

**waph-kanamala**

## **WAPH-Web Application Programming and Hack- ing**

**Instructor: Dr. Phu Phung**

**Student**

**Name:** Lakshmi Narayana Kanamarlapudi

**Email:** kanamala@mail.uc.edu

**Short-bio:** I am having interest towards data science and web development.



Figure 1: Lakhmi Narayana headshot

### **Repository Information**

Repository's URL: <https://github.com/LakshmiNarayanaKanamarlapudi/waph-kanamala.git>

This is a private repository for Kanamarlapudi Lakshmi Narayana to store all code from the course. The organization of this repository is as follows.

### **Labs**

Hands-on exercises in lectures

- <https://github.com/LakshmiNarayanaKanamarlapudi/waph-kanamala/tree/main/labs/lab2>: Front-end Web Development

### **Lab Overview**

**Task-1 : Basic HTML with forms and javascript** - In this task we can learn about the basic tags and forms. Their we can also get to know how to use java script. - In this task as a second section we can also learn about the digital clock, email hide and unhide, analog clock

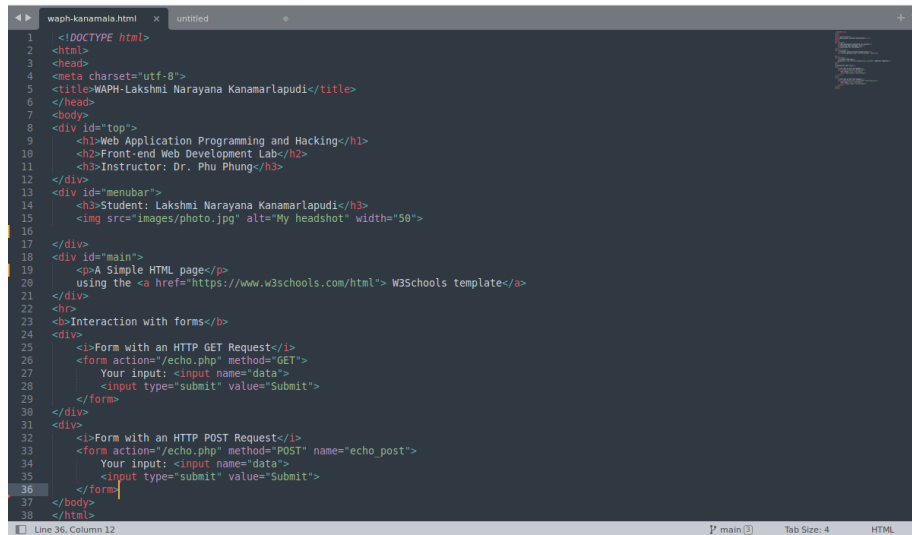
**Task-2 : Ajax, CSS, jQuery and Web API Integration** - In this part we learned about the ajax get request and response and about buttons. - Also we can learn the styling part which includes inlined, internal and external styling. - we will learn about the jQuery and web api integration using the simple joke and guess api's.

## Lab 2

### Task-1 : Basic HTML with forms and javascript

#### A

- In this task we have to create a basic html file with an image tag in it.
- Where I have used h1,h2,h3 tags for the headings and a image tag to insert the headshot in it.
- And also used a form tag to insert the form to check the HTTP get and post request using the echo.php file.
- The following is the form tag "
- The following is the image tag "".
- Also created buttons to test the get and post request.
- Basic HTML code is (fig2) & Basic HTML output is (fig3)



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>WAPH-Lakshmi Narayana Kanamarlapudi</title>
6 </head>
7 <body>
8 <div id="top">
9 <h1>Web Application Programming and Hacking</h1>
10 <h2>Front-end Web Development Lab</h2>
11 <h3>Instructor: Dr. Phu Phung</h3>
12 </div>
13 <div id="menubar">
14 <h3>Student: Lakshmi Narayana Kanamarlapudi</h3>
15 
16 </div>
17 <div id="main">
18 <p>A Simple HTML page</p>
19 using the <a href="https://www.w3schools.com/html"> W3Schools template</a>
20 </div>
21 <h2>
22 <b>Interaction with forms</b>
23 <div>
24 <i>Form with an HTTP GET Request</i>
25 <form action="/echo.php" method="GET">
26 Your input: <input name="data">
27 <input type="submit" value="Submit">
28 </form>
29 </div>
30 <div>
31 <i>Form with an HTTP POST Request</i>
32 <form action="/echo.php" method="POST" name="echo_post">
33 Your input: <input name="data">
34 <input type="submit" value="Submit">
35 </form>
36 </div>
37 </body>
38 </html>
```

Figure 2: Basic HTML code

#### B

- Firstly i have inserted a code in the html file to display basic digital-clock which shows the current date and time.
- For this once the html file was opened web browser we will see a digital clock. If we click on it it will shows the current date and time.
- we have used the inline java script to acheive the task.
- Digital-clock code is (fig4) & Digital-Clock image output (fig5)
- This is also similar to the above task but here we have used the javascript to get the output.

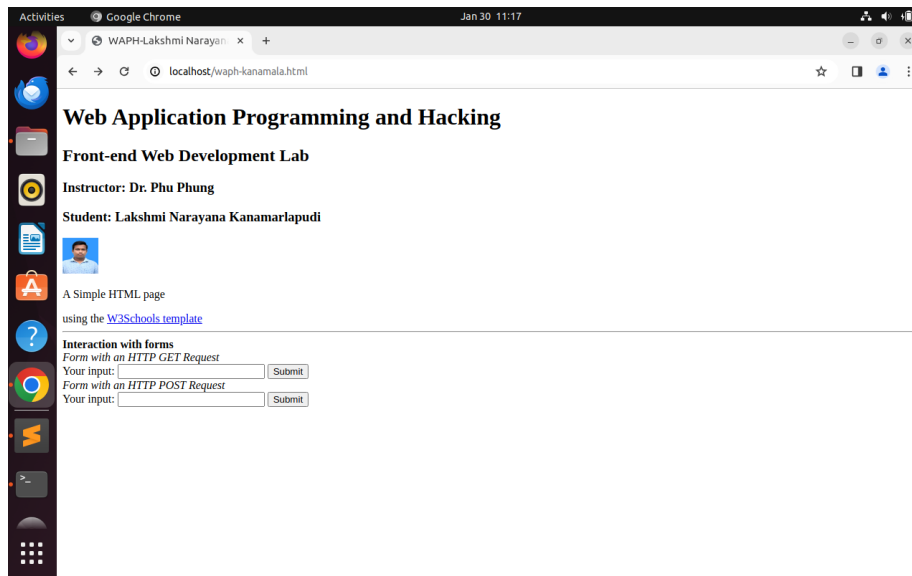


Figure 3: Basic HTML output

```

36 <hr>
37 <b>Experiments with JavaScript code</b><br>
38 <i>Inlined JavaScript</i>
39 <div id="date" onclick="document.getElementById('date').innerHTML=Date()">Click here to show Date()</div>
40 </div>
41 </body>
42 </html>

```

Figure 4: Digital-Clock code

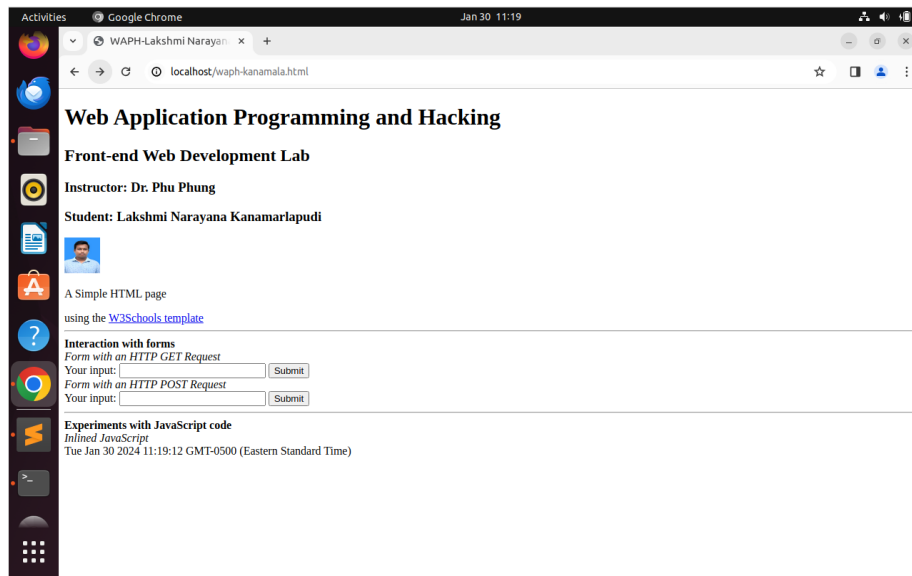


Figure 5: Digital-Clock image

- Here we have used a fuction called “fuction displayTime”. And also we have used the setInterval fuction to give an time interval of 500
- Below is the source code and output.
- Digital-Clock with javascript code (fig6) & Digital-Clock with javascript output (fig7)

```

18 <div id="digit-clock"></div>
19 <script>
20 function displayTime() {
21     document.getElementById('digit-clock').innerHTML = "Current time: " + new Date();
22 }
23 setInterval(displayTime, 500);
24 </script>

```

Figure 6: Digital-Clock with javascript code

- In this we are trying acheive that when we click on show my email id it have to show the email.
- upon another click it has to hide the email. For this we have created a java script file with the specified coding to acheive the task.
- For that we have created a js file in that we have included the required fuctions and elements like “function showhideEmail” and "document.getElementById('email').innerHTML = “Show my email”;" are used to make the task.
- The output and the source code are as follows.
- email javascript code (fig8)& email code (fig9) & email output (fig10)

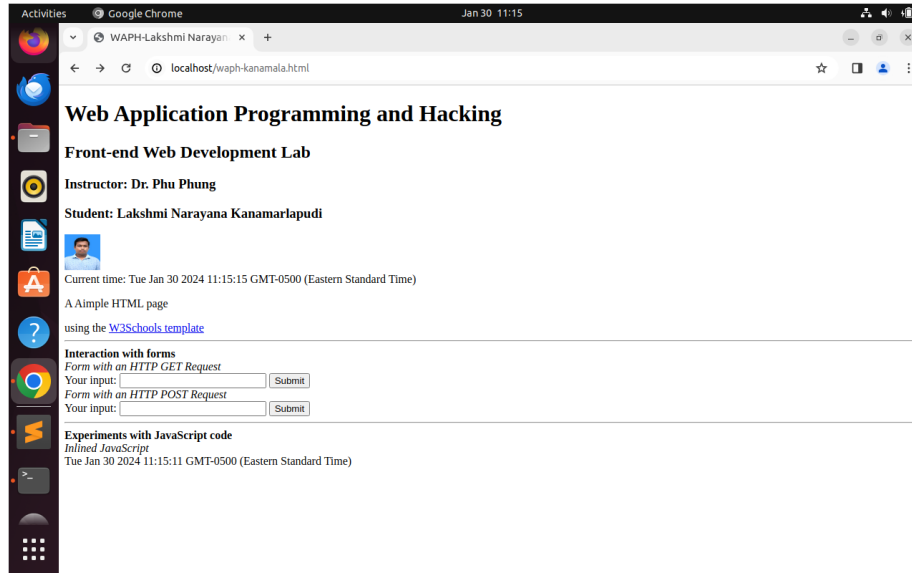


Figure 7: Digital-Clock with javascript output

```

1  var shown= false;
2  function showhideEmail(){
3      if (shown){
4          document.getElementById('email').innerHTML = "Show my email";
5          shown = false;
6      }else{
7          var myemail = "<a href='mailto:kanamala+'@' + 'mail.uc.edu'> kanamala+'@' + 'mail.uc.edu'</a>";
8          document.getElementById('email').innerHTML=myemail;
9          shown=true;
10     }
11 }
12

```

Figure 8: email javascript code

```

15  <div id="email" onclick="showhideEmail()">Show my email</div>
16  <script src="email.js"></script>

```

Figure 9: email code

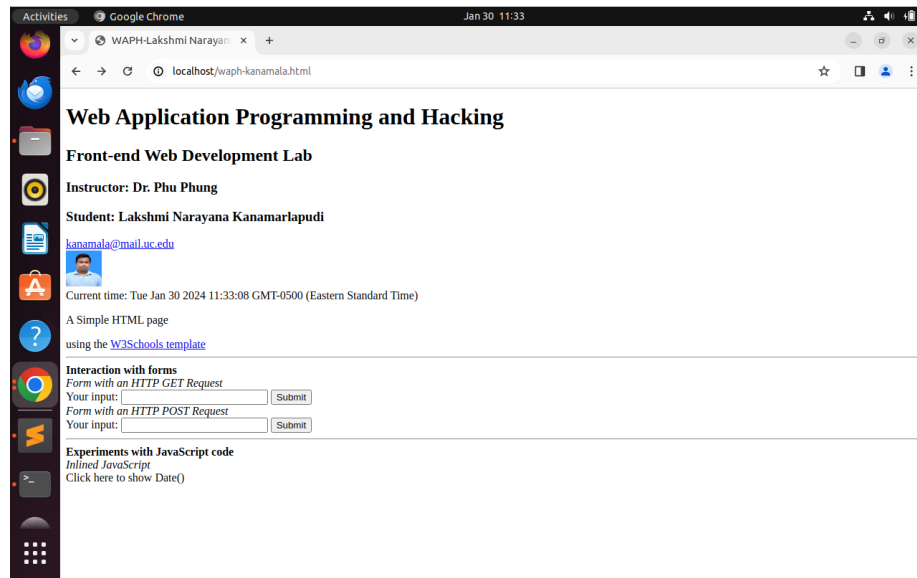


Figure 10: email output

- In this task we are producing an analog clock for the extension of digital clock which was done in the previous tasks.
- We have used the canvas tag to get the 2 dimensional context to the clock.
- Here I have also defined the radius, ctx and interval for the clock.
- And then finally used the draw function to display the analog clock. “function draw()”
- The source code and output are shown below.
- Analog clock code (fig11) & Analog clock output(fig12)

## Task 2 - Ajax, CSS, jQuery and Web API Integration

### A - Ajax

- Ajax is the one which can request the server and get the response from the server without reloading the web page.
- “function get echo” has been used to take care of the input which the user input.
- Then “xhr onreadystatechange” is the one which takes care about the request, response and display of the response.
- After that the get request will be handled by the echo.php which is a php program.
- Once after the request was submitted and then there will be state of the response which was estimated in numerical. That it starts from one and if it equals to four and the status is 200 then the response will be displayed.

```

20 <div id="digit-clock"></div>
21 <canvas id="analog-clock" width="150" height="150" style="background-color:#999"></canvas>
22 <script src="https://waph-uc.github.io/clock.js"></script>
23 <script>
24   function displayTime() {
25     document.getElementById('digit-clock').innerHTML = "Current time: " + new Date();
26   }
27   setInterval(displayTime, 500);
28   var canvas = document.getElementById("analog-clock");
29   var ctx = canvas.getContext("2d");
30   var radius = canvas.height / 2;
31   ctx.translate(radius, radius);
32   radius = radius * 0.99;
33   setInterval(drawClock, 1000);
34
35   function drawClock() {
36     drawFace(ctx, radius);
37     drawNumbers(ctx, radius);
38     drawTime(ctx, radius);
39   }
40 </script>

```

Figure 11: Analog clock code

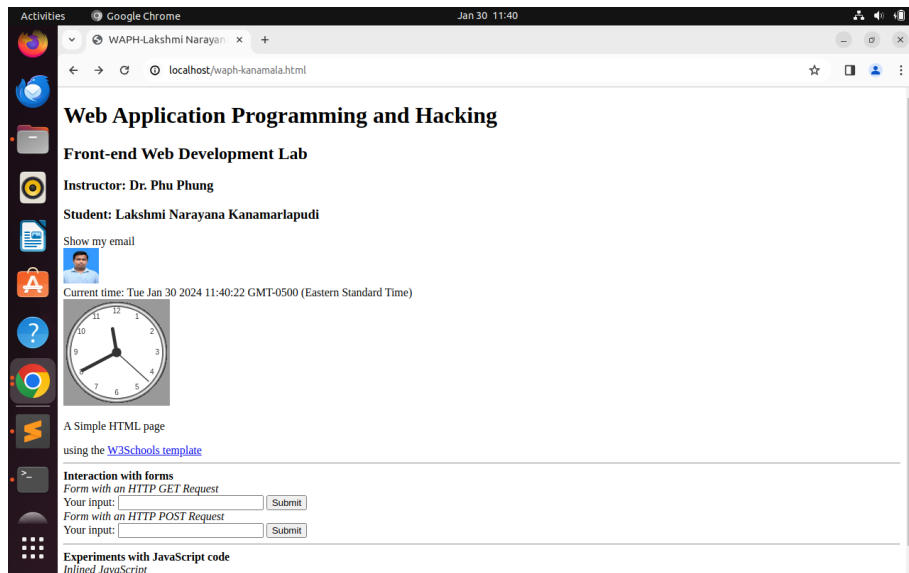


Figure 12: Analog clock output

- We can also check the network log to see the request and response.
- Ajax source code (fig13) & Ajax output (fig14) & Ajax output network fig(15)

```

37 drawNumbers(ctx, radius);
38 drawLine(ctx, radius);
39 }
40
41 function getEcho(){
42     var input = document.getElementById("data").value;
43     if (input.length == 0) {
44         return;
45     }
46     var xhttp = new XMLHttpRequest();
47     xhttp.onreadystatechange = function() {
48         if (this.readyState == 4 && this.status == 200) {
49             console.log("Received data:" + xhttp.responseText);
50             document.getElementById("response").innerHTML = "Response from server:" + xhttp.responseText;
51         }
52     }
53     xhttp.open("GET", "echo.php?data=" + input, true);
54     xhttp.send();
55     document.getElementById("data").value = "";
56 }
57
58 </script>
59
60 <div id="main">
61     <p>A Simple HTML page</p>
62     using the <a href="https://www.w3schools.com/html"> W3Schools template</a>
63 </div>
64
65 <div>
66     <h2>Interaction with forms</h2>
67     <div>
68         <h3>Form with an HTTP GET Request</h3>
69         <form action="/echo.php" method="GET">
70             Your input: <input name="data">
71             <input type="submit" value="Submit">
72         </form>
73     </div>
74     <div>
75         <h3>Form with an HTTP POST Request</h3>
76         <form action="/echo.php" method="POST" name="echo post">
77             Your input: <input name="data">
78             <input type="submit" value="Submit">
79         </form>
80     </div>
81     <div>
82         <h3>Ajax Requests</h3>
83         Your input:
84         <input name="data" onkeypress="console.log('You have pressed a key');" id="data">
85         <input class="button round" type="button" value="Submit" onclick="getEcho()">
86         <div id="response"></div>
87     </div>
88 </div>

```

Figure 13: Ajax source code

- I have observed the network window in which we can get to the request and response.
- In which i have seen the request method as GET and status code as 200.
- We can also check connection status, remote code and many more.

## B - CSS

- In this task we will achieve the styling process which provides the interactivity to the webpage.

### Inline style

- Inline style is a simple style tag which is used to give the basic color to the text in the webpage.
- Where we will include the style inside the heading tag itself.
- inline source code (fig16) & inline source output (fig17)

### Internal CSS

- Internal CSS is a style format where we can define the style for the multiple contents and backgrounds also.
- For this internal CSS we will write a separate script defining the colour and style for the contents in the webpage.
- I have added the style to the heading tag and the background of the website. Which are powder blue to the background and blue for the heading tag.



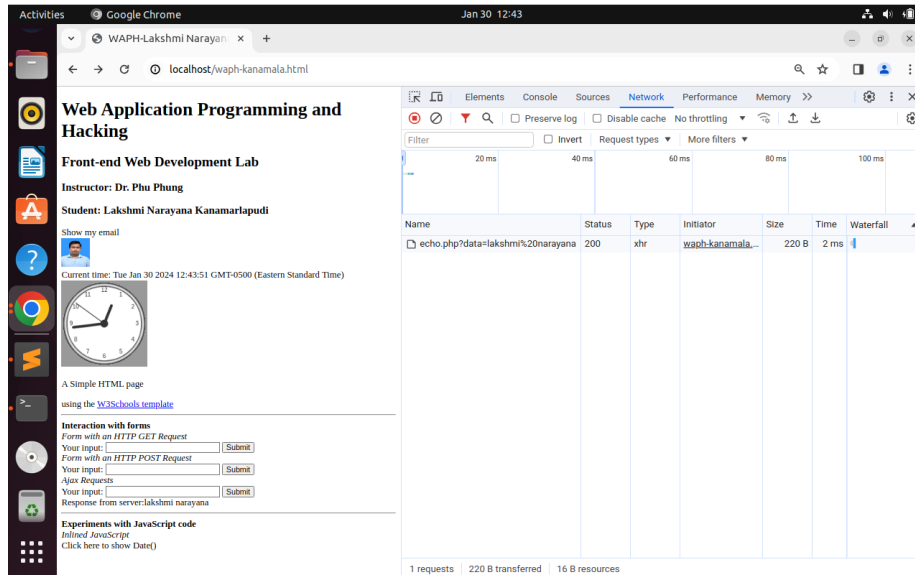


Figure 14: Ajax output

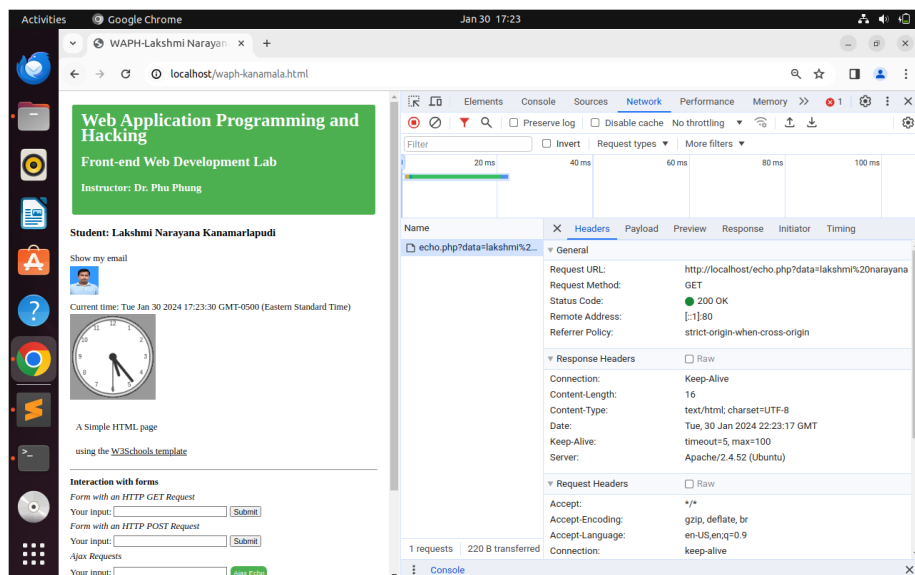


Figure 15: Ajax output network

```

8 <div id="top">
9 <h1 style="color:red;">Web Application Programming and Hacking</h1>
10 <h2>Front-end Web Development Lab</h2>
11 <h3>Instructor: Dr. Phu Phung</h3>
12 </div>
13 <div id="menubar">
14 <h3 style="color:purple;">Student: Lakshmi Narayana Kanamarlapudi</h3>
15 <div id="email" onclick="showhideEmail()">Show my email</div>
16 <script src="email.js"></script>
17 
18

```

Figure 16: inline source code

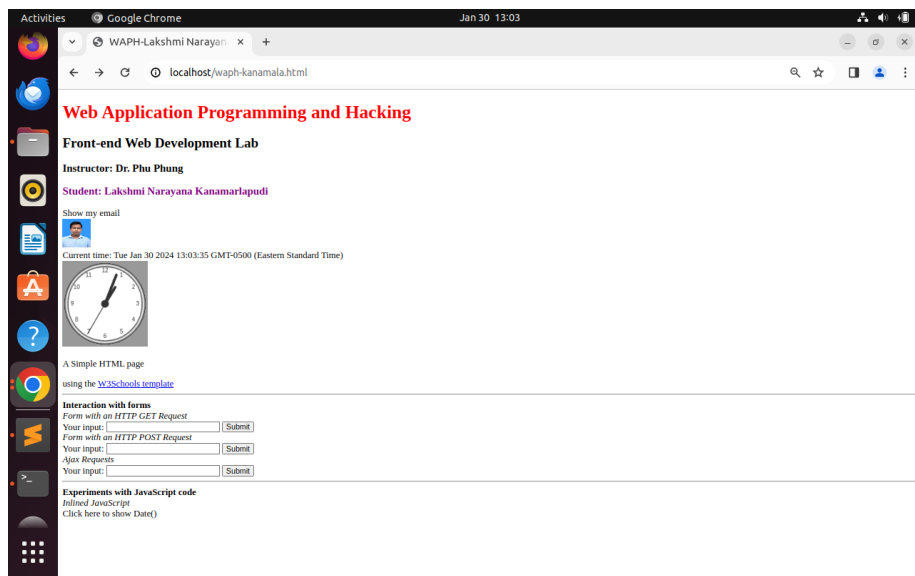


Figure 17: inline output

- internal source code (fig18) & internal output (fig19)

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>WAPH-Lakshmi Narayana Kanamarlapudi</title>
6  <style>
7      body {background-color: powderblue;}
8      h1 {color: blue;}
9  </style>
10 </head>

```

Figure 18: internal source code

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>WAPH-Lakshmi Narayana Kanamarlapudi</title>
6  <style>
7      body {background-color: powderblue;}
8      h1 {color: blue;}
9  </style>
10 </head>

```

Figure 19: internal output

## External CSS

- External CSS is nothing but acquiring the style and color of the outside webpages.
- For this we have added the style sheet using the “rel” and “herf” tag.
- We have added this required code to our html file from the github source which will change our whole page style and color as per the desired values.
- Here we have added green color to the desired values.
- external source code (fig20) & external output (fig21)

```

6  <link rel="stylesheet" href=https://waph-uc.github.io/style1.css>
7  <style>
8      .button{
9          background-color: #4CAF50;
10         border: none;
11         color: white;
12         padding: 5px;
13         text-align: center;
14         text-decoration: none;
15         display: inline-block;
16         font-size: 12px;
17         margin: 4px 2px;
18         cursor: pointer;
19     }
20     .round {border-radius: 8px;}
21     #response {background-color: #ff9800;}
22 </style>

```

Figure 20: external source code

## C - jQuery

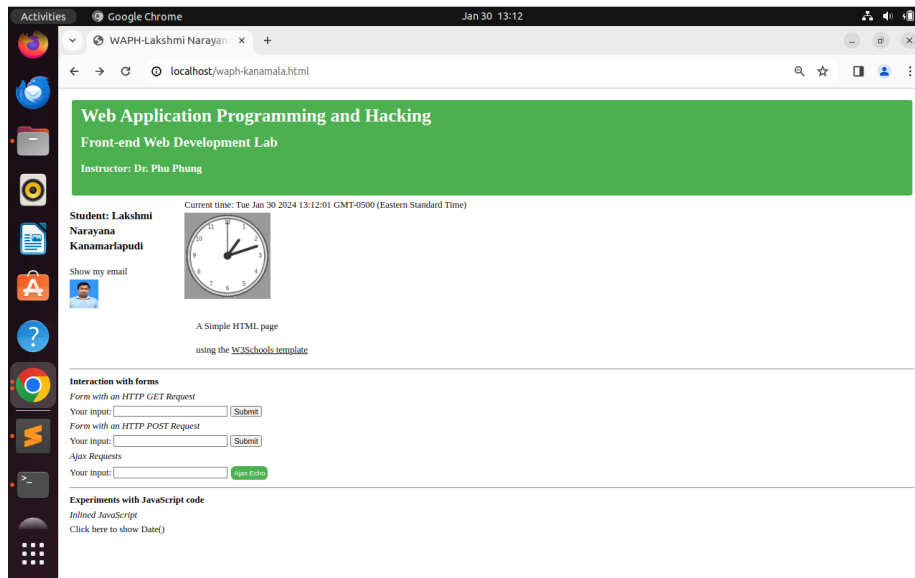


Figure 21: external output

- **jQuery-GET**
- A function call `jQueryAjax` has been defined to handle the get function. Where we have also created a button at the bottom of the html page to add a button.
- I have written a length variable to check the length of the input. And the input will be handled by the `echo.php`.
- Where `$.post("echo.php", {data: input})` `$get` will send the get request to the `echo.php` with an input parameter.
- "function result" and the inside line of the code will display us the response which is got from the server.
- In this way all the code and process will work.
- jQuery GET source code (fig22) & jQuery GET output (fig23)

```

75     }
76     function jQueryAjax() {
77         var input = $("#data").val();
78         if (input.length == 0) return;
79         $.get("echo.php?data="+input,
80             function(result) {
81                 $("#response").html("Response from server:" + result);
82             }
83         );
84         $("#data").val("");
85     }

```

Figure 22: jQuery GET source code

- **jQuery-POST**

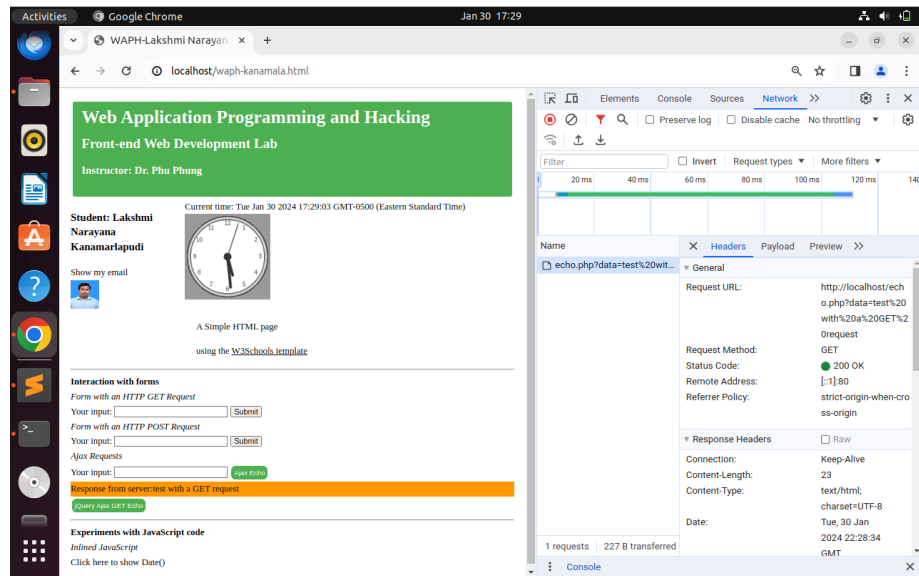


Figure 23: jQuery GET output

- This is as same as the above one the only difference is method. This POST request method.
- I have created a “jQueryAjaxPost” function which is call the post method. And inside the fuction defined a variable called data to check the input from the user.
- And a length variable has been assigned which will checks the length of the data. \$post will sends the post request to the echo.php.
- fuction result will display us the response which we got .
- The network window will be same as of the get request. The only change is request method. Here we got it has POST.
- jQuery POST source code (fig24) & jQuery POST output (fig25)

```

85     }
86     function jQueryAjaxPost() {
87         var input = $("#data").val();
88         if (input.length == 0) return;
89         $.post("echo.php", {data: input},
90             function(result) {
91                 $("#response").html("Response from server:" + result);
92             }
93         );
94         $("#data").val("");
95     }

```

Figure 24: jQuery POST source code

## D - Web API Integration

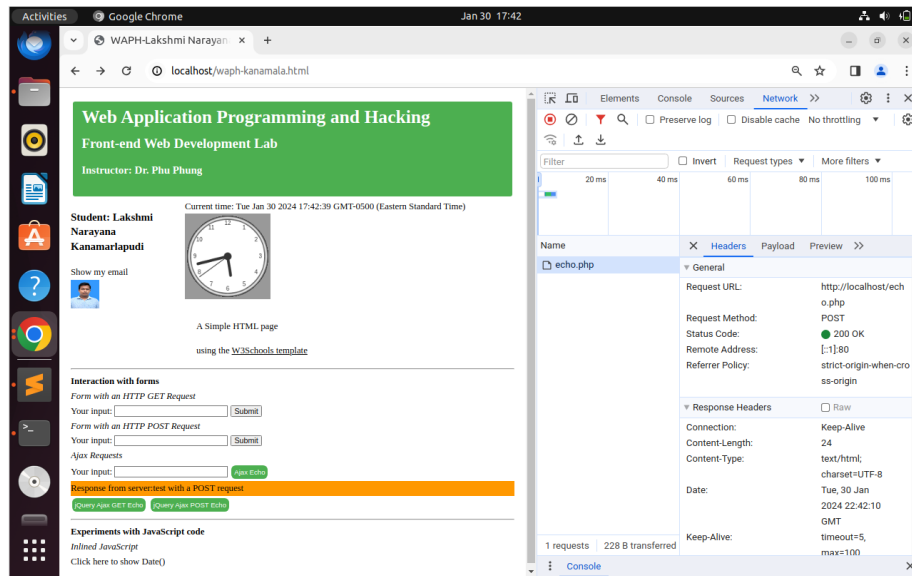


Figure 25: jQuery POST output

### • JOKE API

- Joke api is used to generate the random joke in the web page, where a new joke will be generated every time when we refresh the web page.
- Web page integration is one of the best feature which will makes our work simple and also we can used api as per our requirement.
- For this we are using the API. Where I have writtented a code as per the requirements of the task.
- “\$get” will sends the request to the API, which also make sures that only one joke should be printed at a time.
- A callback function was written which is “fuction(result)” which will displays us the response which was came from the API.

- Where i have also analysed the network window in which the request method was “GET”.

- Joke API source code(fig26) & Joke API output(fig27)

### • Guess API

- Name API is used to guess the age of the input everytime when a new name was entered.

```

95     }
96     $.get("https://v2.jokeapi.dev/joke/Programming?type=single",
97     function(result){
98         console.log("From jokeAPI: " + JSON.stringify(result));
99         $("#response").html("A programming joke of the day: " + result.joke);
100     })
101 );

```

Figure 26: Joke API source code

The screenshot shows a web browser window with the URL `localhost/waph-kanamala.html`. The page title is "Web Application Programming and Hacking" and the instructor is "Dr. Phu Phung". The page displays a clock showing the current time as "Tue Jan 30 2024 17:46:09 GMT-0500 (Eastern Standard Time)". Below the clock, there is a section titled "Interaction with forms" with input fields for "Form with an HTTP GET Request", "Form with an HTTP POST Request", and "Ajax Requests". A yellow box contains a programming joke: "A programming joke of the day: Two C strings walk into a bar. The bartender asks 'what can I get ya?' The first string says 'I'd have a gin and tonic.' The second string thinks for a minute, then says 'I'd like a regular sunrise!' (43f1956@10jKqW0j0\*FNINj0BN134ufh1u33d5f981308d384d081b3964b5f7nag)". The network panel on the right shows a request to `https://v2.jokeapi.dev/joke/Programming?type=single` with a status code of 200 OK. The response headers include `Access-Control-Allow-Headers`, `Access-Control-Allow-Methods`, `Access-Control-Allow-Origin`, `Access-Control-Request-Method`, and `Allow`.

Figure 27: Joke API output

- For this firstly a button was created the code was included at the end of the html code.
- Then asynchronous fuction was written which is called “guessage” with a parameter “name”.
- “const reponse” is a variable which is used to fetch the data form the API and await will waits until the response was came from the API.
- And along with the “\$response” string concatination was included to make a greeting text along with the input data.
- Finally the age will be displayed along with the name.
- Where i have also analysed the network window in which the request method was “GET”.
- Guess API source code fig(28) & Guess API source code(fig29)

```

102     async function guessAge(name){
103         const response = await fetch("https://api.agify.io/?name="+name);
104         const result = await response.json();
105         $("#response").html("Hi " + name + ", your age should be " + result.age);
106     }
107

```

Figure 28: Guess API source code

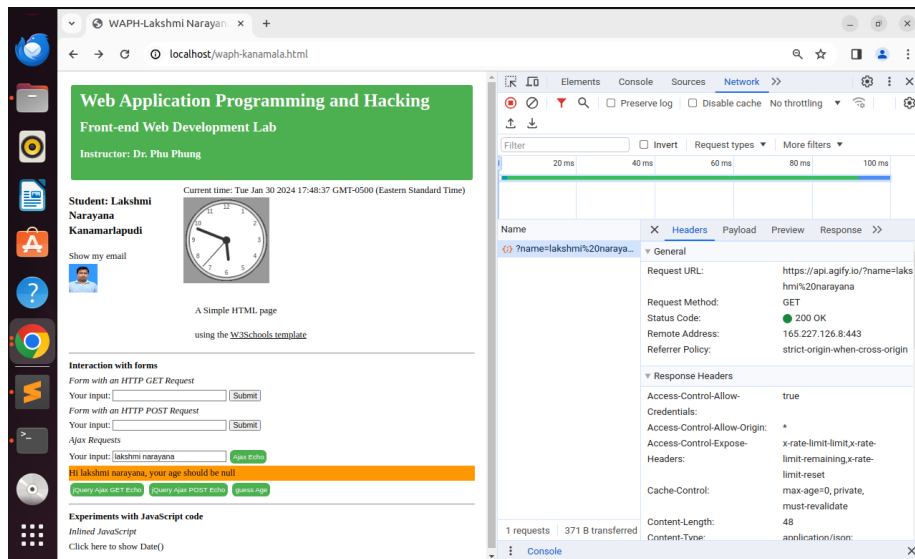


Figure 29: Guess API output