WAPH-Web Application Programming :computer: and Hacking :man_technologist:

Instructor: Dr. Phu Phung

Lab 2 - Front-end Web Development

Student :book:

Name: Vishal Kothapalli :man_technologist:

Email: kothapvl@mail.uc.edu

Short-bio: Graduate Student at UC

:label: Lab2 Information

Lab2's URL: [https://github.com/kothapvl-uc/waph-kothapvl/tree/main/labs/lab2]

Overview

Lab 2 features HTML for structure, simple JavaScript for interactivity, Ajax for asynchronous data exchange, CSS for styling, jQuery for streamlined scripting, and Web API integration for external service interaction. The combined implementation results in a responsive, visually appealing web application.

Task 1: Basic HTML with forms, and JavaScript

a. HTML (5 pts) I have created a straightforward HTML file incorporating fundamental tags, including an image of my headshot, and integrating a form.

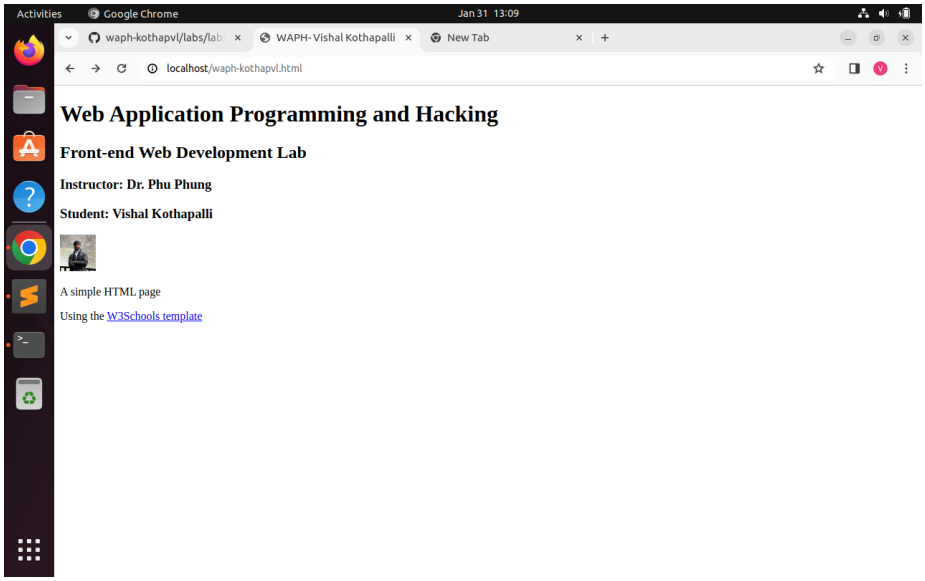

Image 1: Basic HTML

b. Simple JavaScript (15 pts) The HTML page incorporates inline JavaScript to display the current date/time and log key presses when clicked. Additionally, a

tag houses JavaScript code for a digital clock. Furthermore, external JavaScript, coupled with HTML, is used to toggle the visibility of an email address upon clicking. Lastly, an analog clock is displayed using external JavaScript code integrated into the HTML page. These steps enhance interactivity, provide time-related information, and control the visibility of specific content based on user interactions.

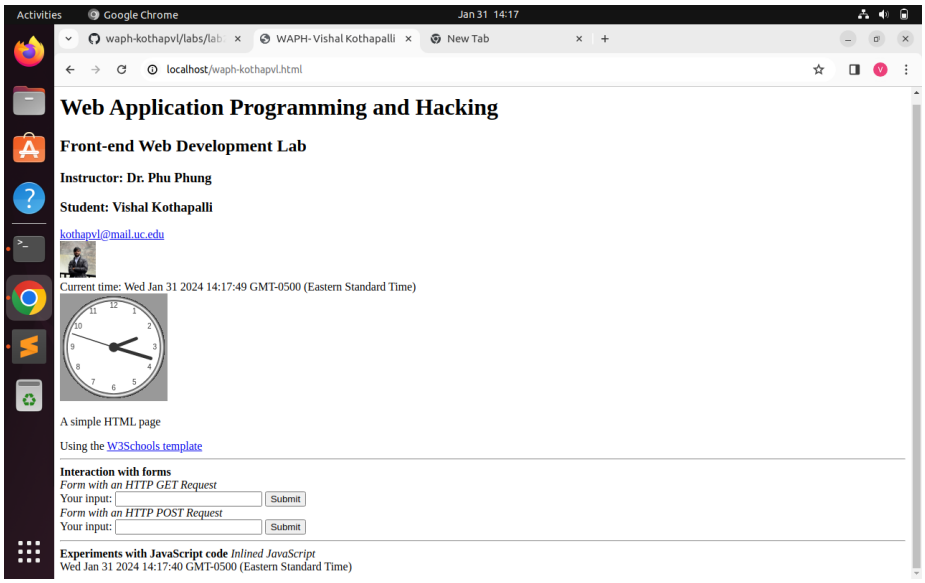

Image 2: Simple JavaScript

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

a. Ajax (7.5 pts) The HTML page is enhanced with a user input field, a button, and a div element. Upon clicking the new button, JavaScript retrieves the user input, initiates an Ajax GET request to the echo.php web application from Lab 1, and dynamically displays the response in the designated div. To understand the Ajax request/response process, inspecting browser network connections provides insights into the underlying interactions between the client and the web application, showcasing the asynchronous data exchange mechanism.

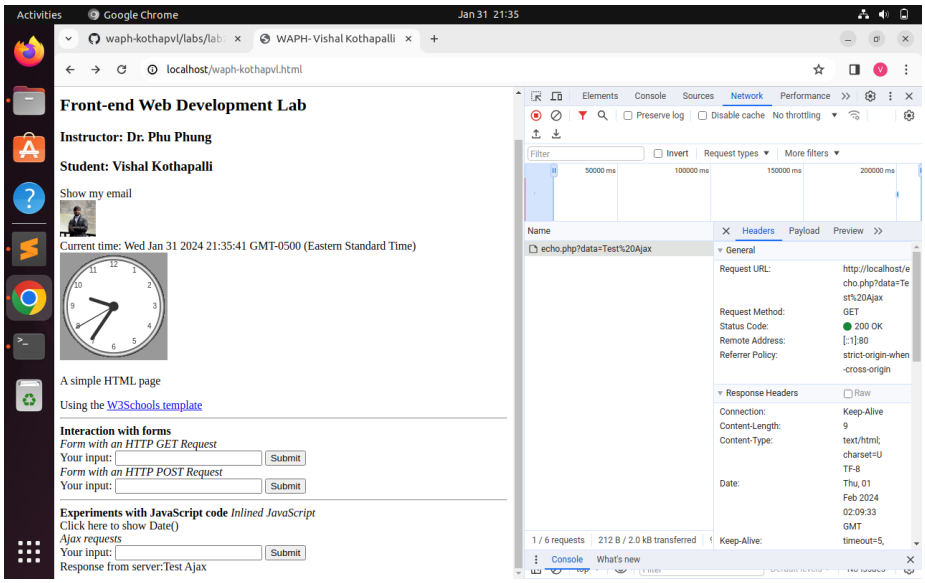

Image 3: Ajax implementation

b. CSS (7.5 pts) The HTML page is enriched with CSS styling through inline, internal, and external methods. Inline CSS is directly applied within HTML tags, internal CSS is housed within a

tag in the HTML file, and external CSS is linked through a remote stylesheet. These diverse CSS approaches collectively contribute to the visual presentation and formatting of the web page, offering flexibility and maintainability in styling implementations.

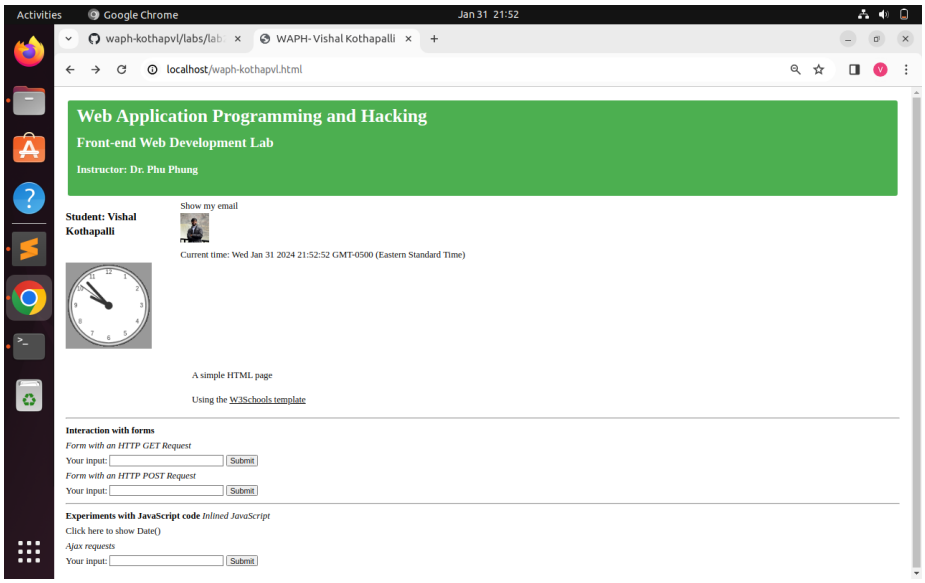

Image 4: Using CSS style hosted publicly

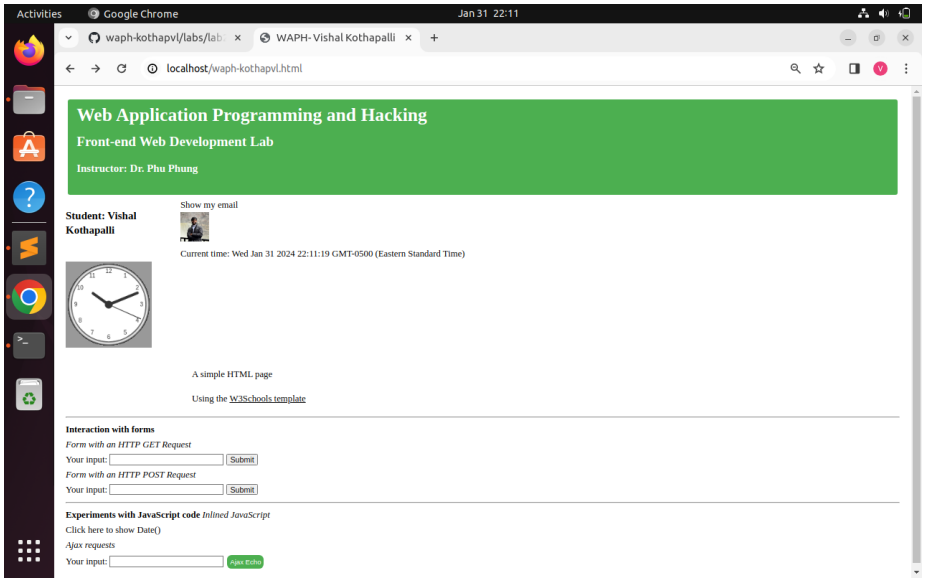

Image 5: CSS Style

c. jQuery (5 pts) **i.** The jQuery library is incorporated into the HTML page, enabling the implementation of HTML and JavaScript code in jQuery. Upon clicking a designated button, jQuery facilitates the initiation of an Ajax GET request to the echo.php web application. The retrieved response content is then

dynamically displayed, showcasing the streamlined and concise nature of jQuery in handling asynchronous requests and enhancing user interactions.

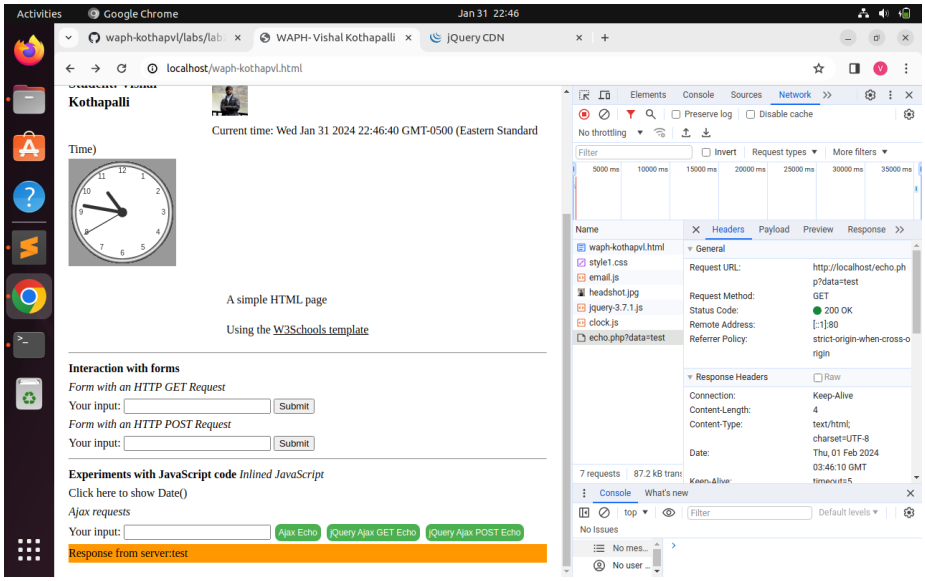

Image 6: jQuery GET

ii. Upon clicking a specific button, an Ajax POST request is triggered through jQuery, directed towards the echo.php web application. The resulting response content is dynamically displayed, showcasing the streamlined and efficient handling of asynchronous requests with jQuery.

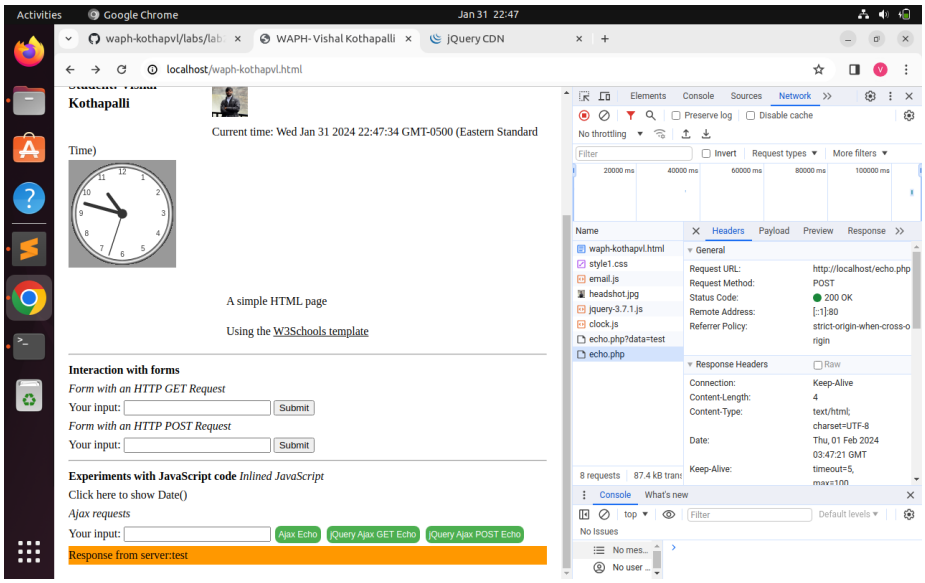

Image 7: jQuery POST

d. Web API integration (10 pts) **i.** JavaScript code with jQuery Ajax is implemented to send a request to the specified API https://v2.jokeapi.dev/joke/Programming?type=single. Upon loading the page, the code handles the response, displaying a random joke. Browser network inspection allows scrutiny of the request and response details, providing insights into the interaction with the external API and the dynamic content retrieval process.

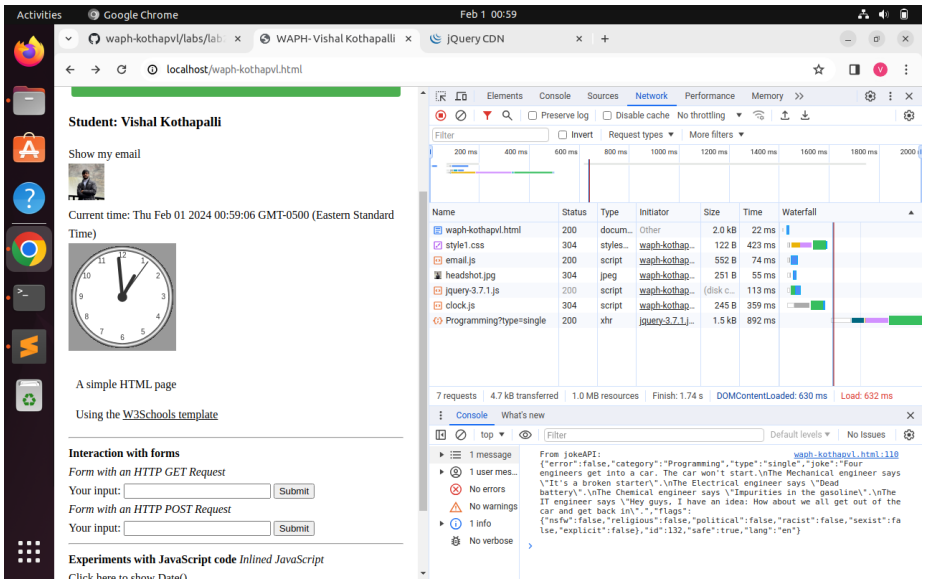

Image 8: Web API integration to get a random joke from a Programming Joke API

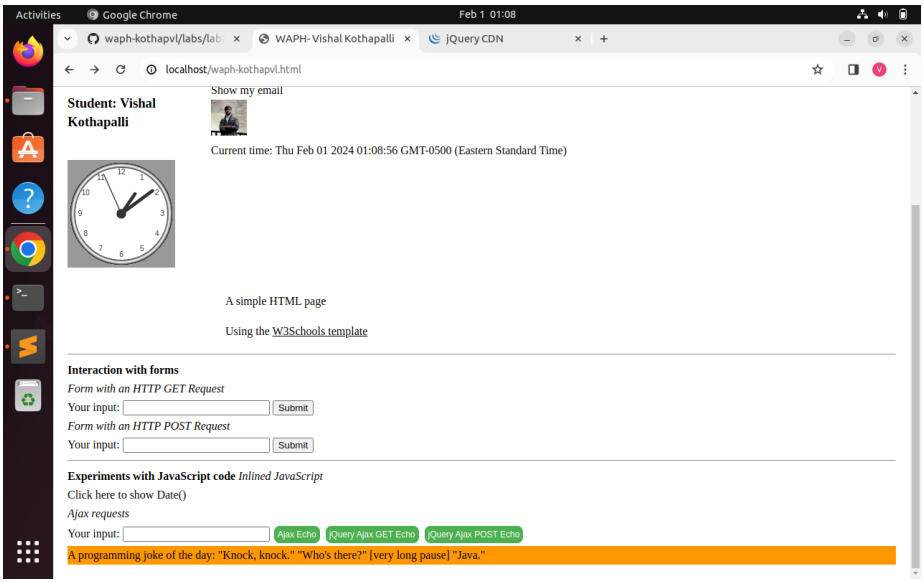

Image 9: Displaying a random joke from a Programming Joke API

ii. HTML and JavaScript code is implemented to utilize the fetch() method, calling the API https://api.agify.io/?name=input based on user input. The response results are dynamically displayed. Browser network inspection facilitates the examination of the request and response details, offering insights into the interaction with the external API and the presentation of dynamic content based on user input.

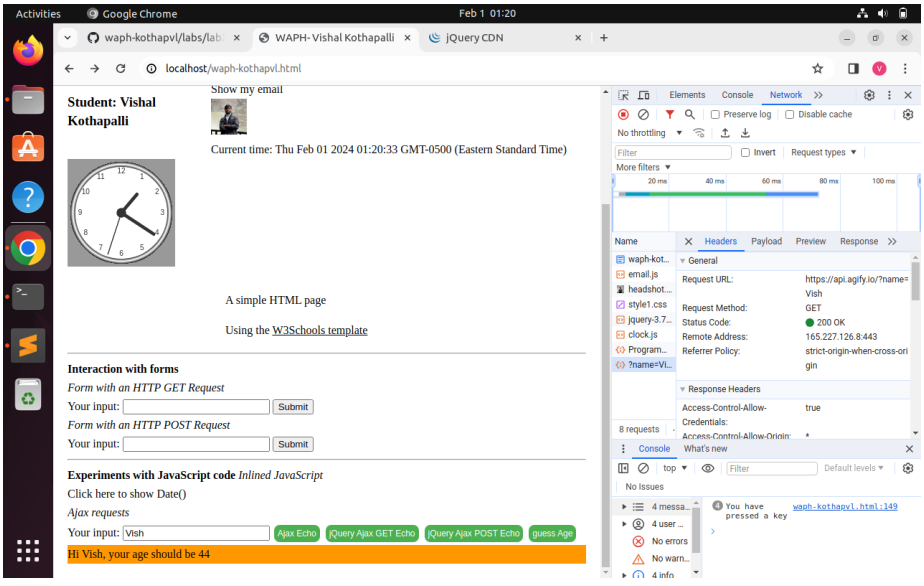

Image 10: Using Agify API to fetch age