

WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: LakshmaReddy Bandi

Email: bandild@mail.uc.edu



Figure 1: LakshmaReddy Bandi

Lab 2 - Front-end Web Development

Overview

This lab covered basic HTML forms, Javascript, Ajax and jQuery basics for front-end requests, basic CSS template, and basic Web API integration for loading programming jokes and age guesses.

I have uploaded the code in my private repository <https://github.com/bandild/waph-bandild/blob/main/labs/lab2/README.md> #### Task 1: Basic HTML with forms, and JavaScript

Task-1 involves creating basic HTML forms, analog-clock, digital, and clock using JavaScript.

a. HTML The task involves creating an HTML file named ‘waph-bandit.html’, which includes basic html tags and includes my headshot image.

b. Simple JavaScript In this part b task-1, I have done the following using javaScript:

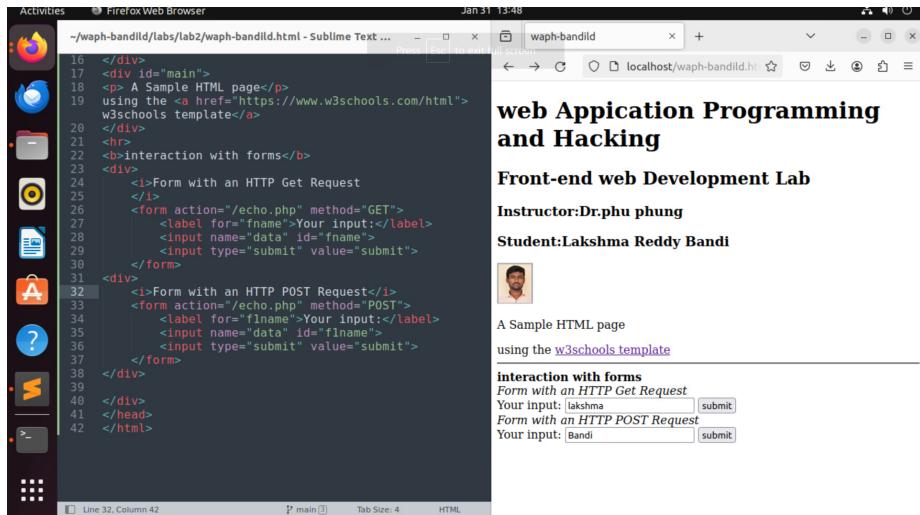


Figure 2: Simple HTML webpage with headshot

The web-page features a JavaScript code that displays the current data-time when a key is pressed, incorporated into the HTML file.

```
<b>Experiment with JS</b>
<i>Inlined Javascript</i>
<div id="date" onclick="document.getElementById('date').innerHTML=Date()">Click Here to view</div>
</body>
</html>
```

After you presses the button.

The data-time display was followed by a digital-clock displayed at the top, incorporating a JS code from an HTML file within Script tag.

```
<script type="text/javascript">

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

The JS code for displaying/hiding an email was written, stored in an external email.js file, and linked to the.html file using a jQuery function.

```
var shown = false;
```

```

16    </div>
17    <div id="digital-clock"></div>
18    <script>
19      function displayTime(){
20        document.getElementById('digital-clock').innerHTML="current time: " + new Date();
21      }
22      setInterval(displayTime,500);
23    </script>
24    <div id="main">
25      <p>A Sample HTML page</p>
26      using the <a href="https://www.w3schools.com/html">
27      w3schools template</a>
28    </div>
29    <b>Interaction with forms</b>
30    <div>
31      <i>Form with an HTTP Get Request
32      </i>
33      <form action="/echo.php" method="GET">
34        <label for="fname">Your input:</label>
35        <input name="data" id="fname" onkeyup="console.log('you have pressed a key')">
36        <input type="submit" value="submit">
37      </form>
38    </div>
39    <div>
40      <i>Form with an HTTP POST Request</i>
41      <form action="/echo.php" method="POST">
42        <label for="filename">Your input:</label>
43        <input name="data" id="filename">
44        <input type="submit" value="submit">
45      </form>

```

Line 22, Column 34 P main [?] Tab Size: 4 HTML

Figure 3: In line javascript for displaying data/time after key pressed

```

30    </form>
31  </div>
32  <div>
33    <i>Form with an HTTP POST Request</i>
34    <form action="/echo.php" method="POST">
35      <label for="filename">Your input:</label>
36      <input name="data" id="filename">
37      <input type="submit" value="submit">
38    </form>
39  </div>
40  <br>
41  <b>Interaction with javascript</b>
42  <div>
43    <p id='data'></p>
44    <button onclick="document.getElementById('data').innerHTML=new Date()">click me to show Date!</button>

```

Line 44, Column 104 P main [?] Tab Size: 4 HTML

Figure 4: After pressed the button

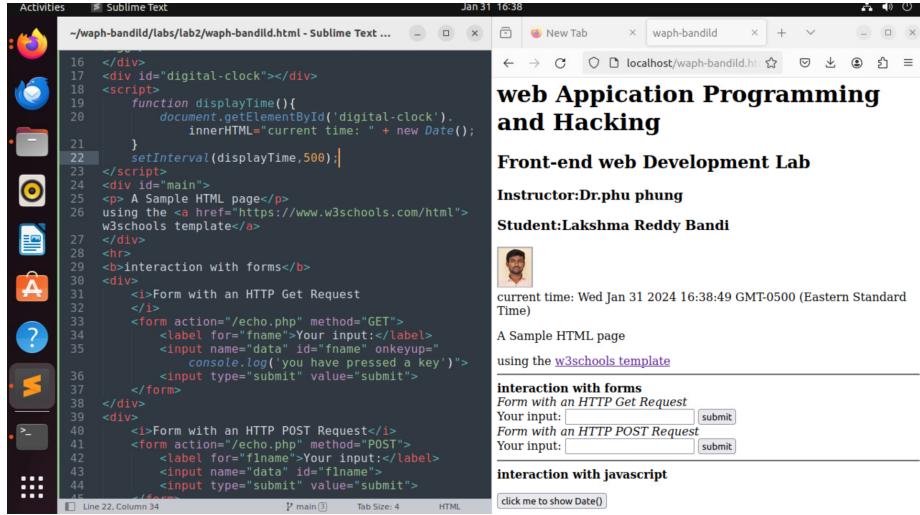


Figure 5: Simple HTML webpage with headshot

```

function showhideEmail()
{
    if (shown)
    {
        document.getElementById('email').innerHTML = "Show my email";
        shown = false;
    }
    else
    {
        var myemail = "<a href='mailto:madavava" + @" + "mail.uc.edu'>madavava" + @" + "mail.u
        document.getElementById('email').innerHTML = myemail;
        shown = true;
    }
}

```

The 2D analog clock was created using external JS code and the provided data.

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

Task-2 involved basic Ajax Echo, using HTML forms, CSS, jQuery code, and Web API integration.

a. Ajax Task-2 added a new HTML template with a form, button, and user input element, using echo.php from lab1 as a back-end file for display.

In the .html file created a new button in the so after getting input from user, need to click new button; submit, displays the user input.And logged in keypressed.

The screenshot shows a Linux desktop environment with a window manager like Unity. On the left, there's a dock with icons for various applications. In the center, there's a terminal window and two windows from the Unity Dash.

Sublime Text Window:

```

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

Browser Window:

Address bar: `localhost/waph-bandild.html`

web Application Programming and Hacking

Front-end web Development Lab

Instructor: Dr. phu phung

Student: Lakshma Reddy Bandi

current time: Wed Jan 31 2024 17:13:51 GMT-0500 (Eastern Standard Time)
`bandild@ucmail.uc.edu`

A Sample HTML page
using the [w3schools template](#)

interaction with forms
Form with an HTTP Get Request
Your input: submit

interaction with javascript

Figure 6: email hide/show on clicked

The screenshot shows a Linux desktop environment with a window manager like Unity. On the left, there's a dock with icons for various applications. In the center, there's a terminal window and two windows from the Unity Dash.

Sublime Text Window:

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>waph-bandild</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:Lakshma Reddy Bandi</h3>
15      
16    </div>
17    <div id="digital-clock"></div>
18    <script>
19      function displayTime(){
20        document.getElementById('digital-clock').
innerHTML="current time: " + new Date();
21      }
22      setInterval(displayTime,500);
23    </script>
24    <canvas id="analog-clock" width="150" height="150"
style="background-color:#999"></canvas>
25    <script src="https://waph-uc.github.io/clock.js"></script>
26    <script>
27      var canvas = document.getElementById("analog-clock");
28      var ctx=canvas.getContext("2d");
29    </script>

```

Browser Window:

Address bar: `localhost/waph-bandild.html`

Student: Lakshma Reddy Bandi

current time: Wed Jan 31 2024 17:50:10 GMT-0500 (Eastern Standard Time)

`bandild@ucmail.uc.edu`

A Sample HTML page
using the [w3schools template](#)

interaction with forms
Form with an HTTP Get Request
Your input: submit

interaction with javascript

Figure 7: Analog clock 2D developed with external javascript

The screenshot displays a Linux desktop environment with two Firefox browser windows open. The top window shows a Sublime Text editor with an HTML file titled 'waph-bandild.html'. The code includes sections for 'interaction with forms' (using GET and POST methods) and 'interaction with javascript' (including an Ajax request to echo.php). The bottom window shows the rendered HTML page. It features a clock, a link to 'Show my email', and a form for interacting with a database. The developer tools network tab is visible, showing an incoming GET request for 'echo.php?data=reddy' and its response.

Created a text box for user input, included action=""/echo.php" from lab1, and sent an Ajax GET request with ajax code within the HTML for tag.

The ajax code successfully executes an HTTP request, displaying the response in the console, and then a POST request is observed in the network, displaying the response on the web page.

The developer tools network tab showed successful Ajax HTTP requests sent to the server, with a simple echo statement recording the response.

The screenshot shows a Linux desktop environment with a dock containing various icons. A Firefox browser window is open, displaying a sample HTML page titled "web Application Programming and Hacking". The page includes a heading, a photo of a person, and a form section. Below the browser is a Sublime Text editor window showing the same HTML code. The terminal at the bottom has a command related to "waph-bandild".

```

Activities   Firefox Web Browser
Jan 31 13:40
~/waph-bandild/labs/lab2/waph-bandild.html - Sublime Text ...
- □ x
=50">
</div>
<div id="main">
<p> A Sample HTML page</p>
using the <a href="https://www.w3schools.com/html">
w3schools template</a>
</div>
<b>Interaction with forms</b>
<div>
<i>Form with an HTTP Get Request
</i>
<form action="/echo.php" method="GET">
<label for="fname">Your input:</label>
<input name="data" id="fname">
<input type="submit" value="submit">
</form>
</div>
</head>
</html>

```

Figure 8: GET request

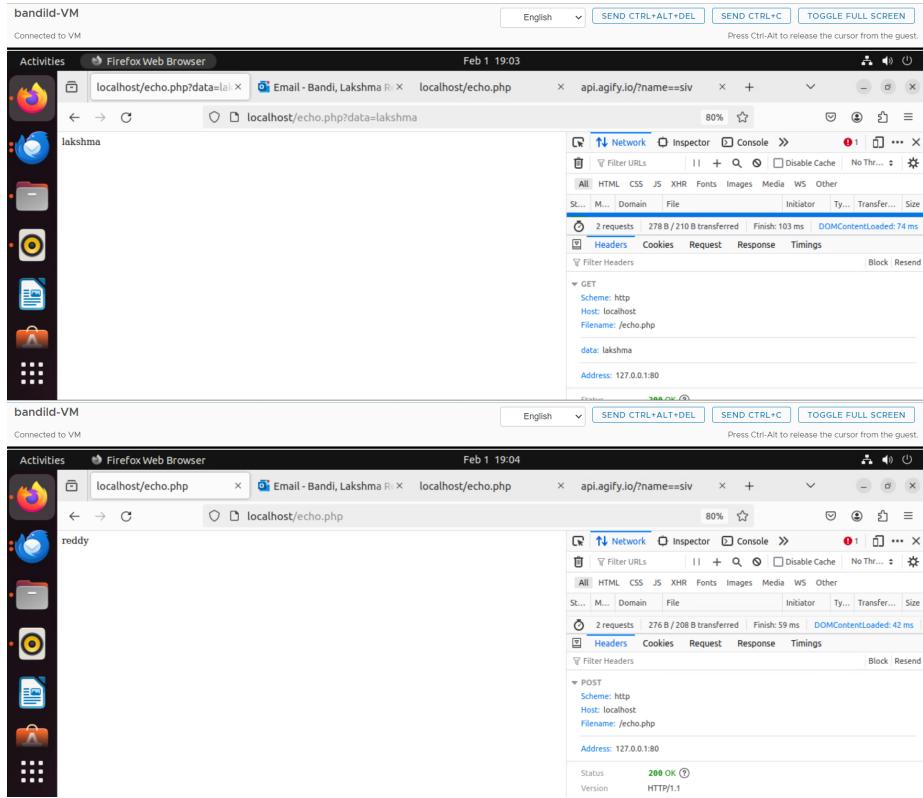
The screenshot shows a Linux desktop environment with a dock containing various icons. A Firefox browser window is open, displaying a sample HTML page titled "web Application Programming and Hacking". The page includes a heading, a photo of a person, and a form section. Below the browser is a Sublime Text editor window showing the same HTML code, but with additional code for a POST form. The terminal at the bottom has a command related to "waph-bandild".

```

Activities   Firefox Web Browser
Jan 31 13:48
~/waph-bandild/labs/lab2/waph-bandild.html - Sublime Text ...
- □ x
=50">
</div>
<div id="main">
<p> A Sample HTML page</p>
using the <a href="https://www.w3schools.com/html">
w3schools template</a>
</div>
<b>Interaction with forms</b>
<div>
<i>Form with an HTTP Get Request
</i>
<form action="/echo.php" method="GET">
<label for="fname">Your input:</label>
<input name="data" id="fname">
<input type="submit" value="submit">
</form>
<div>
<i>Form with an HTTP POST Request</i>
<form action="/echo.php" method="POST">
<label for="fname">Your input:</label>
<input name="data" id="fname">
<input type="submit" value="submit">
</form>
</div>
</div>
</head>
</html>

```

Figure 9: POST request



b. CSS The CSS was applied inline with the style attribute set to blue.

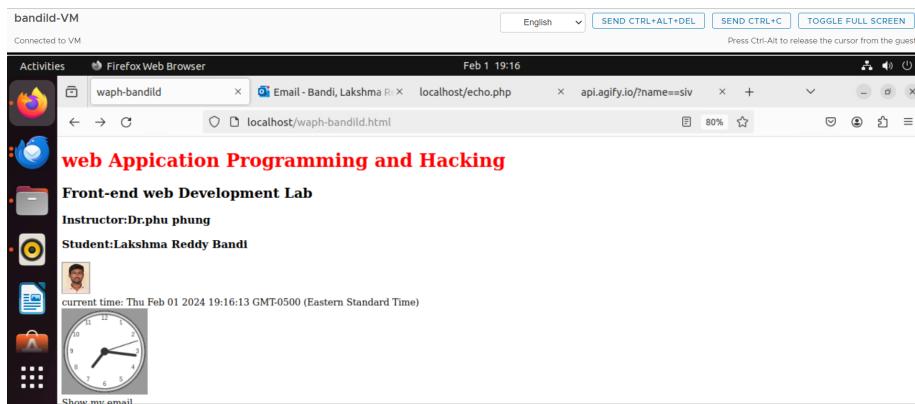


Figure 10: inline css

For internal css the css for button class, round class, and response id elements were added, within the style tag in the head part of waph-bandild.html file.

```

css
<style>
  body{background-color: red;}
  h1 {color: blue;}
</style>

```

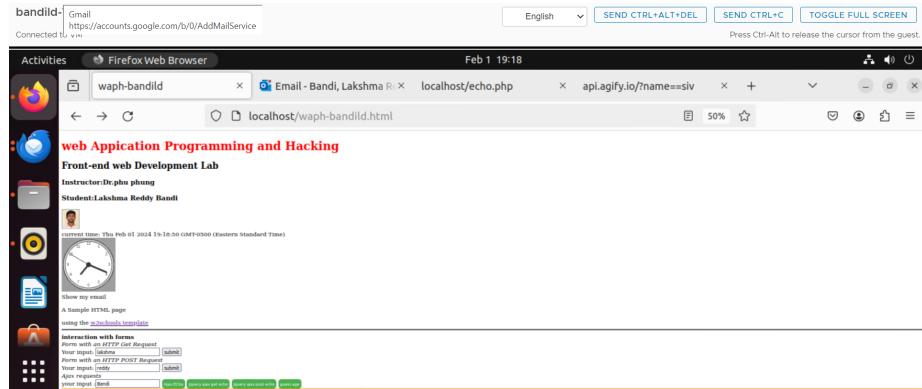


Figure 11: internal css

Then linked an external css template number 2, with link rel="stylesheet" src="https://waph-uc.github.io/style.css">

```

css
<link rel="stylesheet" href="https://waph-uc.github.io/style1.css">

```

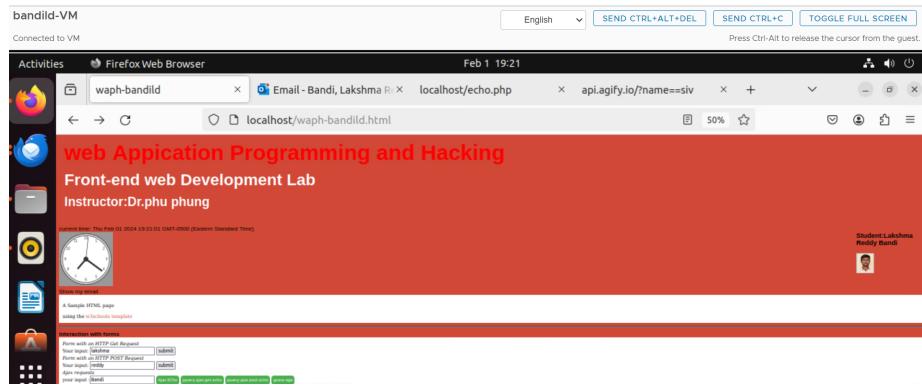


Figure 12: external css

c. jQuery The jQuery library file was installed and executed as per the provided instructions.

- i. The user input now includes a button for sending an Ajax GET request to echo.php with user input, using jQuery \$.get() in the getjQueryAjax function.

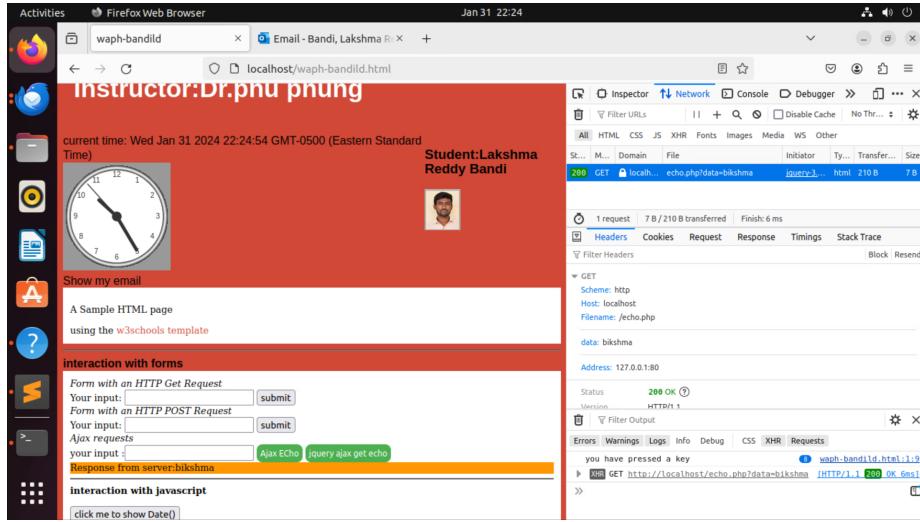


Figure 13: jQuer ajax GET

ii. A button was added to send an Ajax POST request with jQuery to echo.php, displaying the response in the div element and confirming the same response in the network tab.

d. Web API integration **i.** The final task involved integrating an API for displaying a joke in the div element in HTML, using the provided API link. script tag. [https://v2.jokeapi.dev/joke/Programming?type=single\]](https://v2.jokeapi.dev/joke/Programming?type=single)

The jQuery Ajax code is used to send a request to display the response of the HTTP request to a random joke on the webpage.

ii. The integration of the API for guess age with fetch in the jQuery Ajax code with API is being considered. - <https://api.agify.io/?name=input>

The jQuery ajax function, guessAge, is used to display the results of an API method when a user enters an input. The browser network tab is used to view the requests and responses.

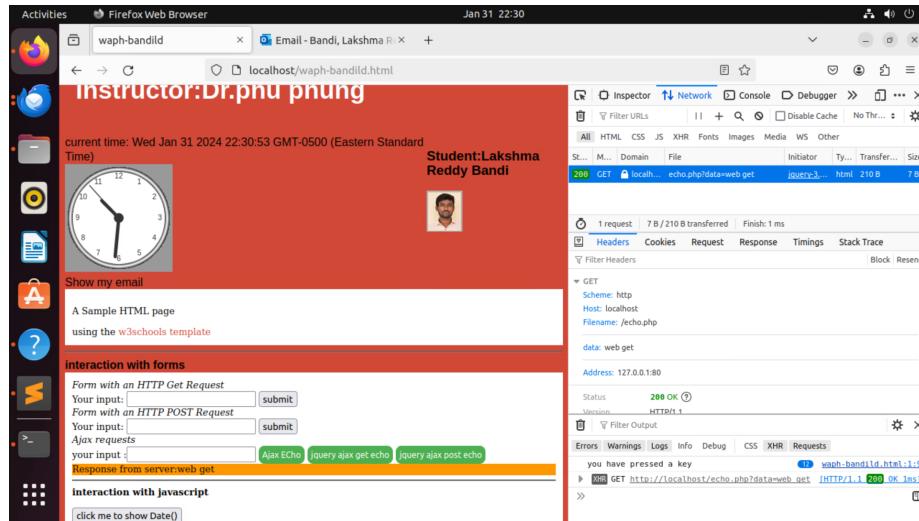


Figure 14: jQuery POST

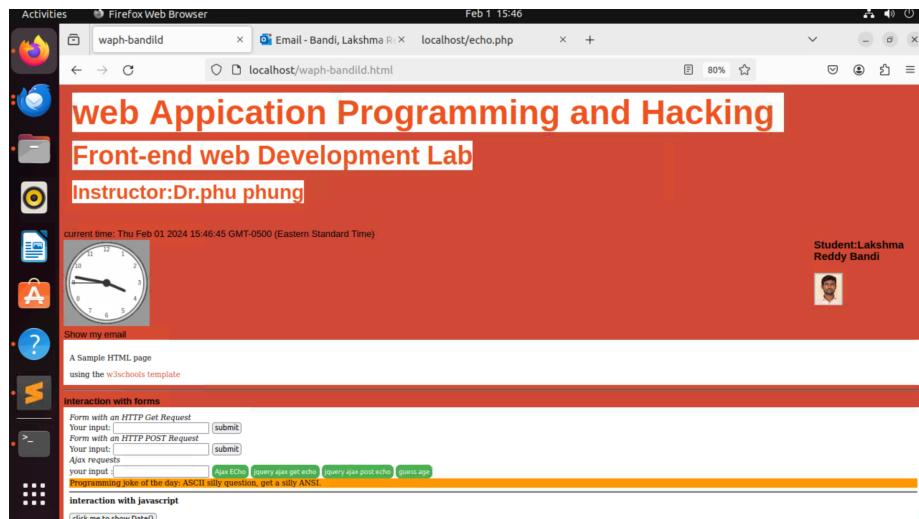


Figure 15: joke api

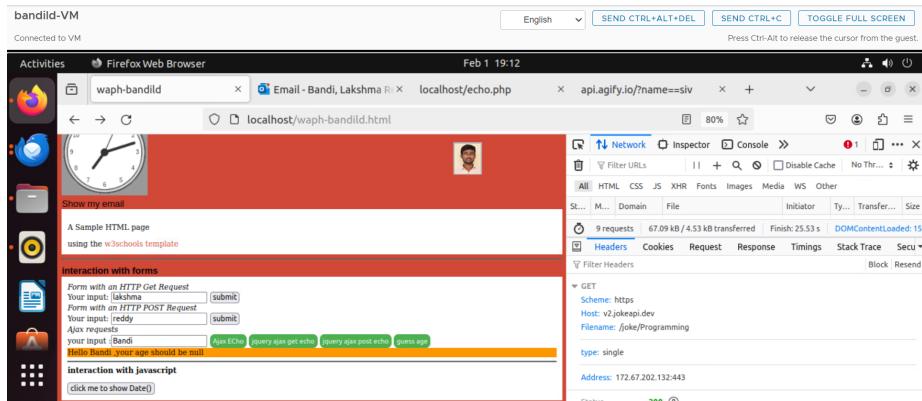


Figure 16: fetch agify

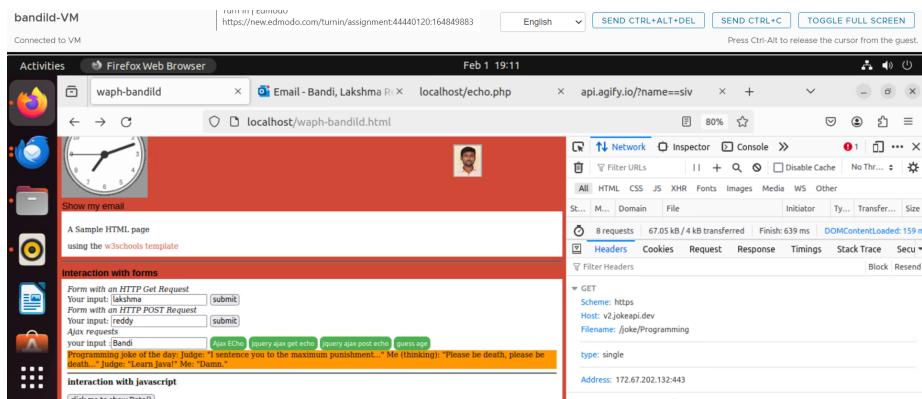


Figure 17: network server for api