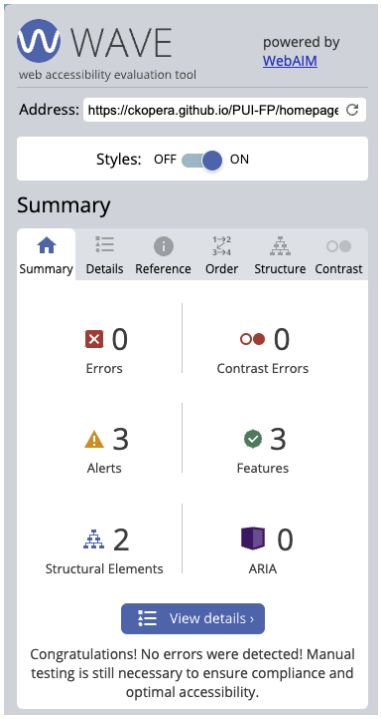
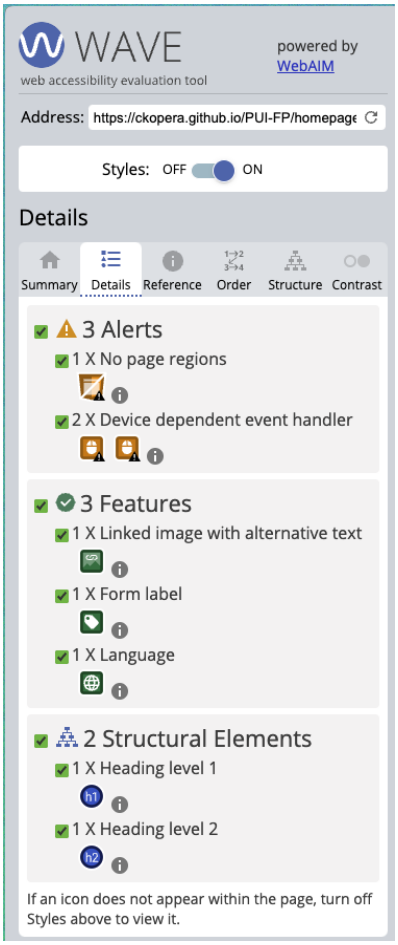
Deliverables

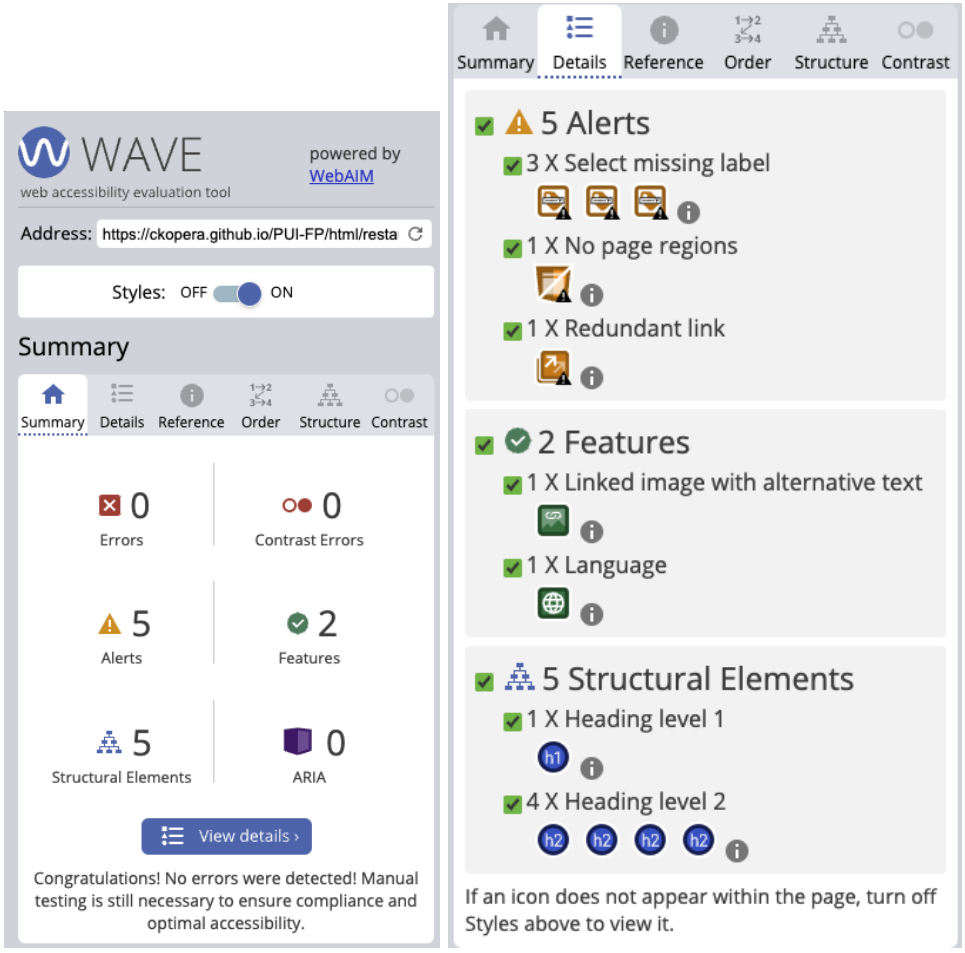
Responsiveness: The website can be viewed on a desktop screen (1920 x 1185), and on iPad (1024 x 1366).

Accessibility
Homepage:

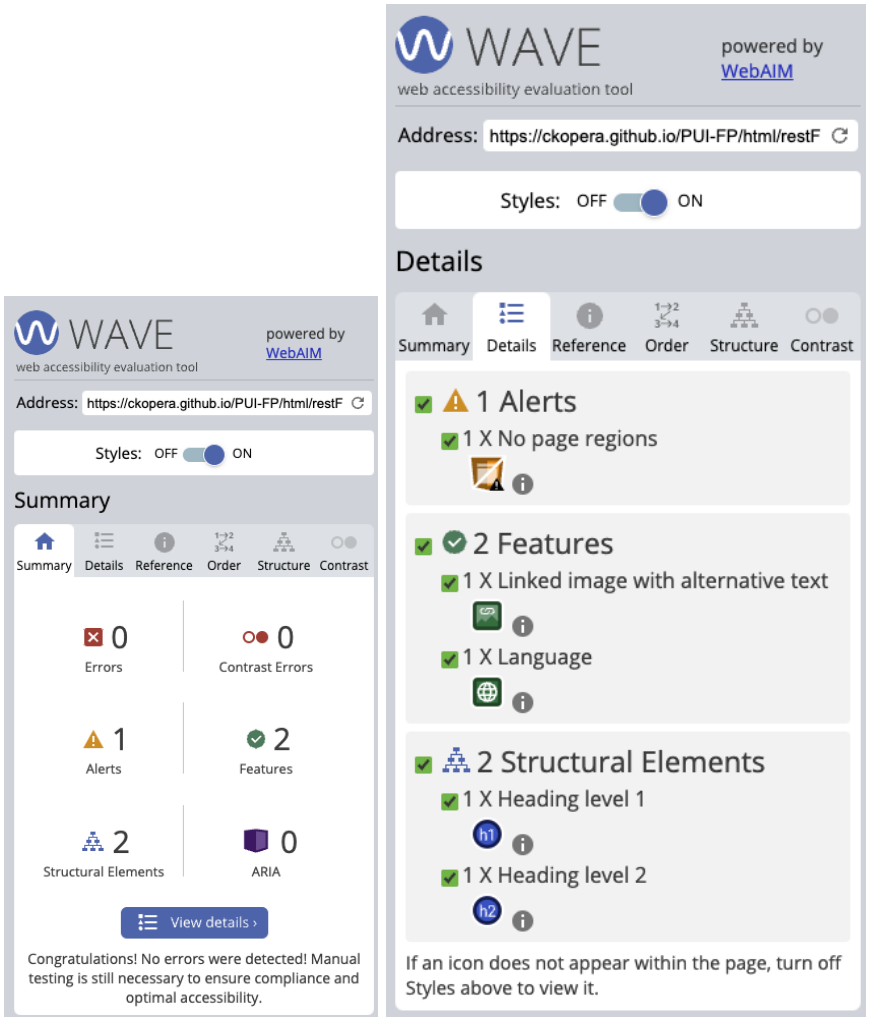




Restaurant Finder Page:



Results Page:



Write-up: Christof Kopera

To use the website, please input the API Key in the text box at the bottom of the home page

API Key: AIzaSyA1LGwHjkyqiddrzY3fjoUUXqqCFjr6hJY

Part 1

The purpose of my website, Eat Explorer, is to help CMU students answer the dreaded question "What should I eat?". My website shows the 5 nearest restaurants to CMU's Cohen University Center (CUC). This is the center spot of campus and allows for anyone on campus to accurately use my website. It is interesting since it is a useful tool people can use to genuinely help them in everyday life. It also engages the user since, if they don't even know what cuisine they'd like to eat, price point, or proximity, they can simply be assigned a random restaurant. The target audience for my website is CMU students, faculty, or anyone who is near or on CMU campus. When getting the list of returned restaurants, users can click on the restaurant items to view the restaurant's website.

Part 2

- From the homepage, the user chooses whether they'd like to find a restaurant, or receive a surprise recommendation
    - Interaction type: click
    - Reproduce: click on "Find a restaurant" text or "surprise me" text on the homepage.
    - Users need to include the API key in the "API Key" text box at the bottom of the page.

Now, two flows begin, so I will outline each flow separately. Flow 1: Find a restaurant

- Find a restaurant
    - Users are brought to a new page with 3 input parameters. Each of these parameters can be inputted through a dropdown menu: Cuisine, Price, and Proximity. Once the three parameters are filled, users click "go".
        - Interaction: click, hover, dropdown menu
        - Reproduce: click the dropdown menu under each parameter, and click on a specific input, then click "go"
- Restaurant results
    - Users are brought to a scrollable list of 5 restaurants that correspond to their parameters. On each of the restaurant items, restaurant name, address, price point (if applicable), average reviews, and total reviews show up.. Users can click on a restaurant item to navigate to the restaurant's website or menu
        - Interaction: click, hover, link
        - Reproduce: view on any generated restaurant item and click on them to visit the restaurant's website.

Flow 2: Surprise me

- Users click on “surprise me” text to generate a random restaurant near CMU campus. **Please remember to input the API Key before each click on “surprise me”.**

Part 3

I used the Google Places API, specifically the Places (New) API’s Nearby Search function. I decided to use the Google Places API since it has the most expansive list of restaurants, and would be great practice to use such a detailed, useful, and industry-standard tool. This function allows API calls to return a list of restaurants and all of their data in the form of a large array. I created javascript functions to sort through each array and only pull relevant information (restaurant name, address (street number, street name, city, and state), average reviews, total reviews, and URL). I then took this information and displayed them as restaurant items on a results page, allowing users to click on each item to visit the restaurant’s website.

This API is essentially the entire functionality of my website, since it can dynamically retrieve and load restaurant information based on three parameters.

Part 4

My original design varied slightly from my final website design. Due to technical difficulties and the extremely complex nature of this API, it was very difficult for me to retrieve certain things from the API. For example, I chose not to include photos in my final design, since I could not get the photos URL to work properly and display at all. Furthermore, I wanted to tie the random restaurant feature to a zodiac sign, and have it populate dynamically based on birth date, but I could not get that to work either. I ended up deleting several HTML pages due to this, since creating the zodiac signs dynamically was not feasible for me. I also got feedback from my peers to change some color schemes to create a more “restaurant website” look, which I did.

Part 5

As mentioned above, I experienced several challenges while implementing my website. Aside from the zodiac and photos challenges, I had a difficult time in implementing local storage onto two html pages, and saving user inputs from the drop downs and putting them into the API call. After a lot of trial and error, though, I finally managed to do so. I also had challenges with setting up the API in the first place, and ended up having to “translate” linux code into javascript, since the API gave me linux code. I also had trouble with hiding the API key when uploading to GitHub, so I ended up just having users input the API key in a text field.