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-bandi.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:

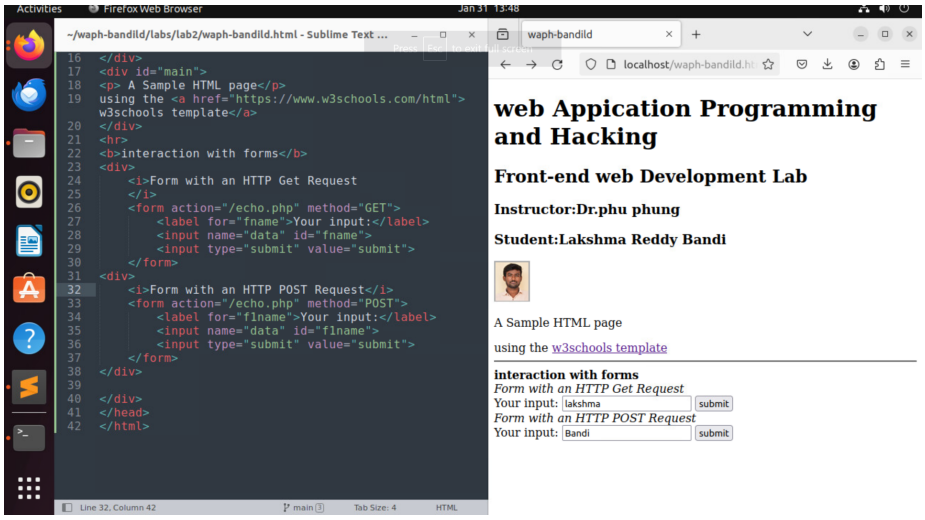


Figure 2: Simple HTML webpage with headshot

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
</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;
```

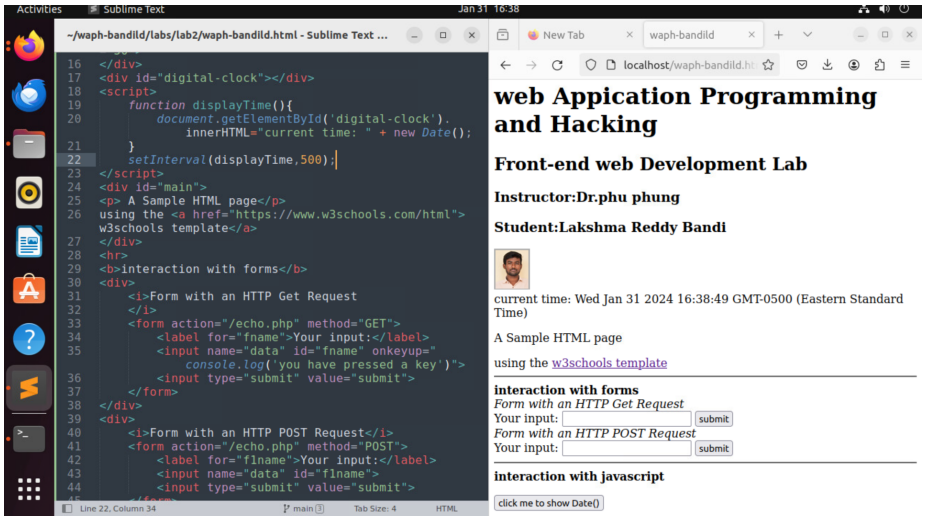


Figure 3: In line javascript for displaying data/time after key pressed

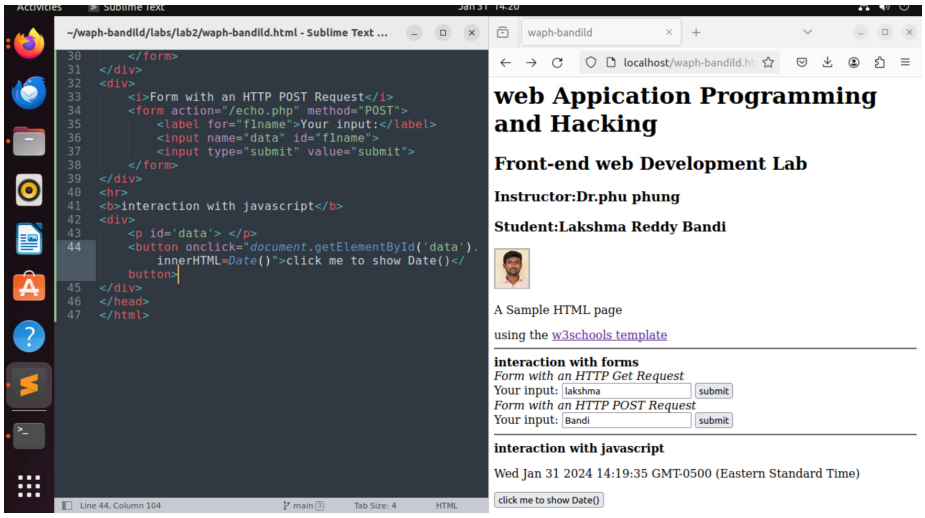


Figure 4: After pressed the button

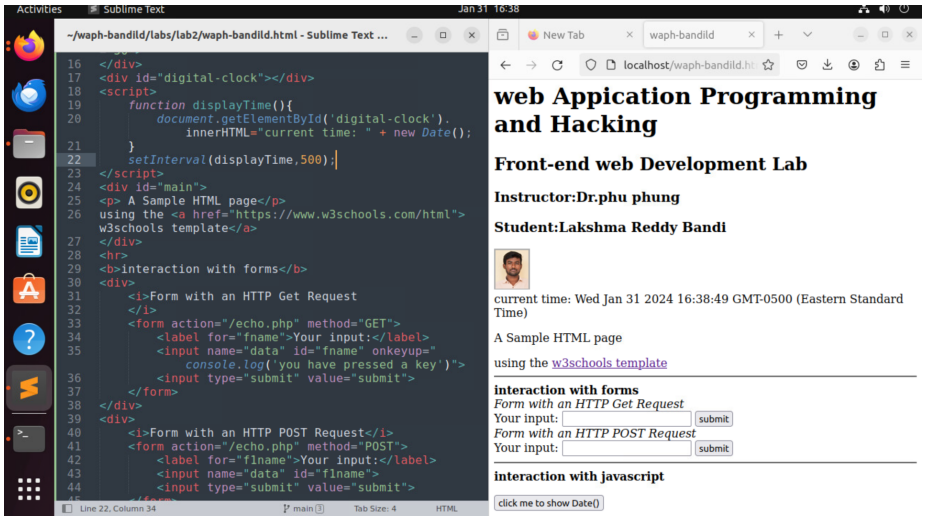


Figure 5: Simple HTML webpage with headshot

```
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.

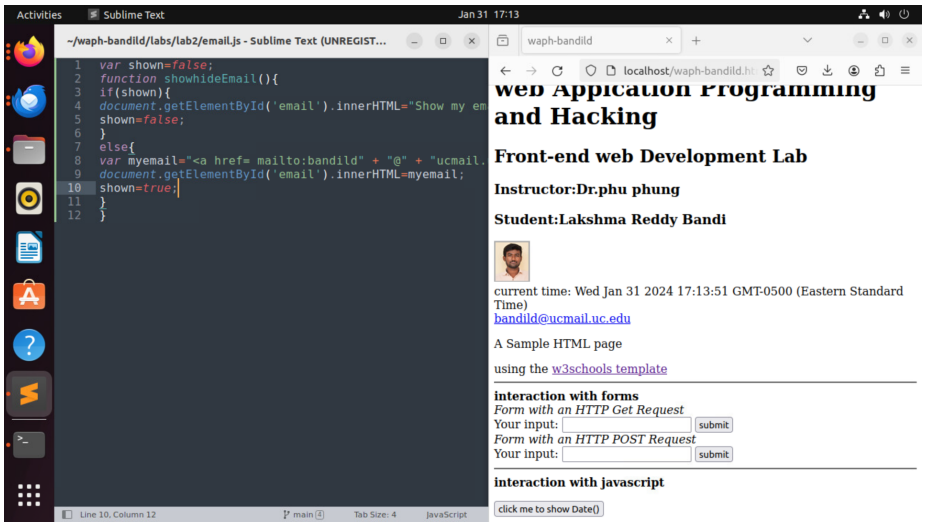


Figure 6: email hide/show on clicked

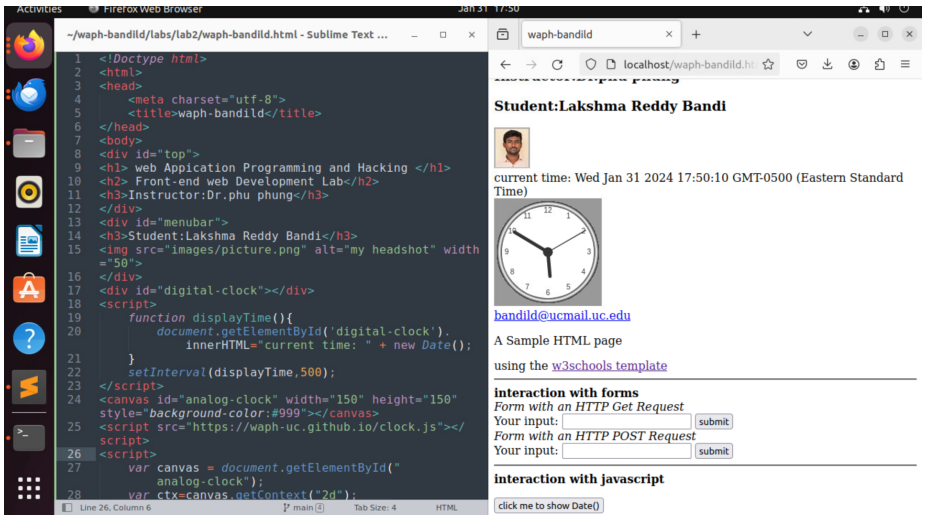
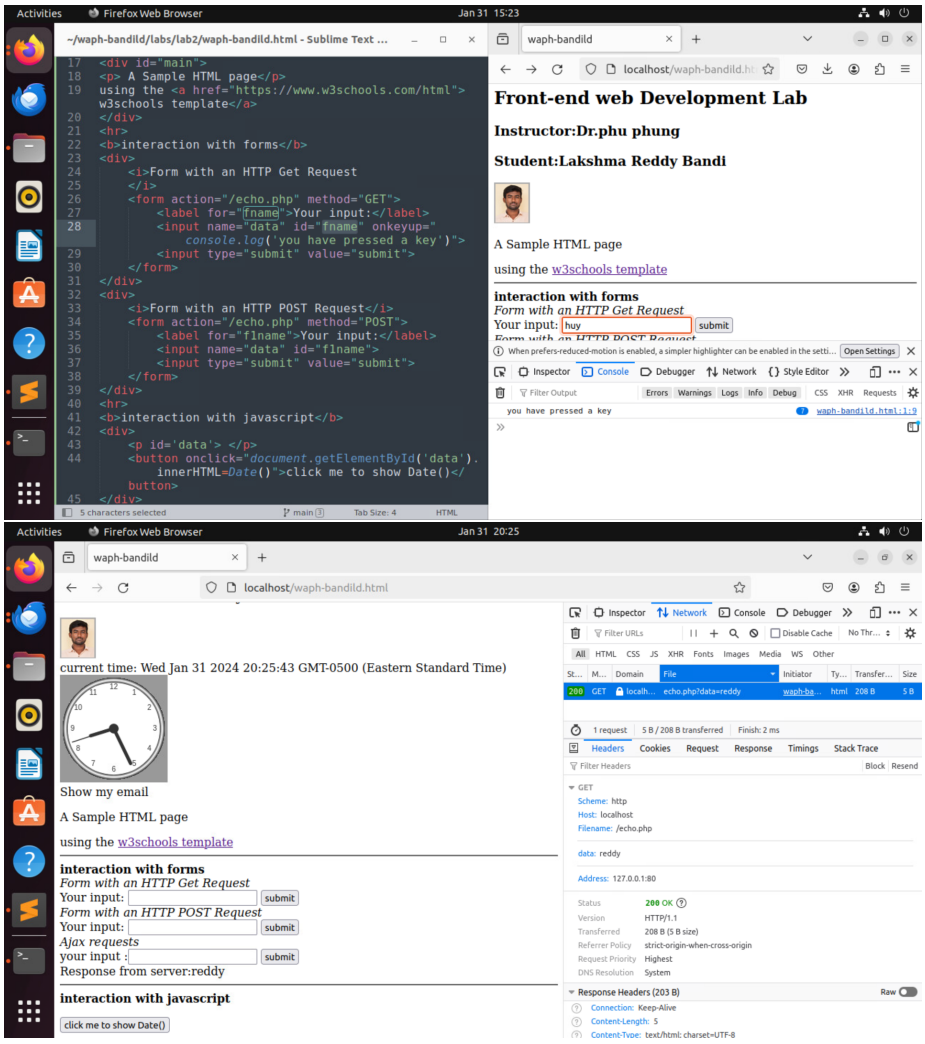


Figure 7: Analog clock 2D developed with external javascript



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.

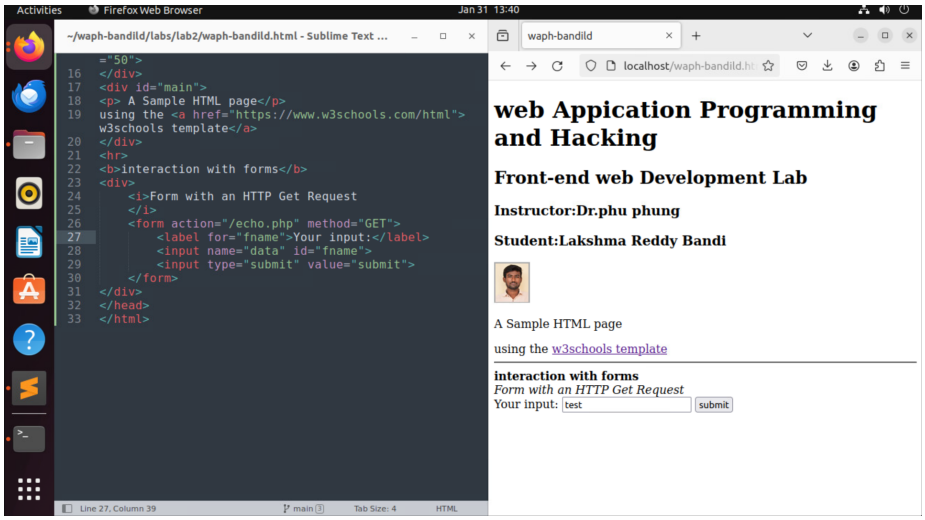


Figure 8: GET request

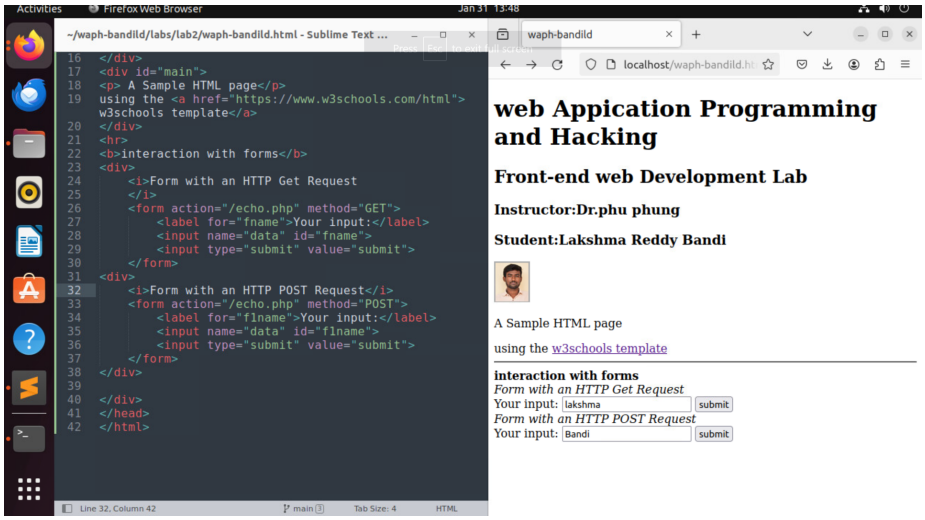
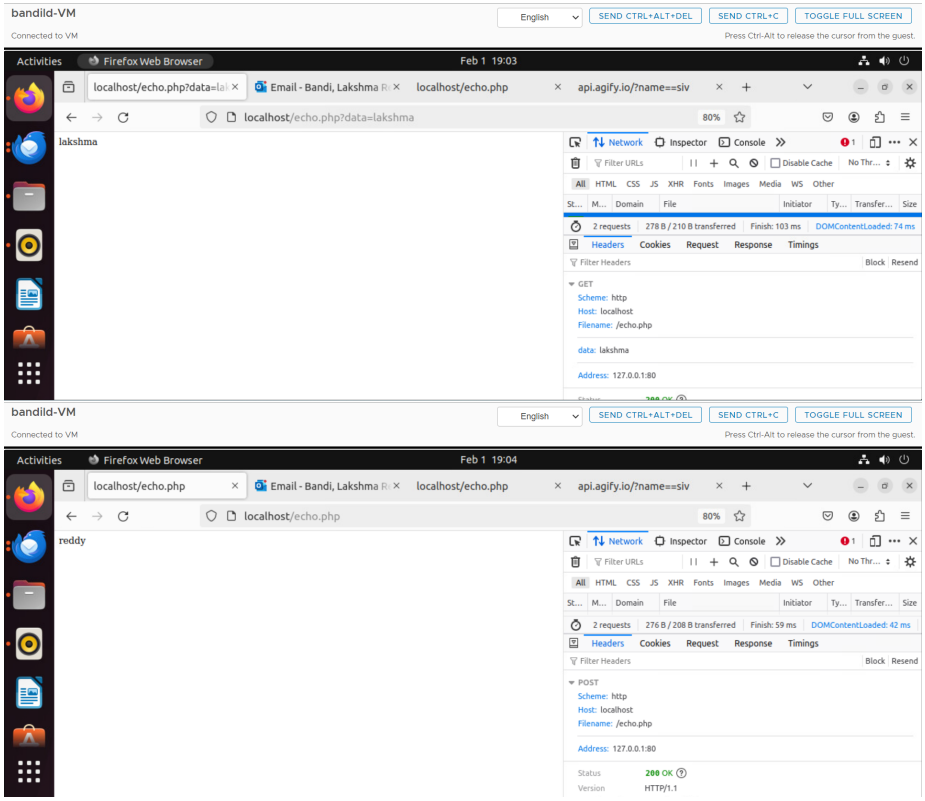


Figure 9: POST request



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

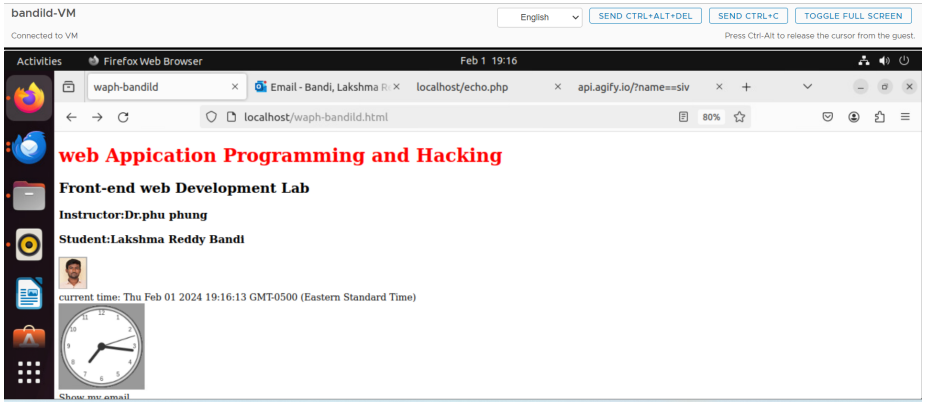


Figure 10: inline css

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>
```

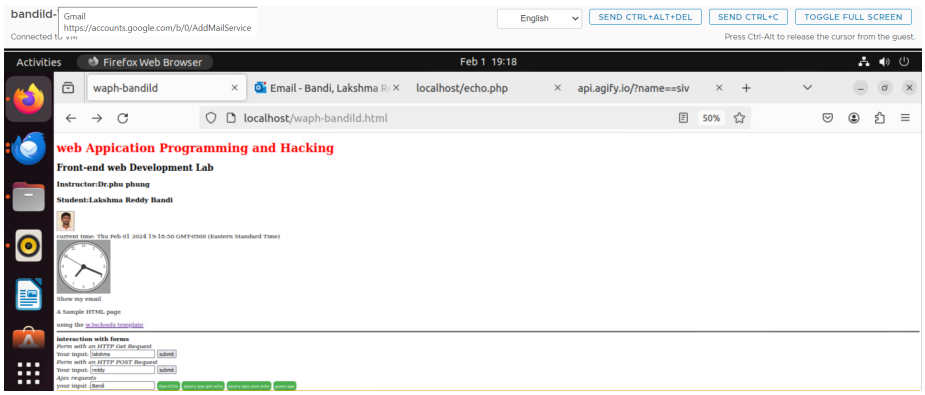


Figure 11: internal css

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

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

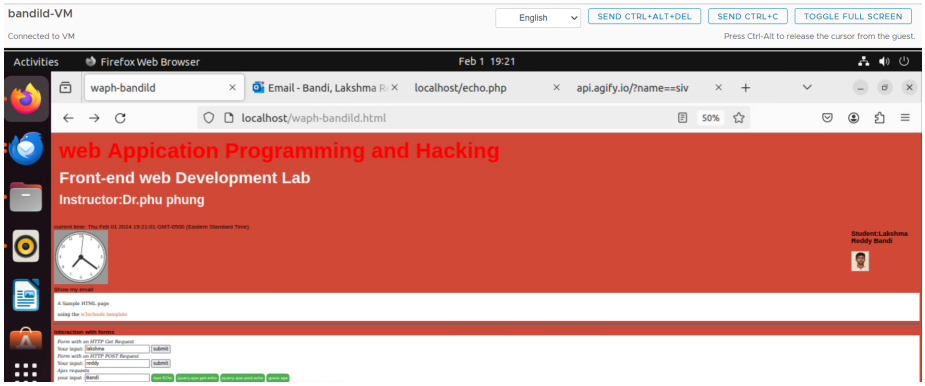


Figure 12: external css

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 iQuery $.get() in the getjQueryAjax function.

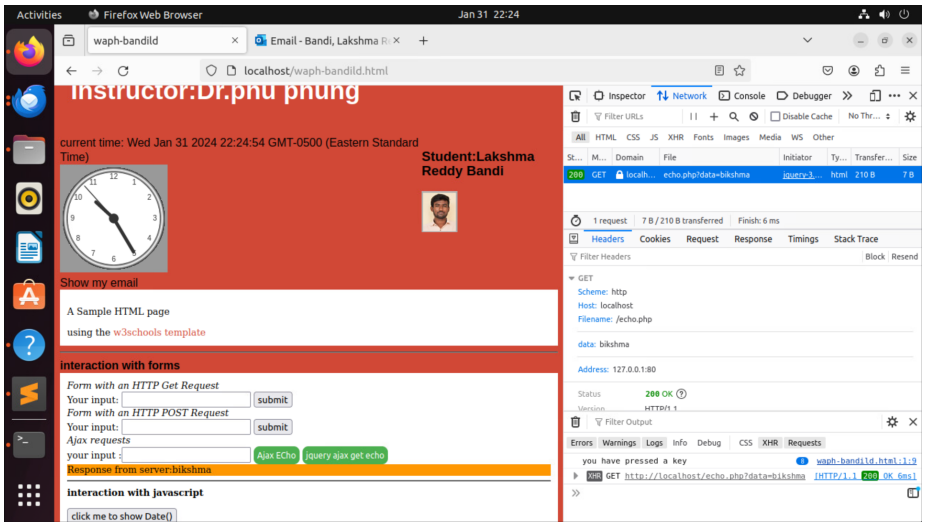


Figure 13: jQuer ajax GET

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 dsplaying a joke in the div element in HTML, using the provided API link. script tag. 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.

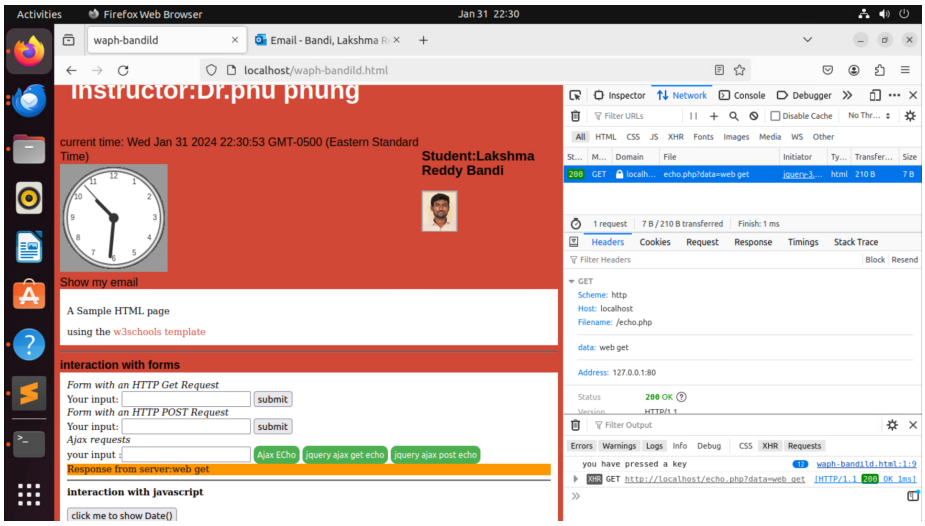


Figure 14: jQuery POST

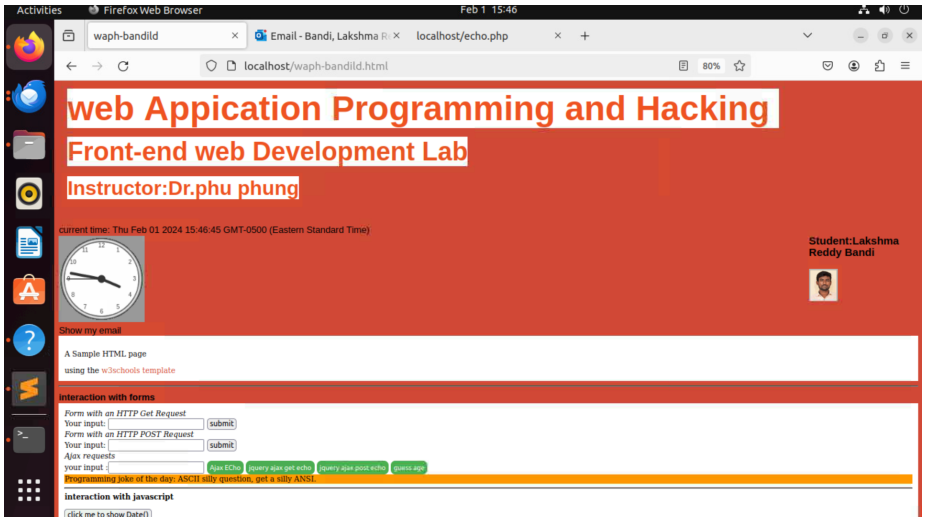


Figure 15: joke api

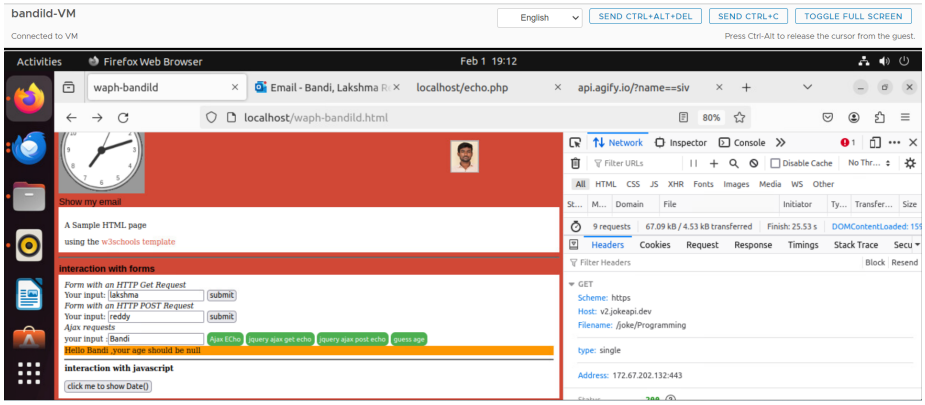


Figure 16: fetch agify

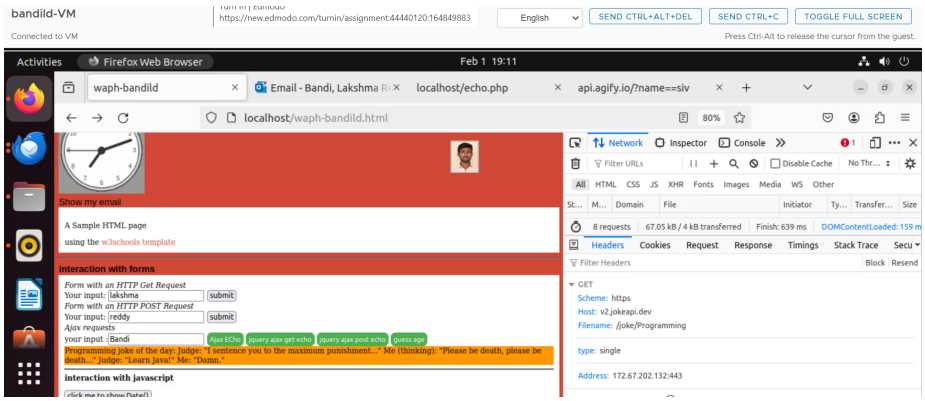


Figure 17: network server for api