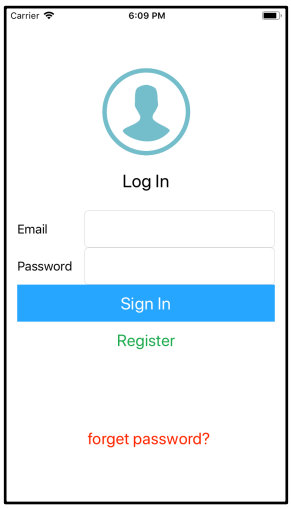
Project Description

This project enables users to upload images from mobile devices to a cloud database for the purpose of facial recognition. Separately, law enforcement will be able to add to the list of suspect faces that is used to find matches. As images are uploaded by users, the faces are extracted and checked for matches. Facial match percentages, locations, and timestamps are presented to law enforcement on the web application in hopes of aiding in finding suspects during an emergency.

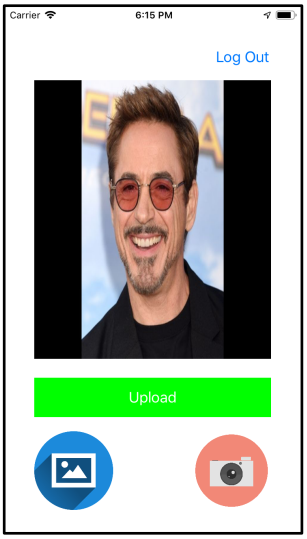
User Manual (iOS Application)

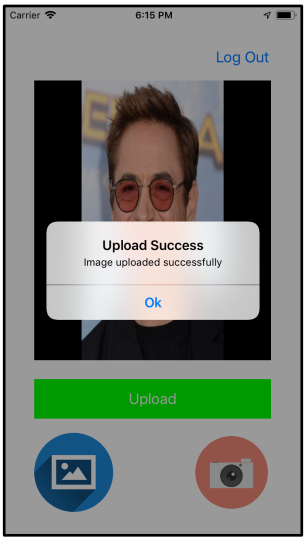
1. Users are first prompted to log in.



2. After logging in, the user is presented with the following screen. The buttons give the user the option of choosing an image from the device's camera roll or taking a picture inside the application.

3. After selecting or taking an image, the user is able to upload said image to the cloud database by hitting the now-illuminated “Upload” button.

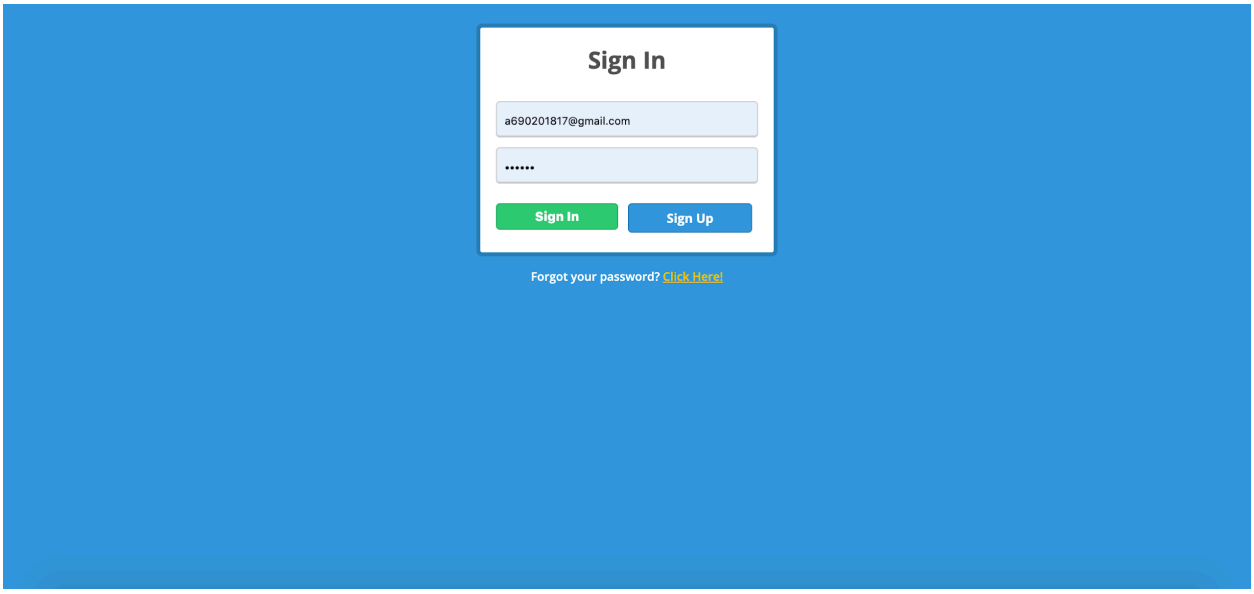




User Manual (Web Application)

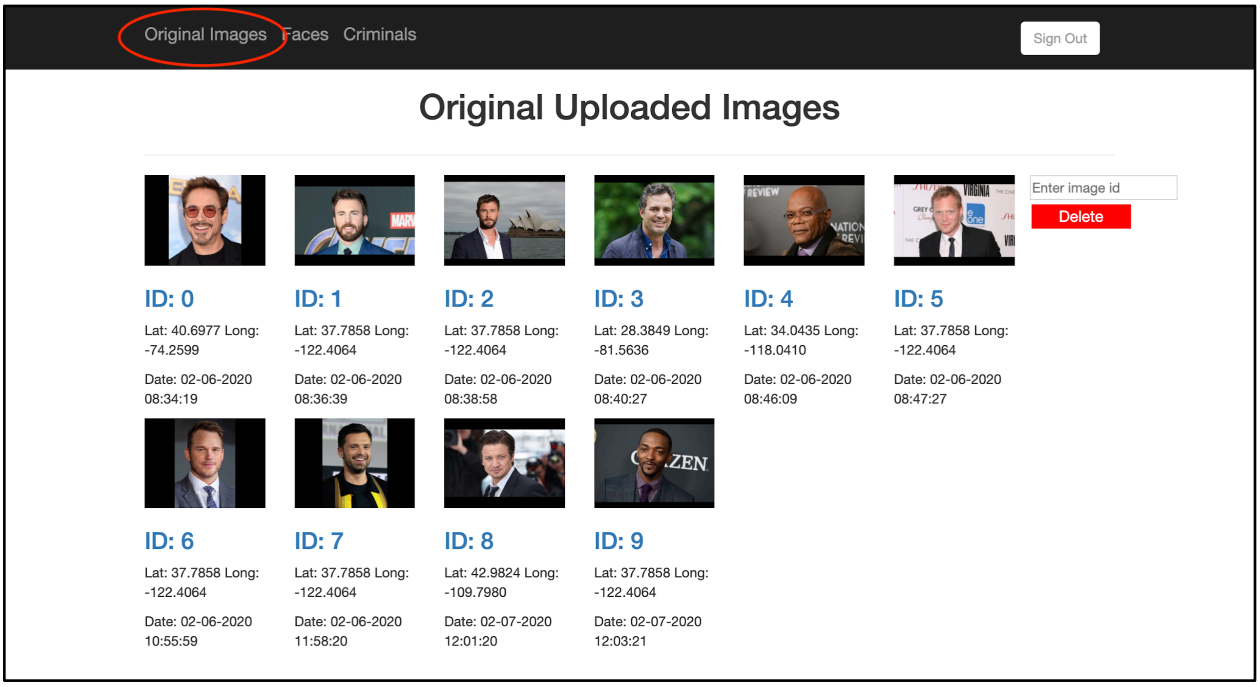
URL: https://iosreserach2017firebase.firebaseapp.com/

1. Users will first be met with a login/sign-up screen after visiting the above URL.

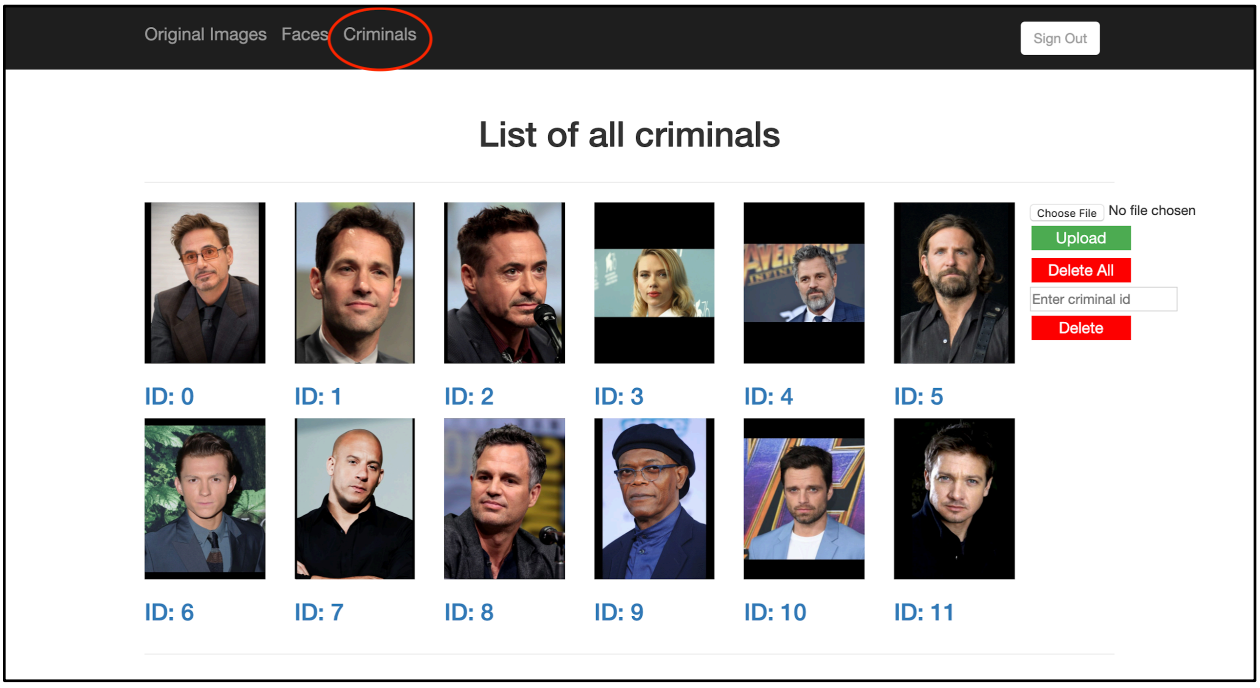


2. The home page is the “Original Uploaded Images” page, also accessible by clicking the “Original Images” tab on the navigation bar. This displays all of the images uploaded by users of the mobile app, along with location and timestamp. Users of the web application are able to manage the images by deleting individual ones based on ID using the “Delete”

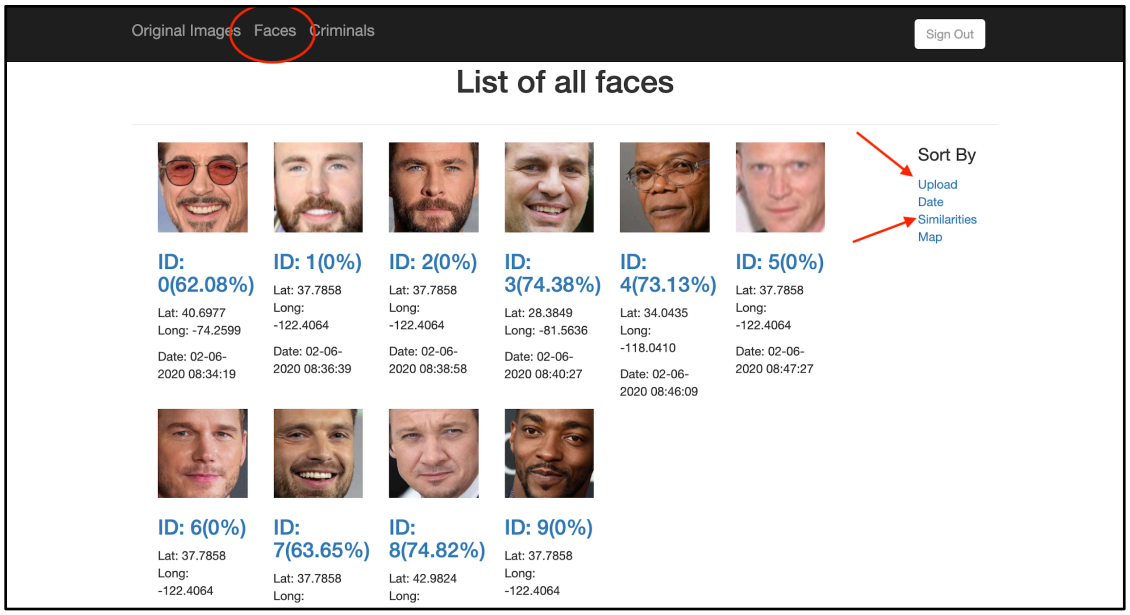
button towards the right of the page. Users can sign out at any time by clicking the “Sign Out” button at the top right.



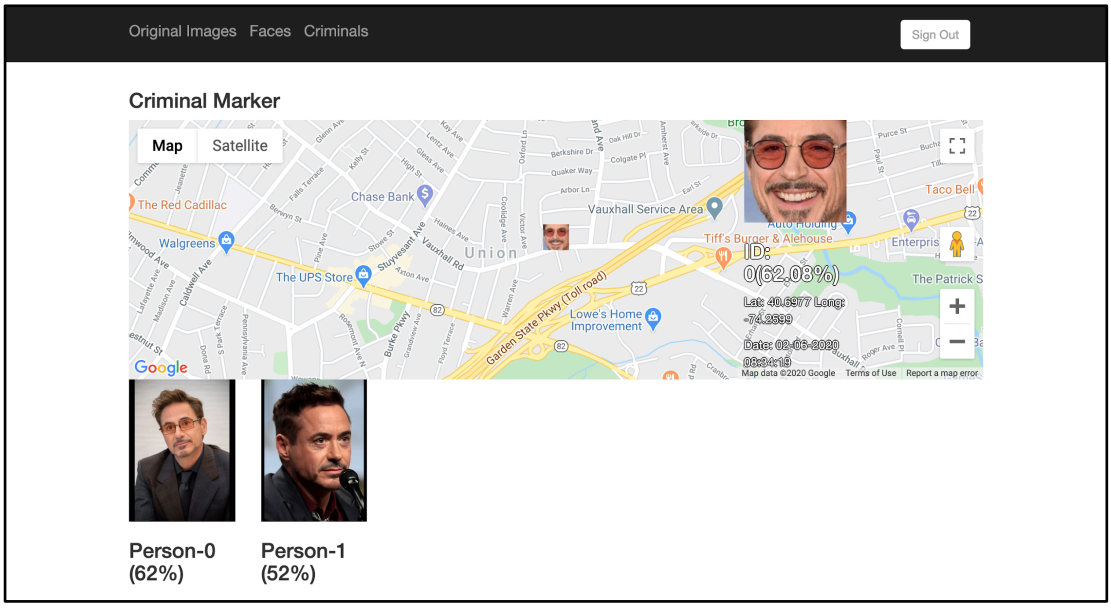
3. The “Criminals” tab takes the user to the following page, where the collection of criminal images used to calculate matches is managed. Here, a user can upload new images from their device as well as make deletions.



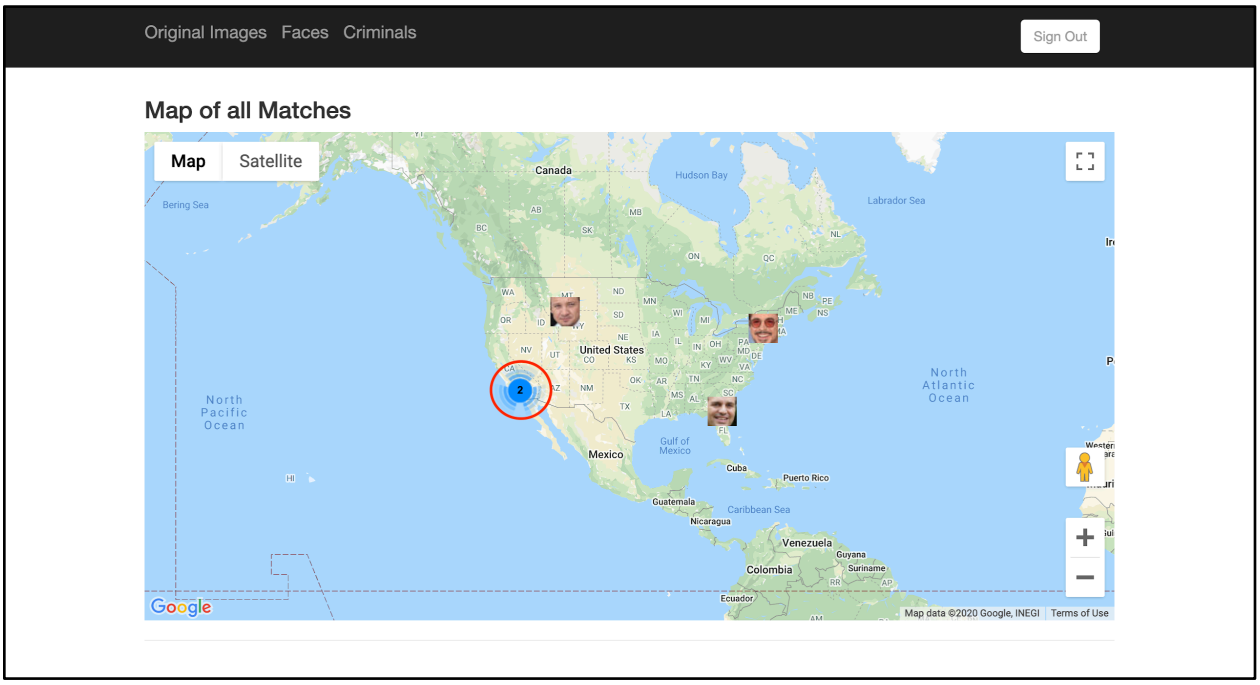
4. The “Faces” tab takes users to the following page which displays the face thumbnails extracted from the images uploaded through the mobile application. In addition to location and timestamp, each thumbnail is accompanied by the highest match percentage it has with a face from the criminal list. These thumbnails can be sorted by upload date or match percentage by clicking the links on the right hand side. The “Map” link takes the user to the match page explained in (6).



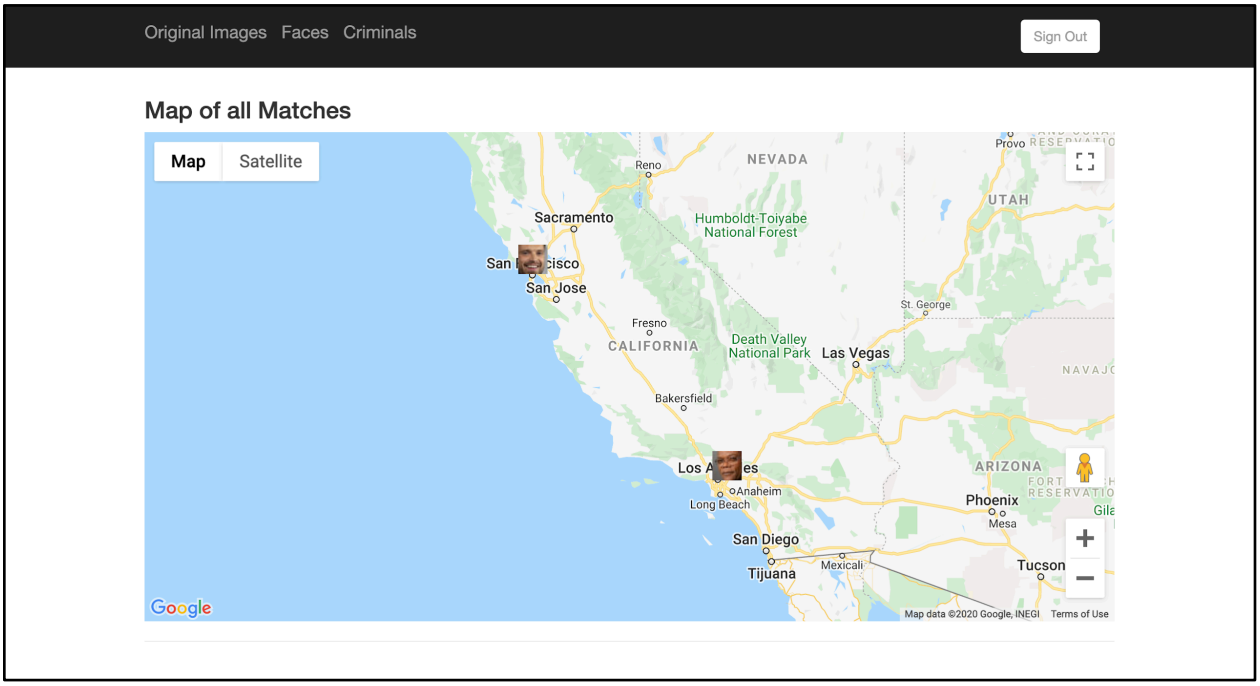
5. Clicking an ID of any thumbnail with a match higher than 0% will take the user to a page like the following one. This page places a marker of the suspect on a map where the image of the suspect was taken. Under the map are the images of the criminals that the suspect matched with along with the match percentages.



6. Returning to the “Faces” tab, users can get to the following page by clicking the “Map” link on the right hand side. The page displays a master map containing markers for each of the suspects with criminal matches. Markers grouped too closely to distinguish at the original zoom level are clustered together, as shown by the red circle.



7. Grouped clusters can be clicked to zoom in and reveal the actual suspect markers.



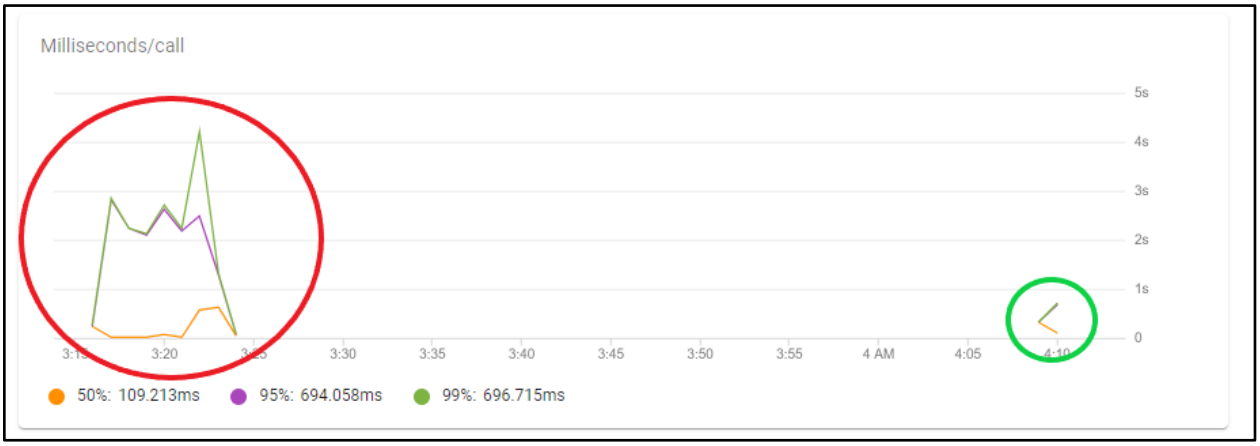
Testing

Testing of the iOS application and cloud trigger function was done by utilizing Amazon Web Service's (AWS) Device Farm through automated tests - written using the Appium Python framework. Through AWS Device Farm and automated testing, we are able to observe the effect of concurrent and rapid-fire image upload from multiple "real" devices in an assorted pool:

✓	Apple iPhone 7	12.0	✓ 3
✓	Apple iPhone 7 Plus	12.0	✓ 3
✓	Apple iPhone 8	12.0	✓ 3
✓	Apple iPhone 8	11.0.3	✓ 3
✓	Apple iPhone 8 Plus	12.1	✓ 3

Concurrent Upload

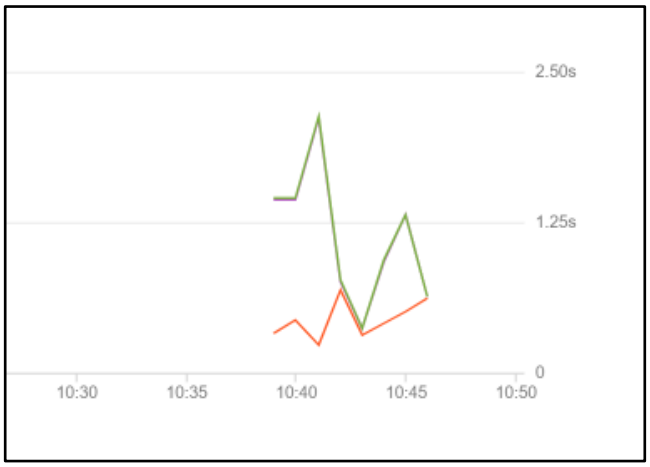
The following graph shows the effect that concurrently uploading five images on five different devices has on the execution time of the cloud function. This is shown within the red circle, where the time needed to complete the function call dramatically spikes as the number of concurrent calls increases. Ideal execution circumstances are shown in the green circle, with only one call being executed from a single device.



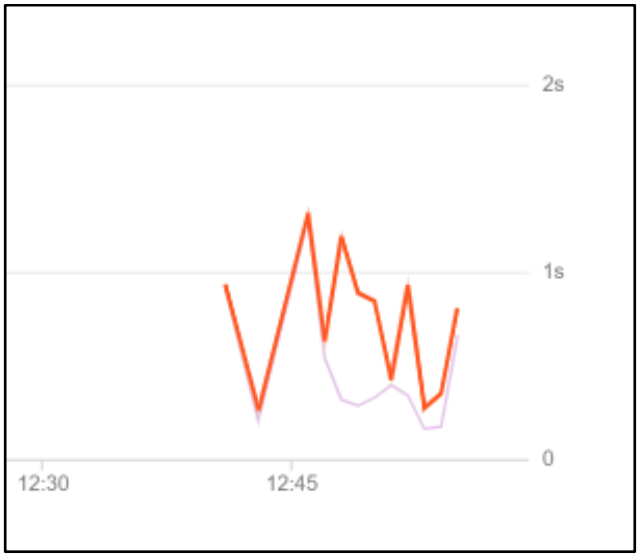
This issue of increasing execution times could possibly be addressed by allocating more cloud resources for such a function or by separating upload functionality from thumbnail generation and similarity calculation (implementation without the use of a trigger).

Multiple Uploads from Single Device

The following graph shows the effect that uploading 25 images sequentially on one device has on execution times. This is significantly less than the example shown above, with the same number of images but multiple devices uploading concurrently. The use of one device may weigh less on the resources used by the cloud trigger function than when calls are made concurrently.



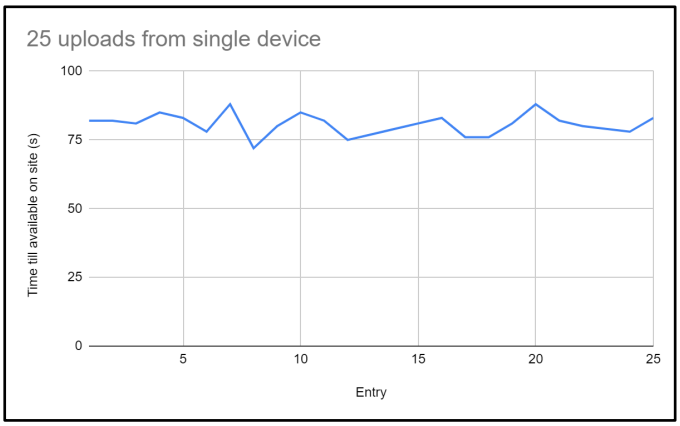
The effect of uploading consecutive images from one device can vary, as shown in the graph below. This shows the execution time of the cloud trigger function in a separate run with the same conditions as above, illustrating that there can be variance in the performance even with all things held constant.



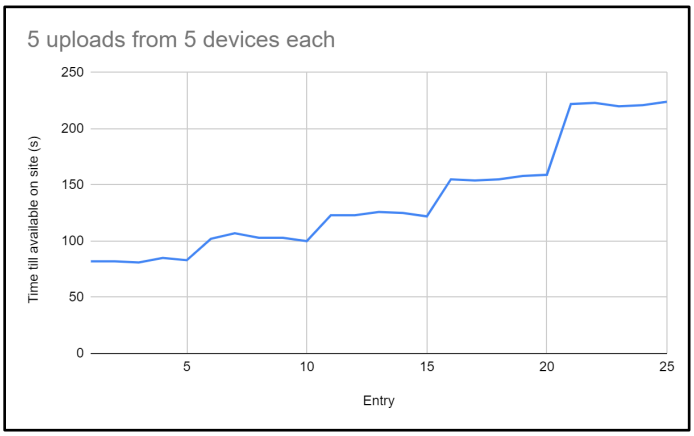
Display Time on Web Application

Though image upload occurs quickly and is reflected in execution times, thumbnail generation and similarity calculation is done asynchronously. How long these processes take is equivalent to how long it takes for thumbnails and similarity percentages to be displayed on the site.

Time till available on site refers to the time between when an image is uploaded from the iOS application and when that image's thumbnail and similarity measures are displayed on the web application. With one device uploading 25 images consecutively, time till displayed remains fairly constant as follows:

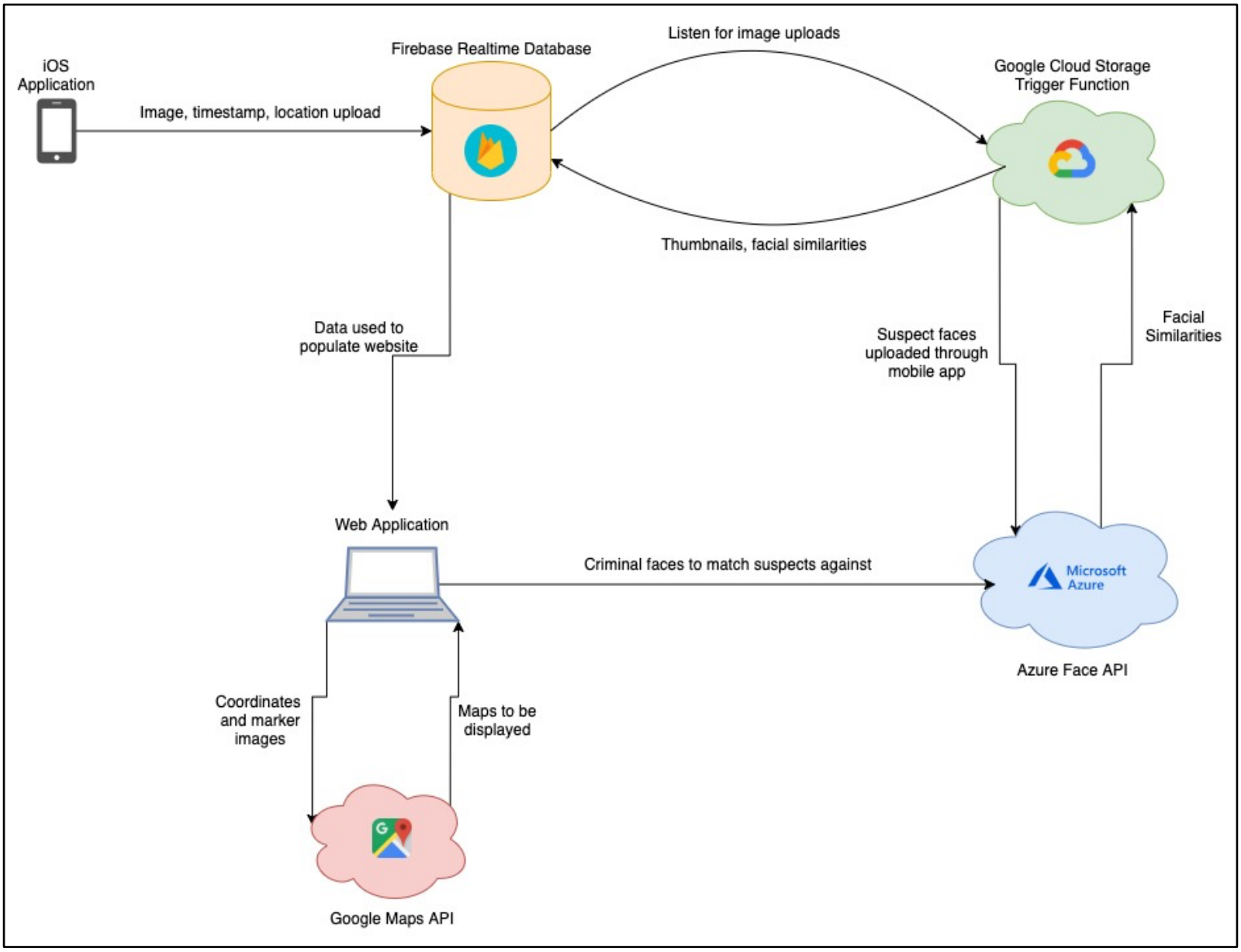


With 5 devices uploading 5 images each concurrently, time till available on site changes as follows:



It can be observed from these results that concurrent calls to the cloud trigger function from multiple devices is less efficient than the same number of calls made one after another from a single device.

Architecture



List of Functionality

Mobile Application:

- User login with email and password
- Logout
- Choosing images from camera roll
- Taking new pictures from within the application
- Image upload

Web Application:

- User login with email and password
- Logout
- Display uploaded images, locations, and timestamps from mobile app
- Display images uploaded to criminal list
- Display thumbnails extracted from mobile app images
- Display match percentages
- Sort by match percentages
- Individual match page with map and criminal matches shown
- Master map page with suspect thumbnail markers
- Marker clustering for markers too close together

Firebase Cloud Trigger Function:

- Crop thumbnails of faces from uploaded images
- Resize thumbnails for use as map markers
- Detect faces from uploaded images
- Call Azure API to determine facial similarities
- Store similarities and other associated data in Firebase Realtime Database