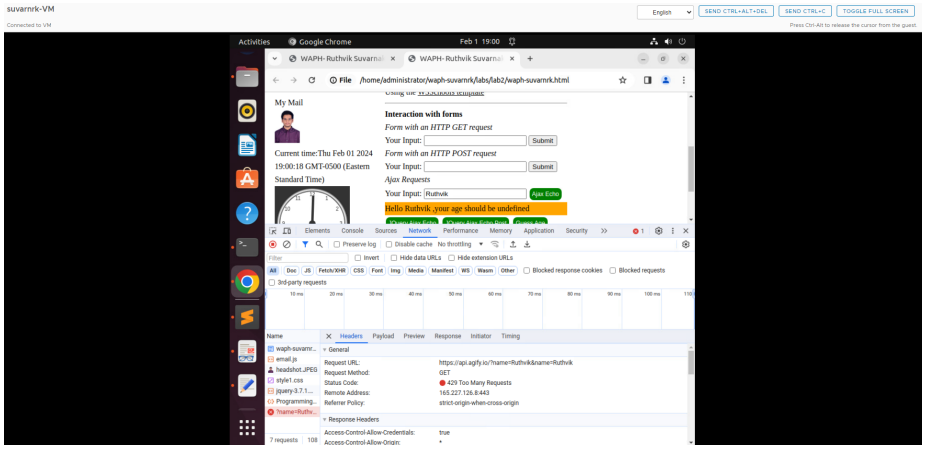

Figure 16: Response from API

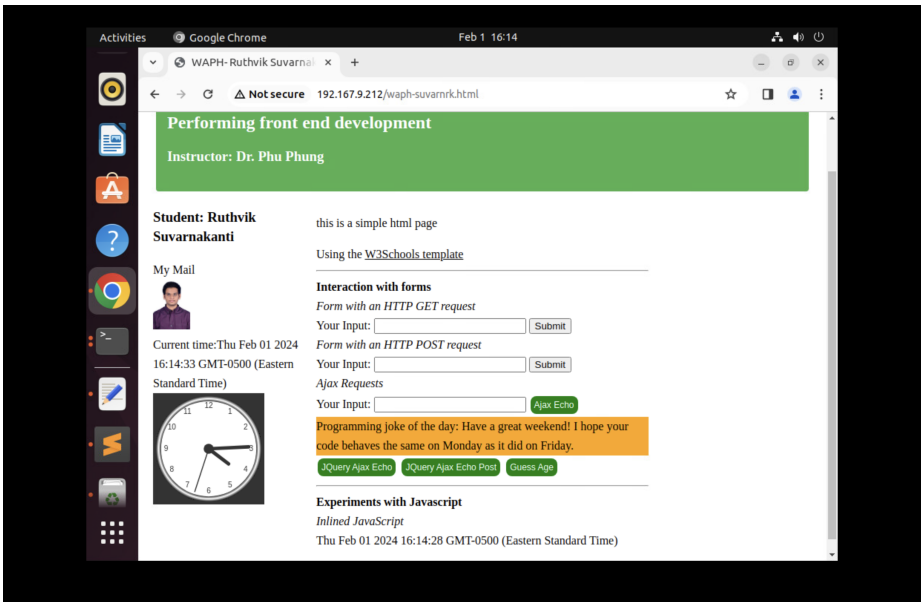

Figure 17: Ruthvik Suvarnakanti Final Page