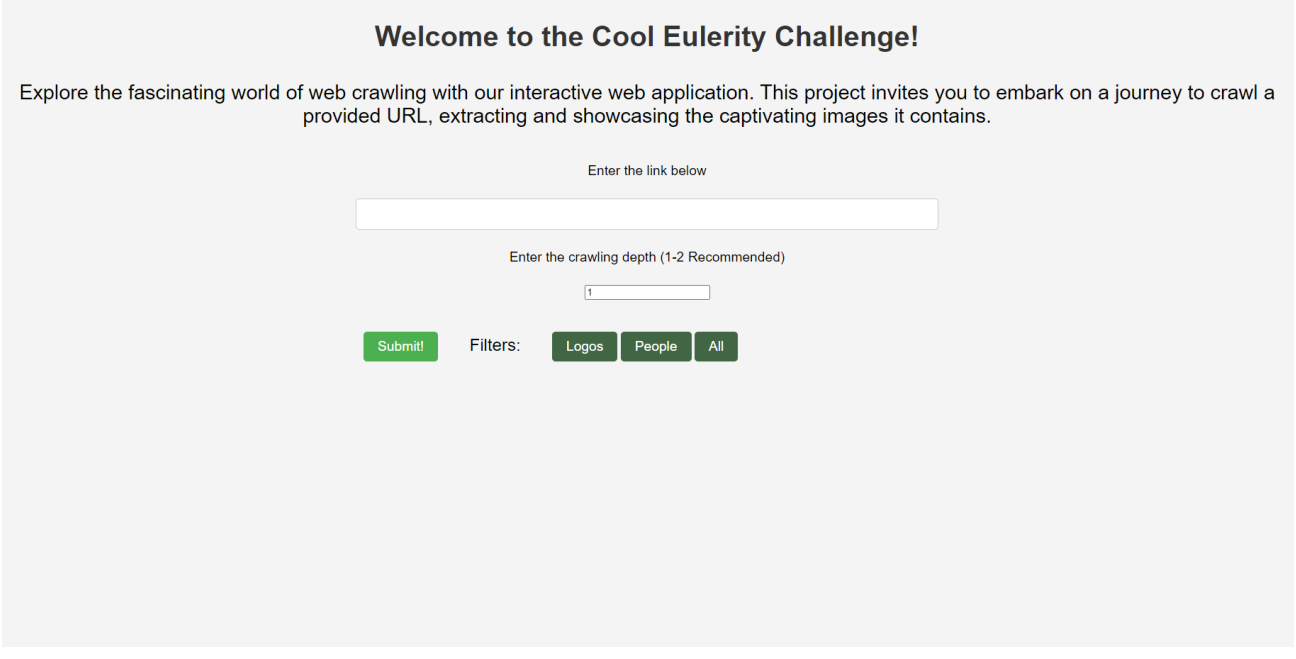
Eulerity ImageFinder Documentation

Overview: This document provides an overview of the ImageFinder tool and its usage.

Step 1: start the server by running the **mvn clean test package jetty:run** command on the terminal.

This will start the server on http://localhost:8080 which will load the home page below

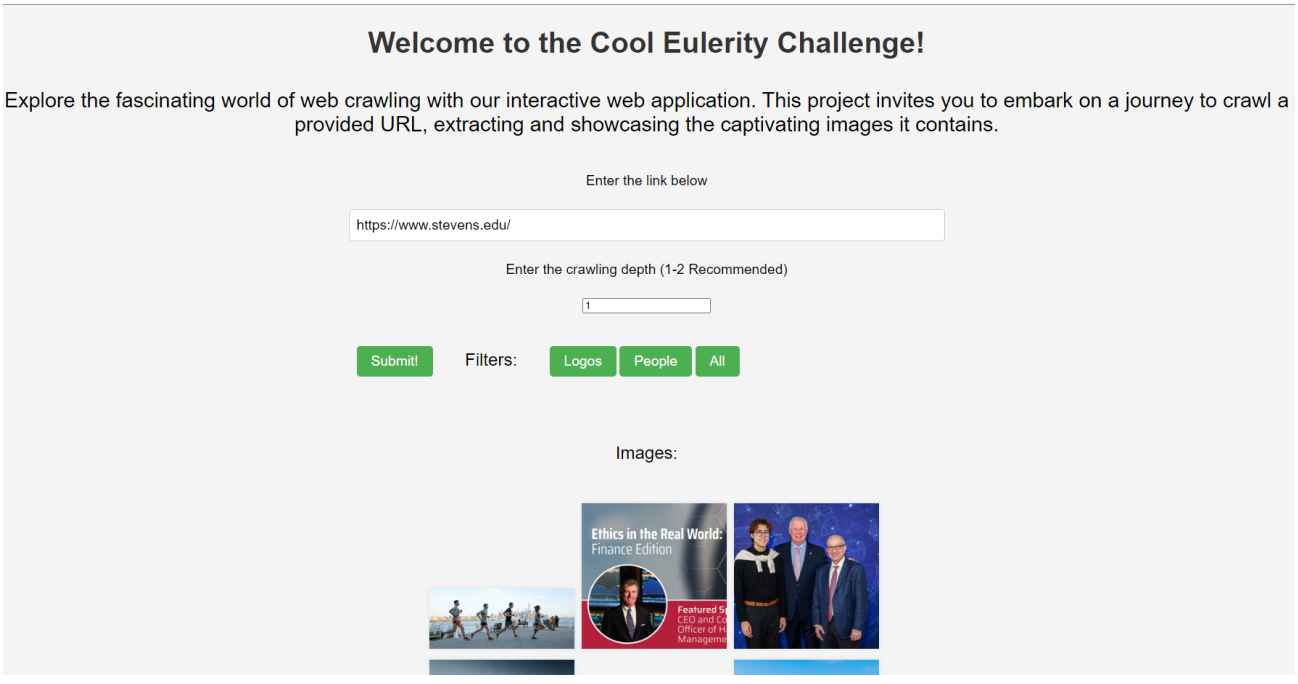


Here you can enter the URL that needs to be crawled for images.

You have the option to specify the crawling depth, which determines the extent of the web crawl. By default, the depth is set to 1, allowing the system to retrieve images solely from the current page of the provided URL. This configuration is optimal for efficient image extraction. However, if you increase the depth to 2, the system will expand its scope. In addition to crawling the initial web page, it will gather all available URL links on that page. Subsequently, the system will perform subsequent crawls on each of these URLs to extract images. It's important to note that a heavier webpage with numerous URLs may result in longer processing times as the system diligently crawls and loads all images.

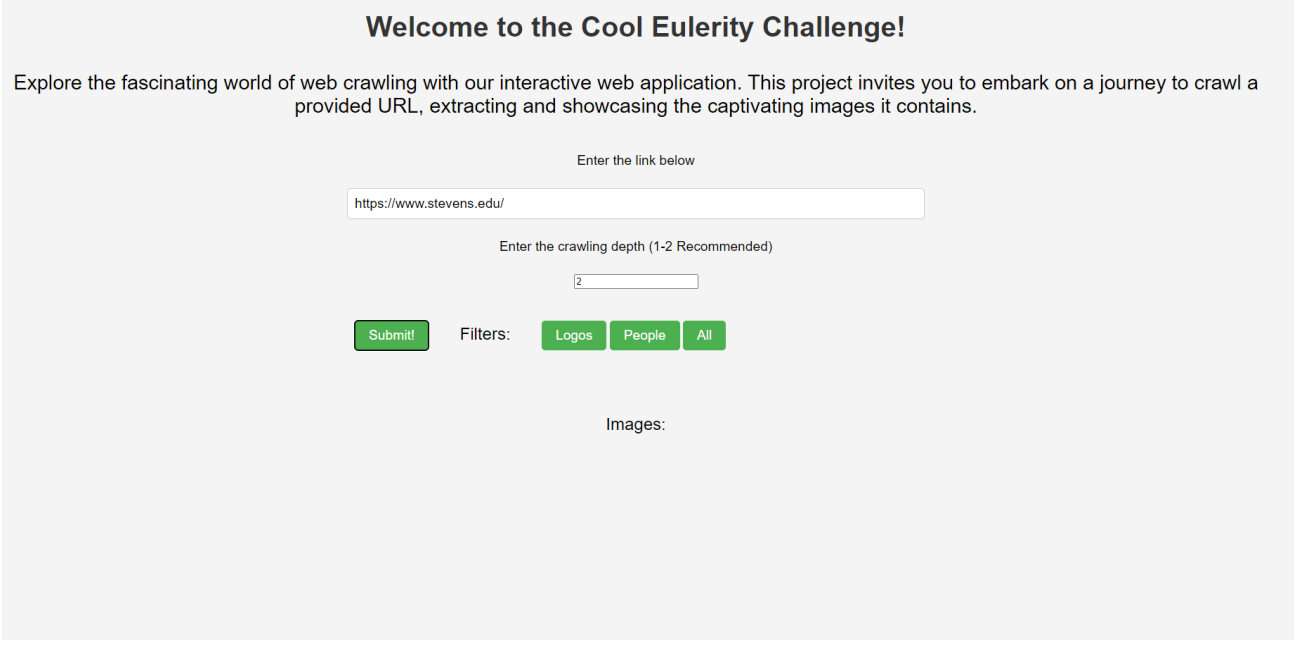
Example:

We will crawl over https://www.stevens.edu with depth = 1 first.



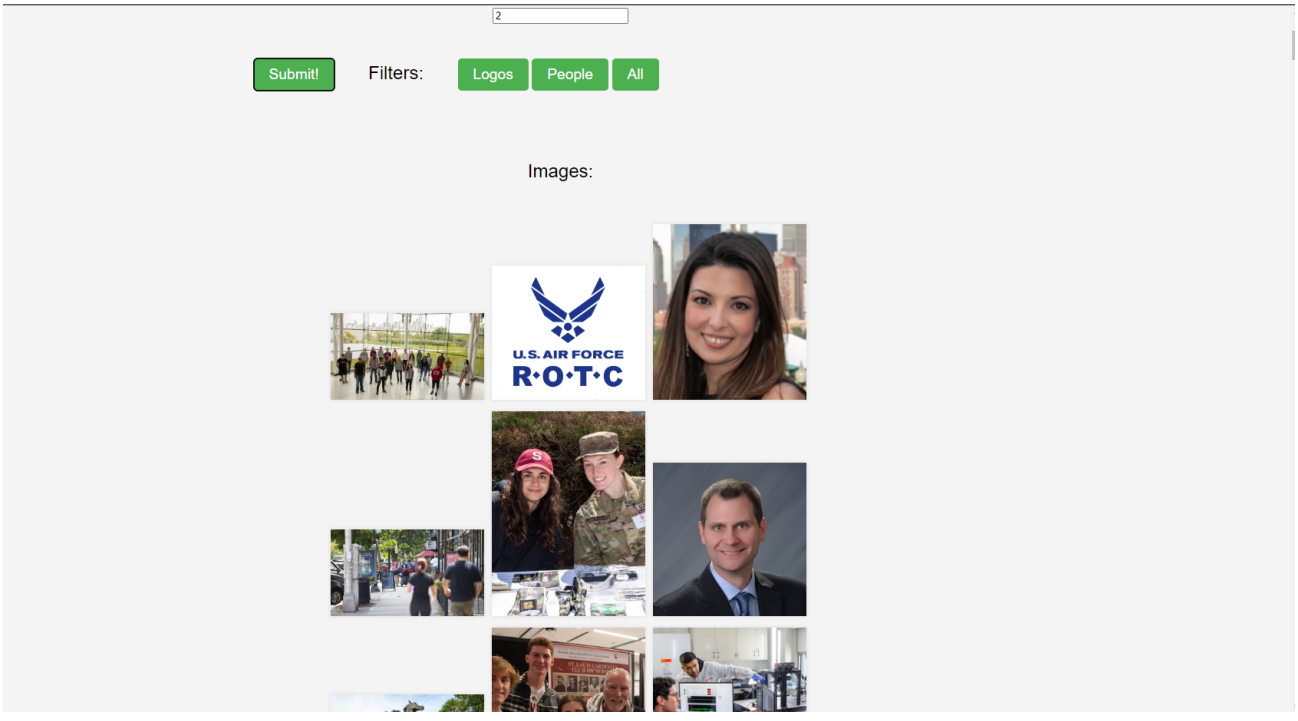
This instantly loads all the images extracted from the current page.

Now, we will try again with the same URL but with depth = 2

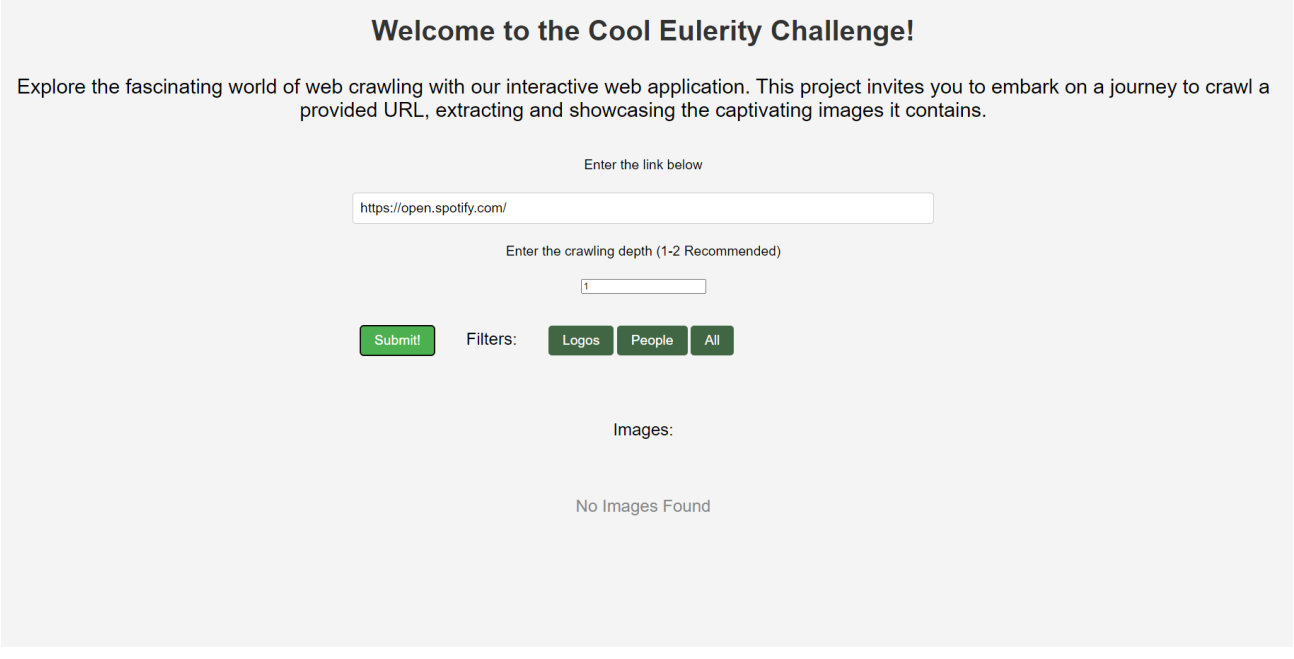


Till the time all images are extracted and displayed we see this page with no Images yet. After 4-5 seconds, the system displays all the images. This may vary every time depending upon the URLs, their server access time, the number of links available on this URL, and other factors.

Once the page loads we see all the images.



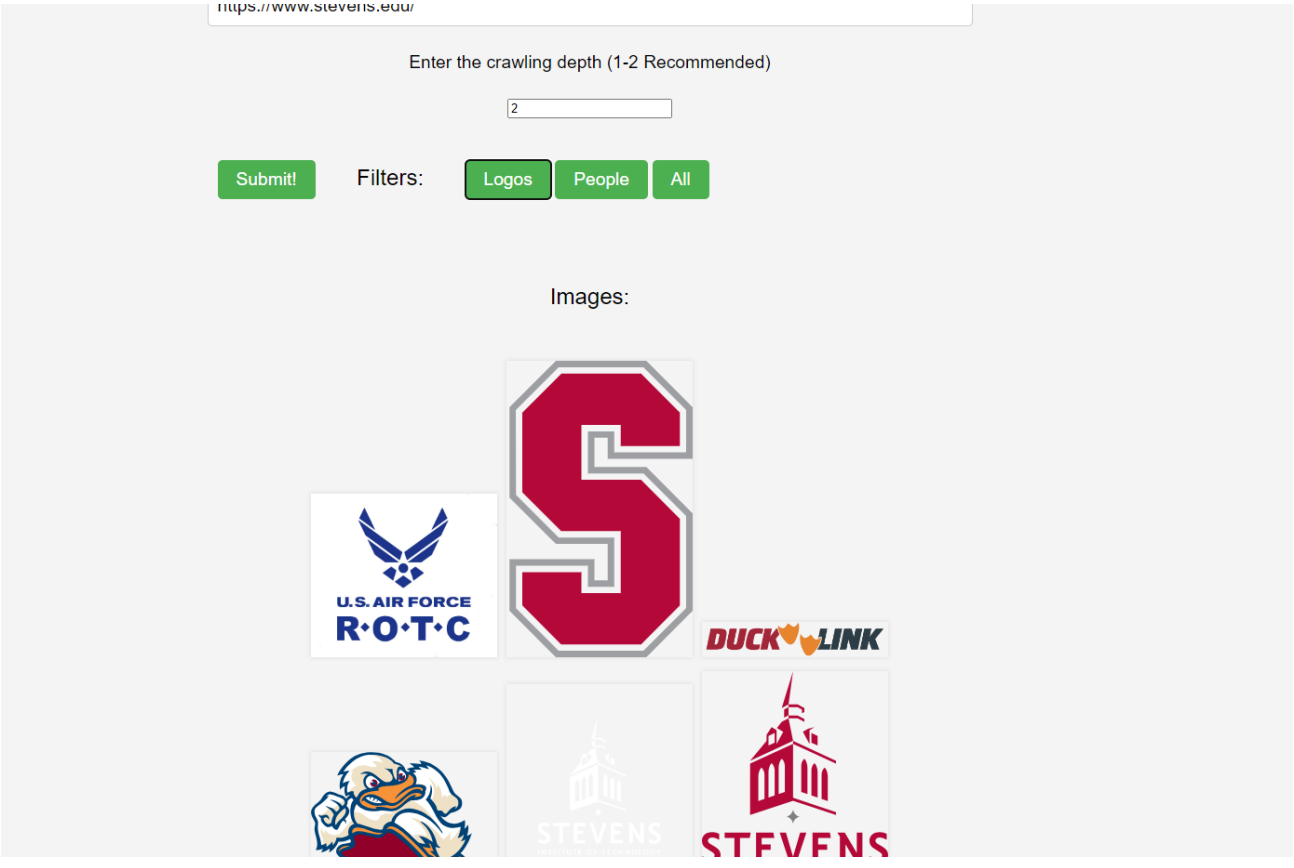
If there are no images found on any particular website due to restricted access or no images available then we get the following display, stating “No Images Found”.



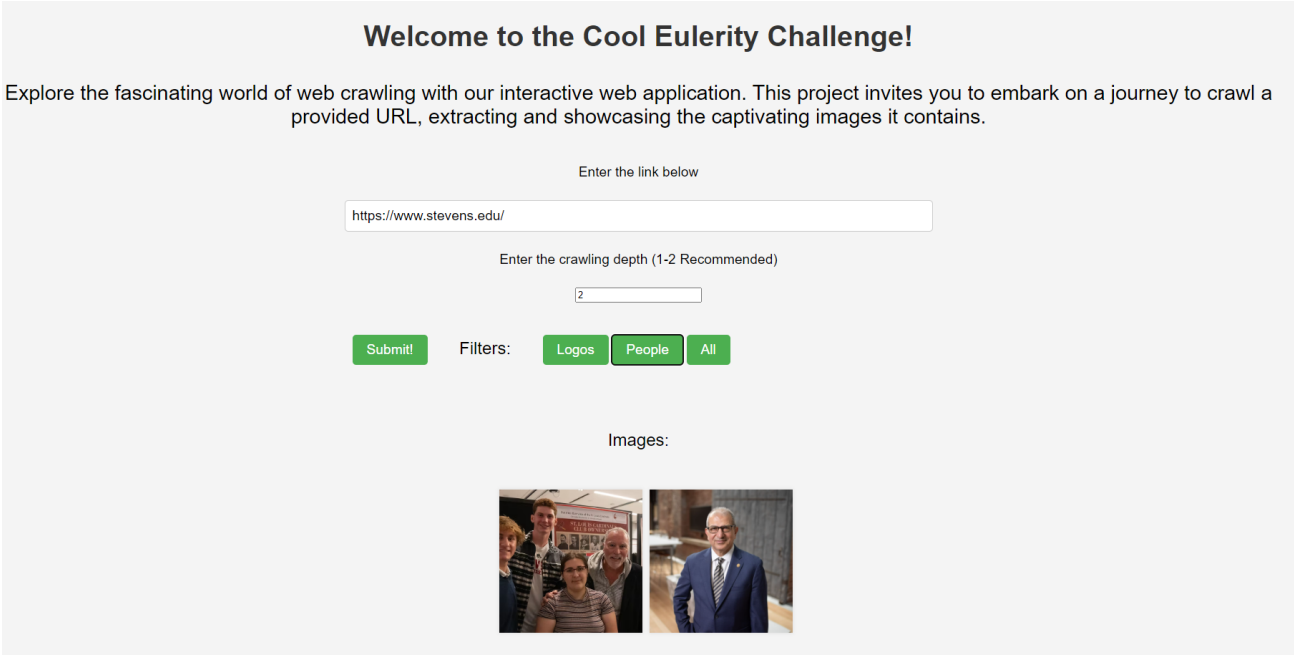
Filtering: We can filter all the images to check if they are logos or if they have people in them by clicking on the Logos or People buttons.

(As this filtering is based only on the keywords present in the image URL string, associated with logos or people respectively, it may not give the best and 100% accurate results)

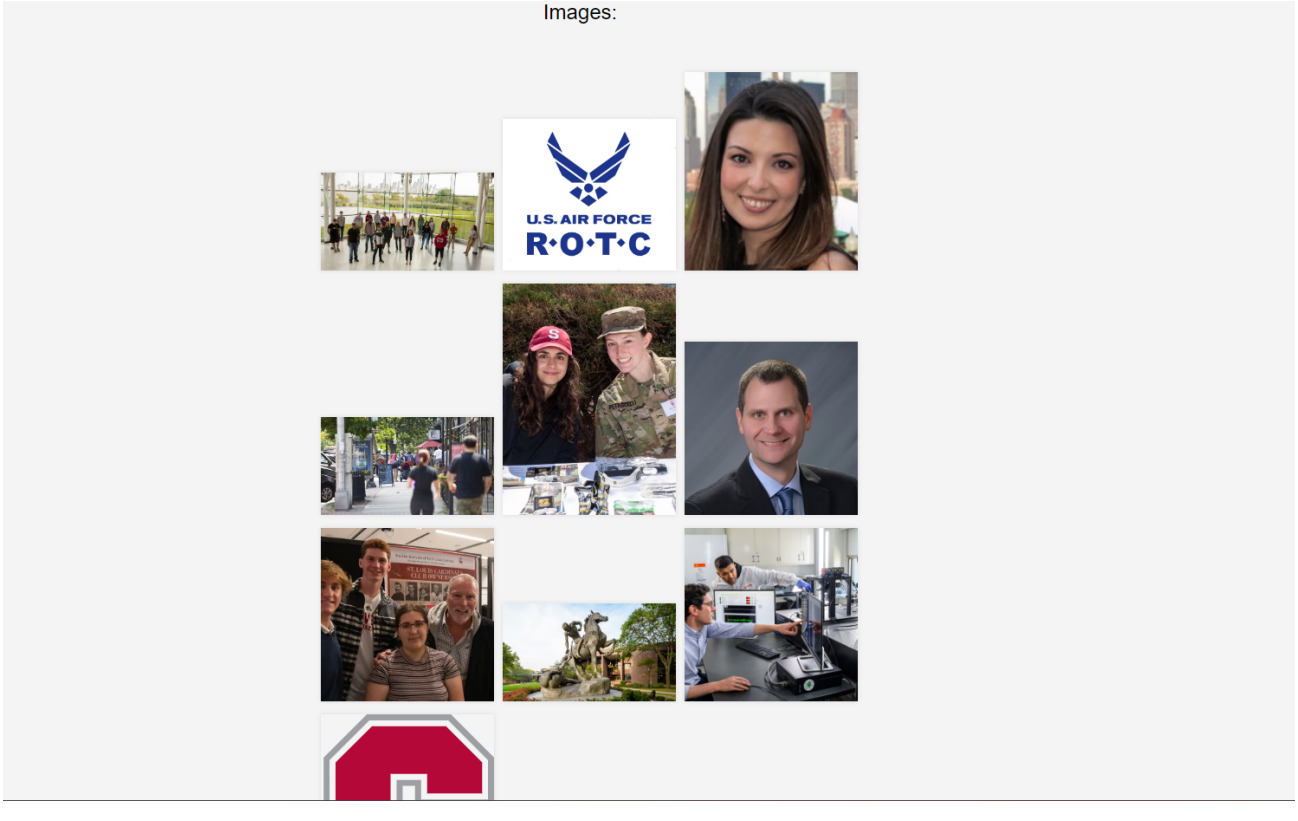
We will try filtering for Logos for the URL: https://www.stevens.edu with depth = 2



We will now try filtering images having People for the same URL

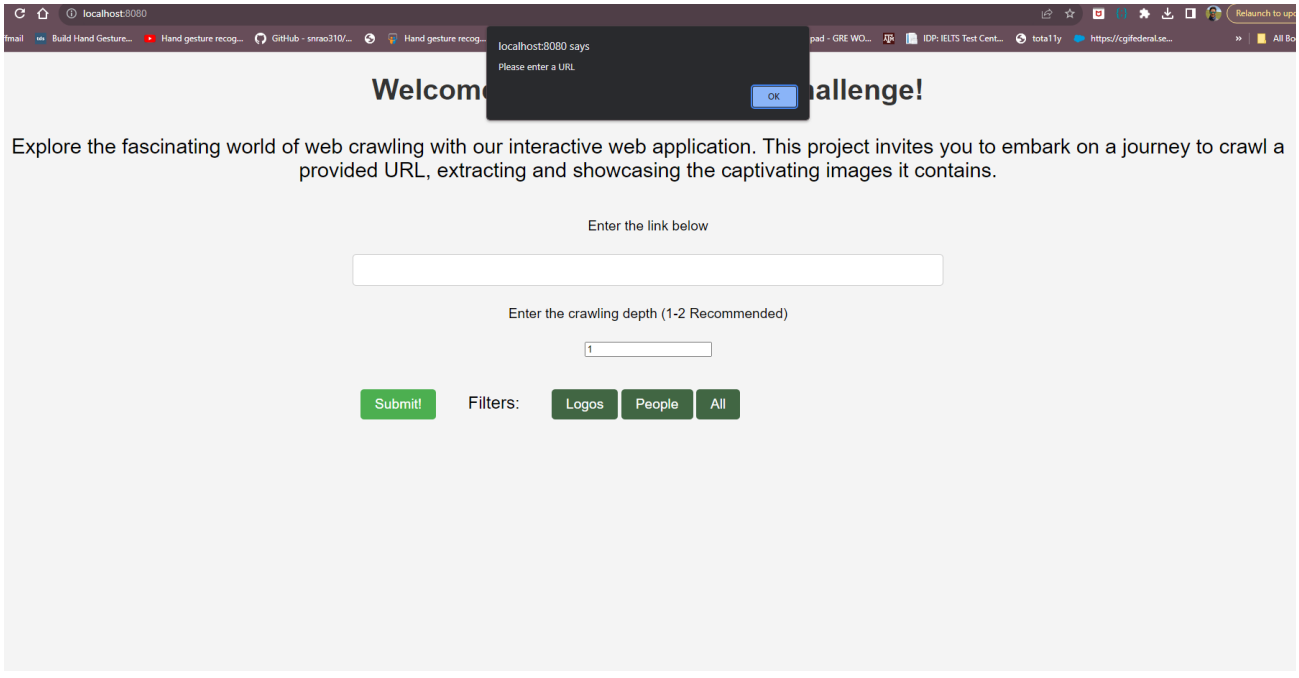


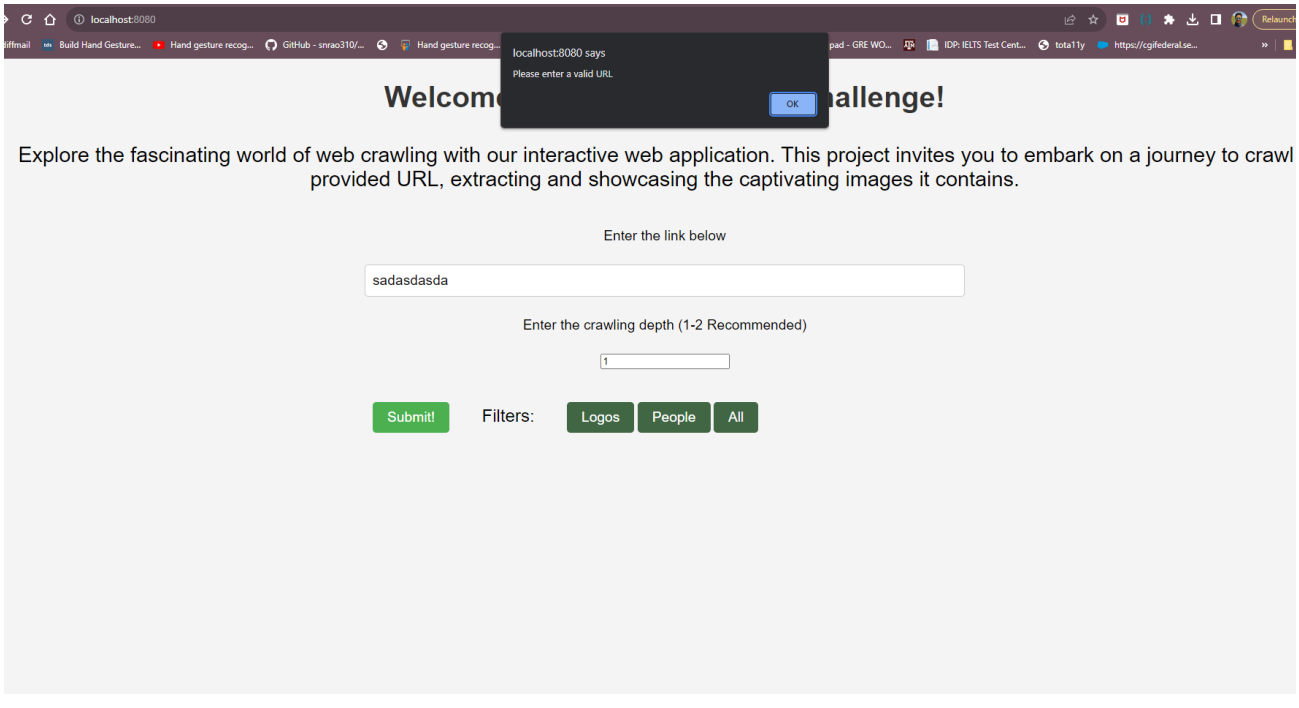
If we click on the All filter or again click the Submit button having the same URL and depth as the currently fetched data, then we will display all the images from the cached memory avoiding re-crawling and fetching data again.



Client Side Validations:

Attempting to submit an empty or invalid URL triggers an alert, reminding the user to input a valid URL before proceeding.





Notes:

- **Please keep the depth till 2 to get the best results. If the depth = 3 or more and if the URL is heavy and with a lot of sub URLs and images then it can take a very long time to process. Although I have used multithreading, it still may take time.**

- **For some URLs which are secure enough, This app is unable to fetch images eg. https://open.spotify.com**