Final Project Report

Gregory Petri

CSCE 489
5/2/2022

Summary

The project consists of a frontend web server that uses a device's camera to take photos of a user's eye. The user then has the option to "signup" or "login". The the user chooses to "signup", the backend performs feature detection with `python-opencv SIFT detection` on their eye image to attain metrics and store them in a database with the user's nickname as the key. SIFT detection intelligently and quickly discovers keypoints and calculates the descriptors around these keypoints. If the user chooses "login", the backend still performs the same feature detection process on their eye image, but does not save the data to the database. Instead, the backend fetches every user's data from the database finds all the matches between that data and the "logging-in" user's keypoints and descriptors. With `python-opencv FLANN matching`, SIFT descriptors can be easily compared and the matches plotted in a graphic. Once the matching is complete, the user is then identified with the nickname corresponding to the maximum matches.

Instructions

Open the web app

Locally

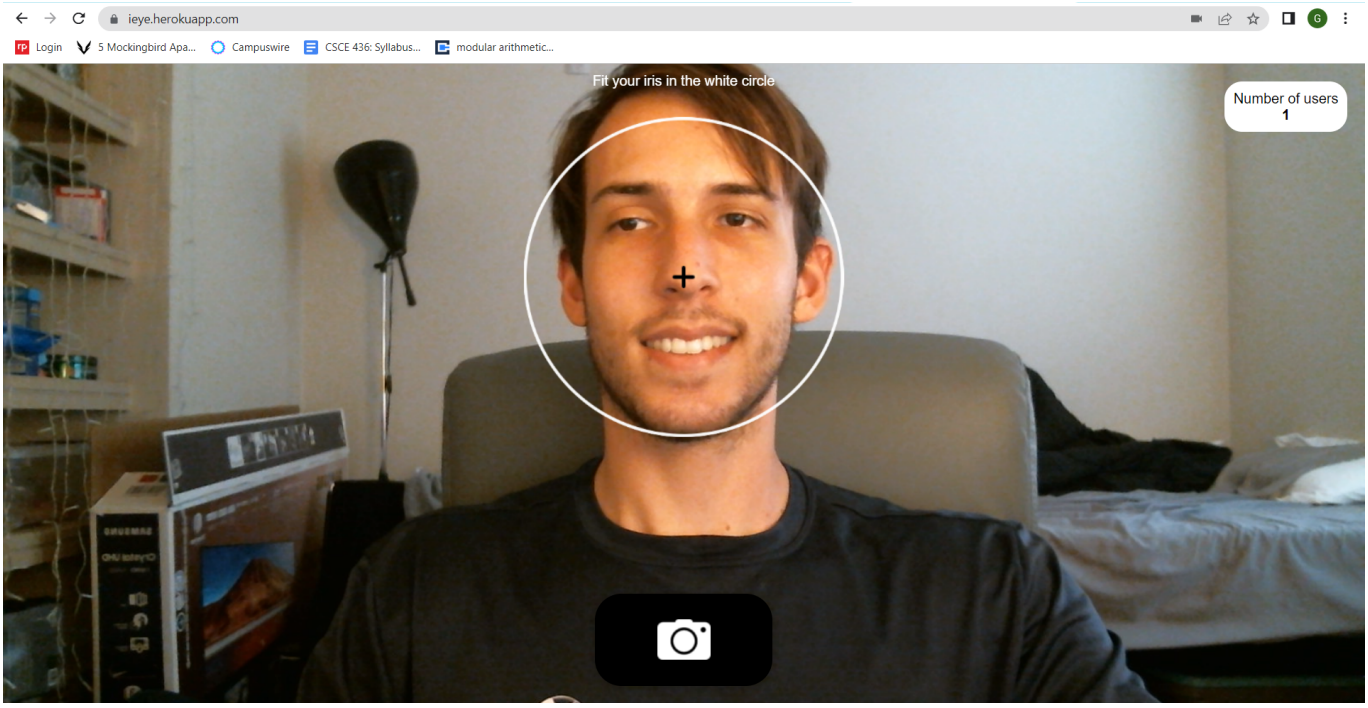
```
.\init-ps1 # PowerShell
# - or -
./init.sh # Bash
```

http://localhost:5000/

Online

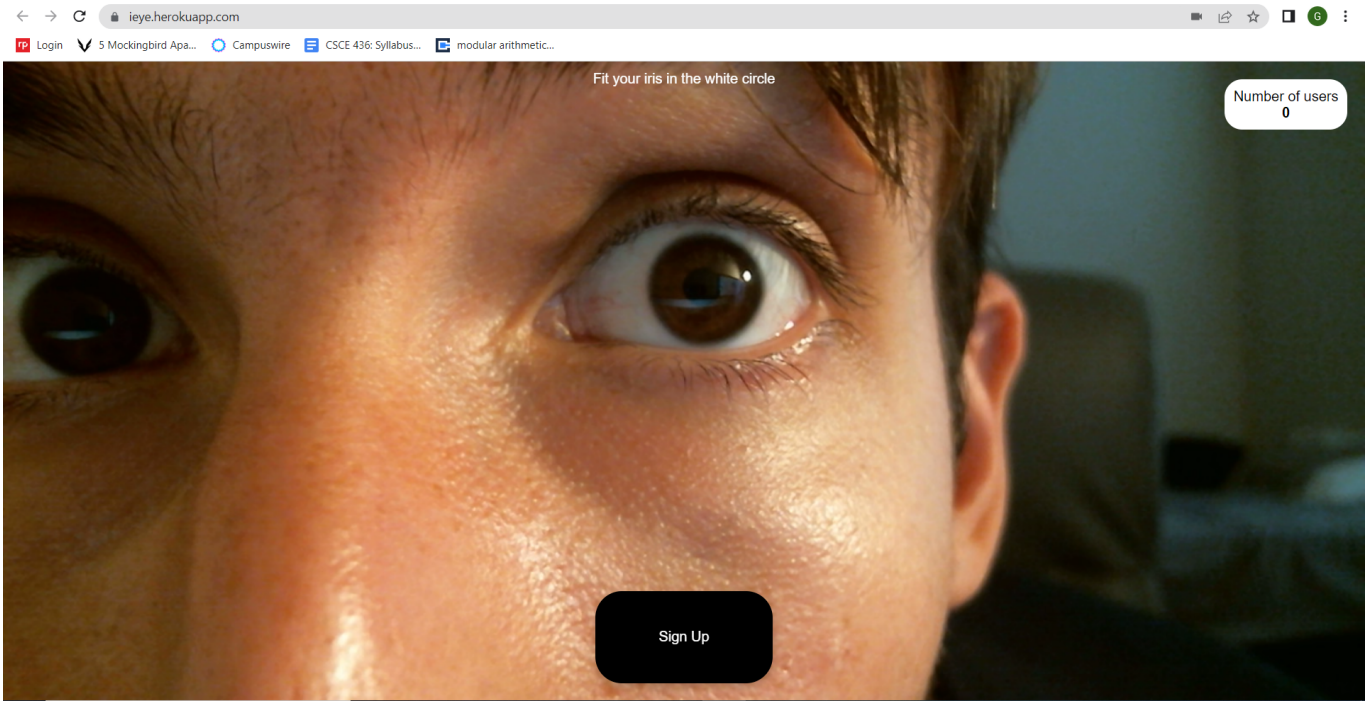
https://ieye.herokuapp.com/

View the Home Page



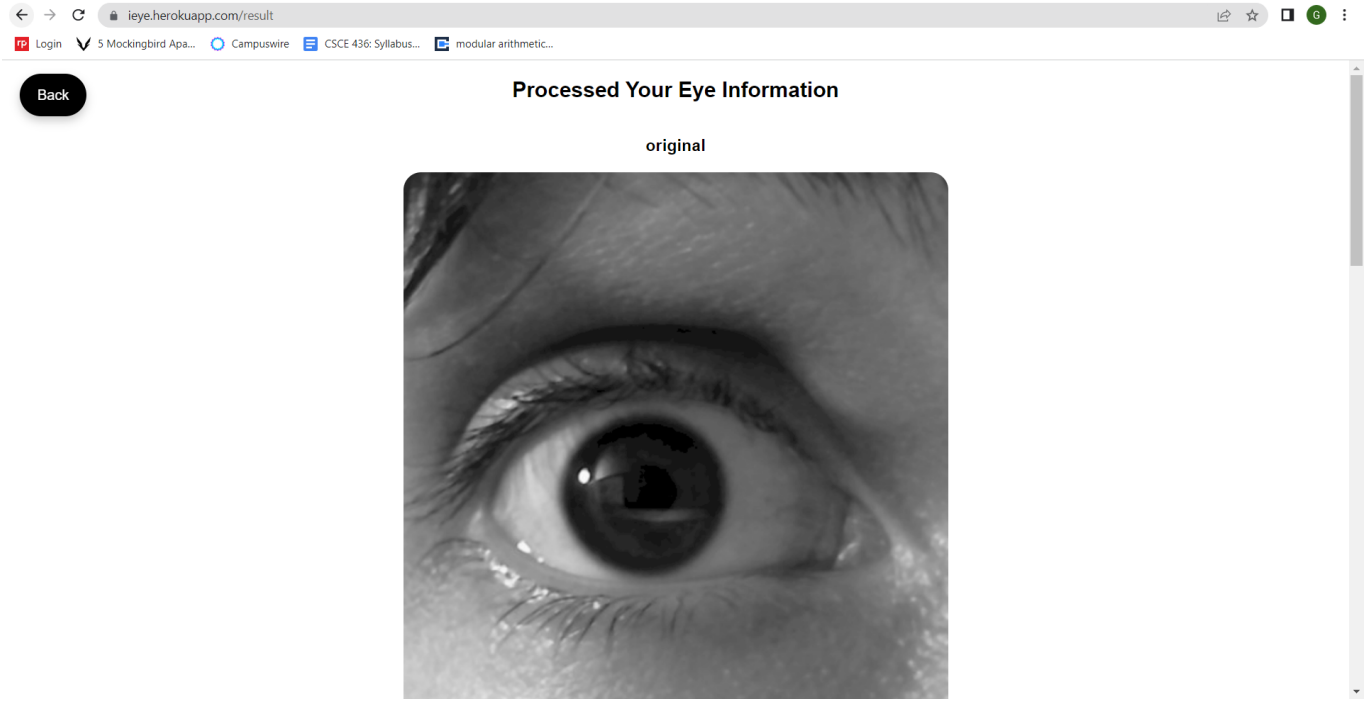
Take an Eye Picture

*And enter a unique nickname**

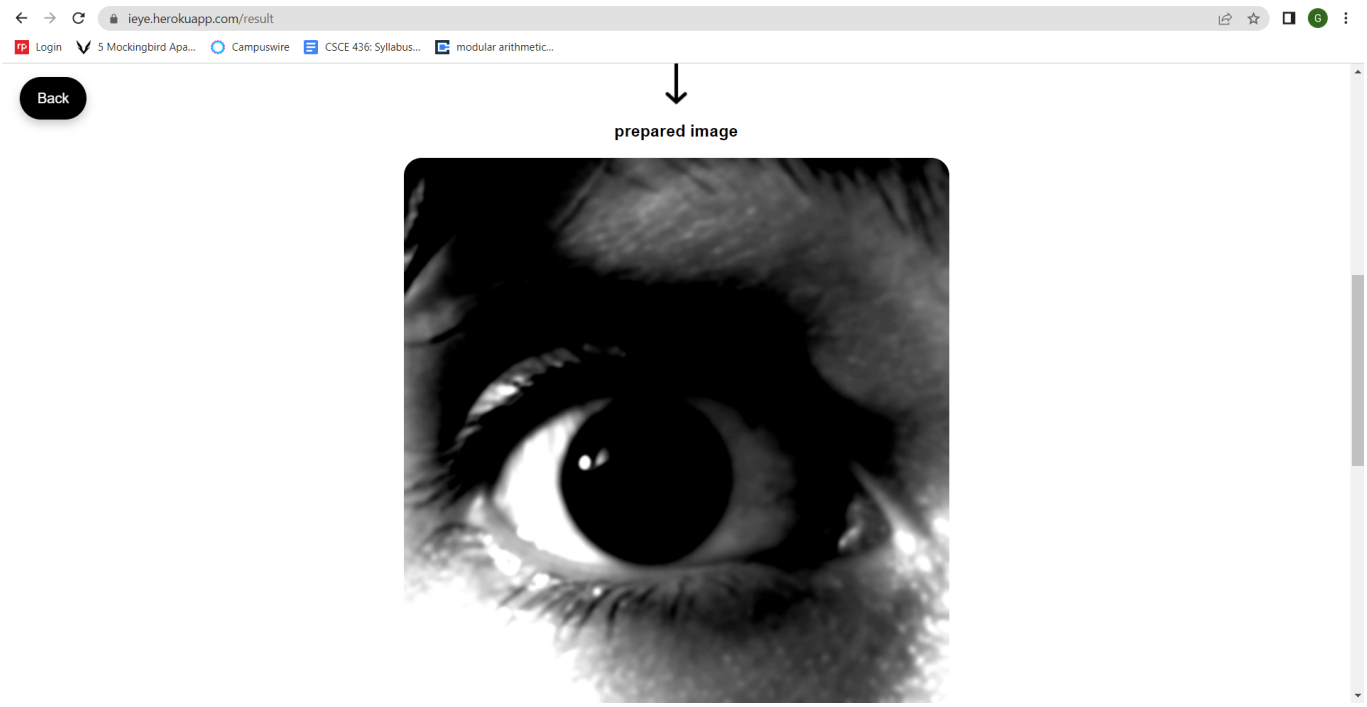


Sign Up

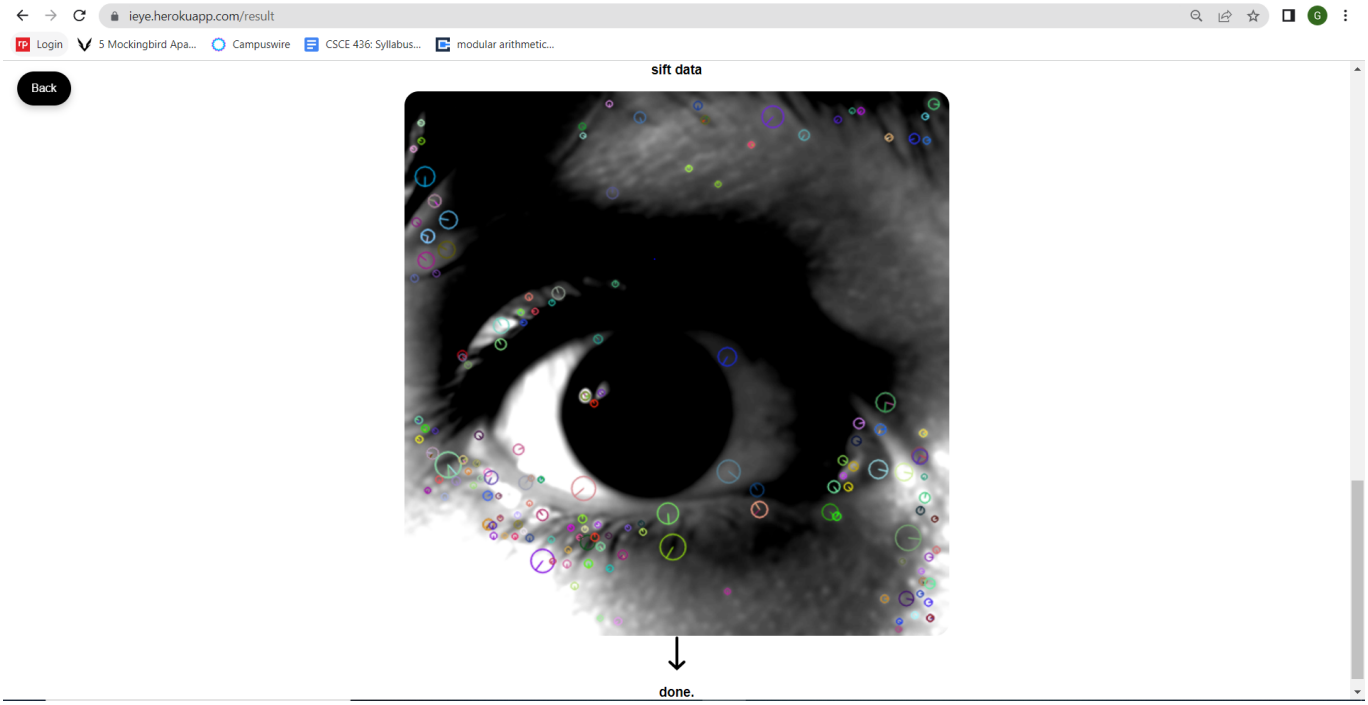
Original Image



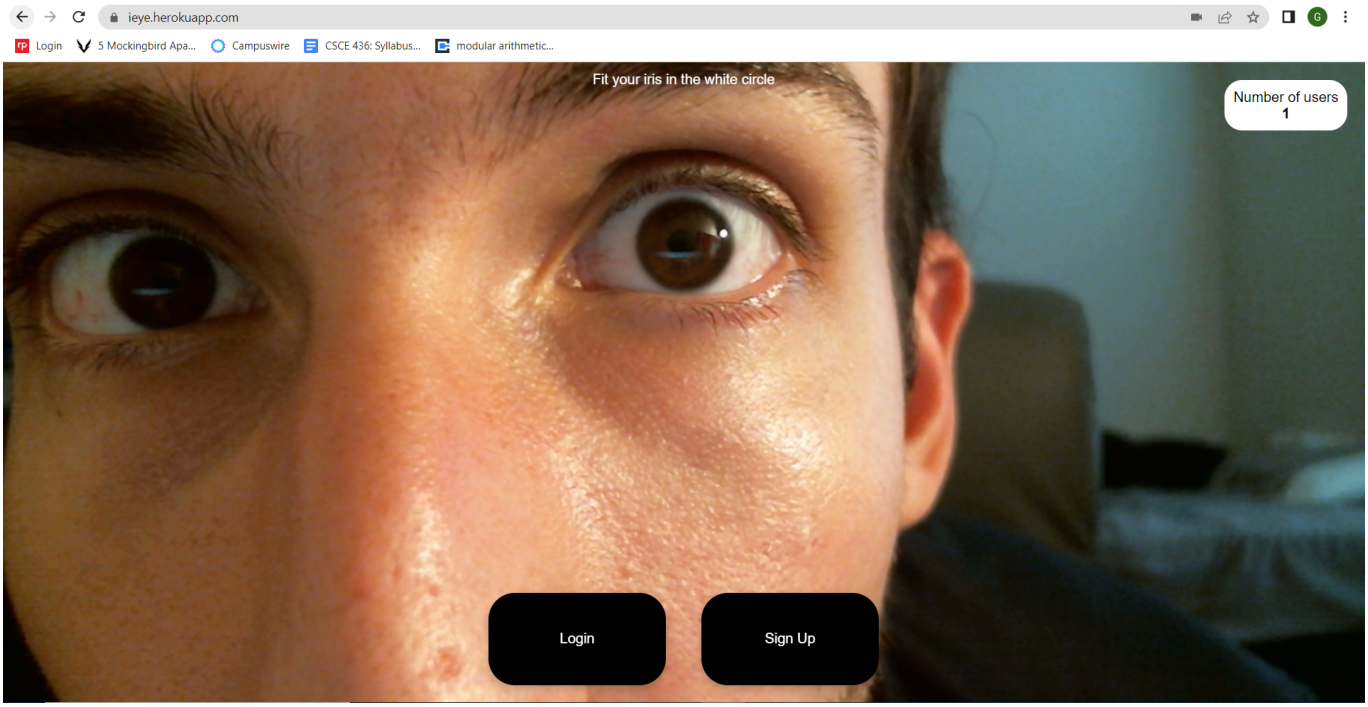
Resized, Smoothed, and Contrasted Image



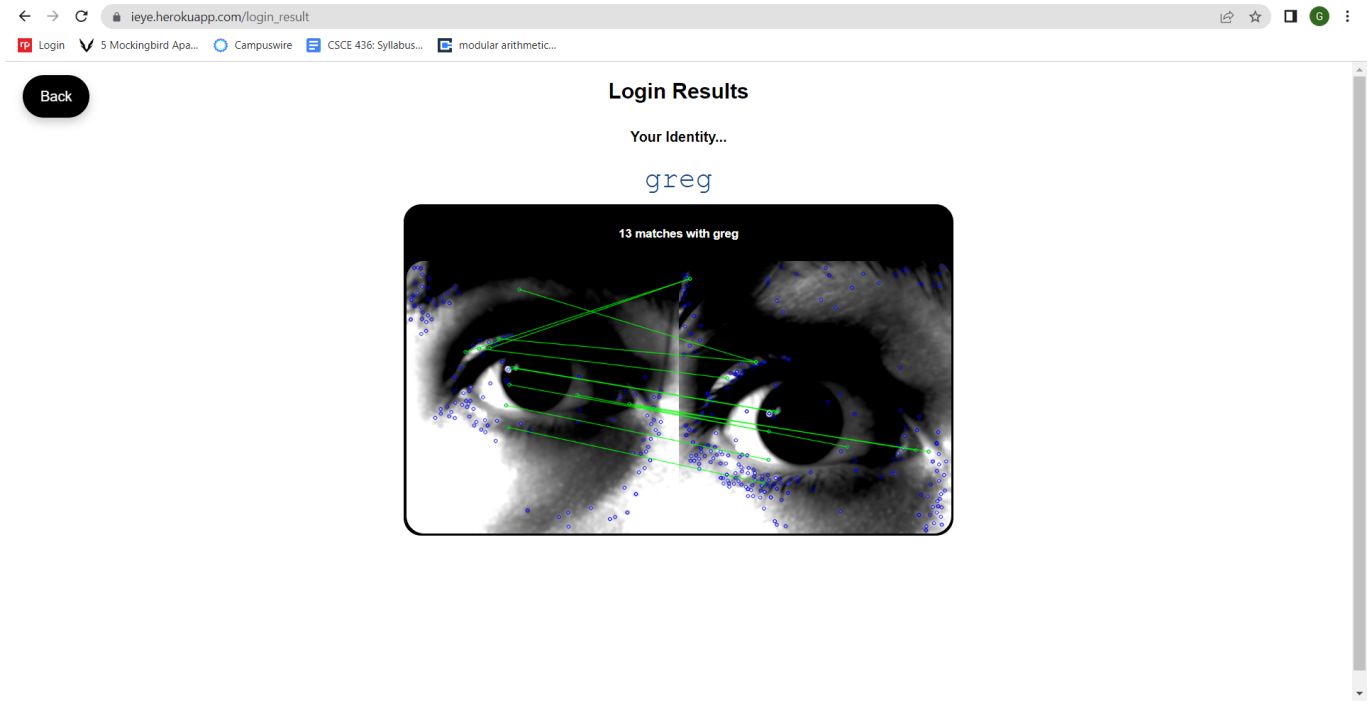
SIFT KeyPoints and Descriptors Display



Back to Home and Take Another Image



Login



Discussion

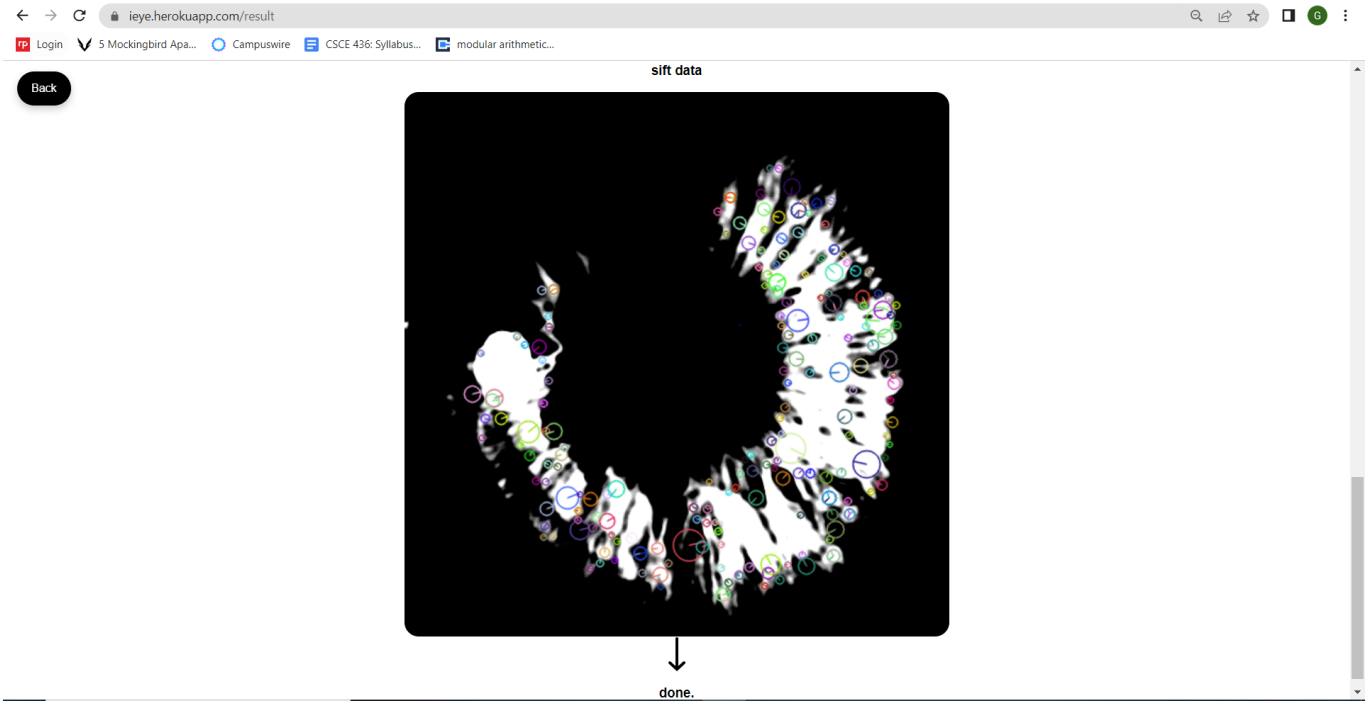
Implementation

I started the implementation with bare-bones code. I created a custom gaussian kernel, harris detector, and heuristic for the keypoints. Some of the code for these is still in `harris.py`. Unfortunately, not only did my implementations give questionably accurate and slow results, but also calculating keypoint descriptors proved too challenging. In the interest of run-time and my busy schedule, I turned to the miracle that is the `python-opencv` library. This allowed me to quickly compute descriptors and make them orientation invariant with `cv2.SIFT_create().detectAndCompute()`. This saved me a lot of time, especially since it comes with a compatable matching function `cv2.FlannBasedMatcher()`. The main challenge was storing the images, keypoints, and descriptors in a database. I decided to use MongoDB Atlas because it is free and JSON objects can be large. Once the I solved converting numpy arrays to binary and vice-versa, the remaining work was trivial.

Conclusion

I worked really hard on this so I hope it is enjoyed!

The closer the perimeter of the iris is to matching the circumference of the white circle, the better the results will be. To see the intended results, go to https://ieye.herokuapp.com/test


ieye.herokuapp.com/result
Login
5 Mockingbird Apa...
Campuswire
CSCE 436: Syllabus...
modular arithmetic...
Back
sift data
done.