

HiQ – Photo Uploading System with Face Detection and Recognition

User and Development Guide

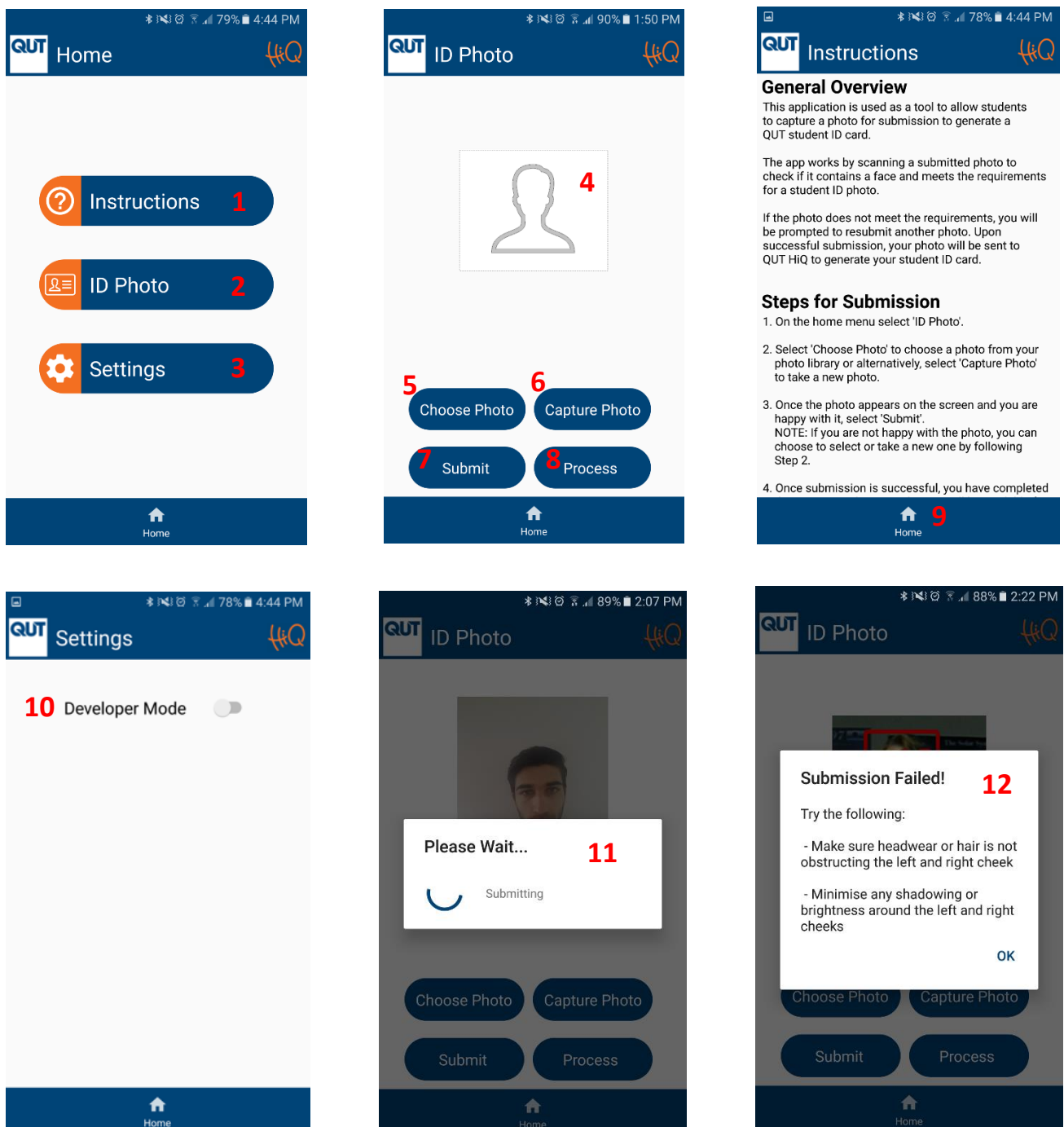
Last Updated: 1st November 2017

1.0 User Guide

This is a User Guide which outlines the features which make up the Face Detection Application from a user's perspective.

1.1 Android Application User Interface

The following diagrams depicts each of the screens within the Face Detection Application with the associated buttons and features labelled. An explanation of each of these labels can be seen in the following table.



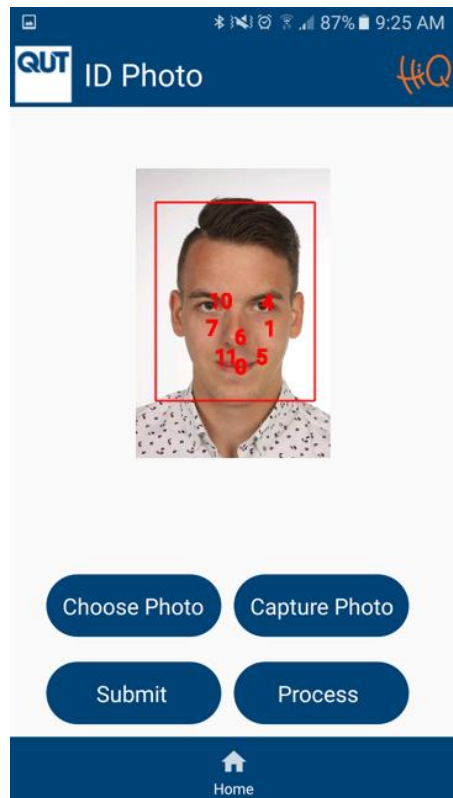
Label	Description
1	Instructions button which directs to the Instructions screen when tapped. This screen contains information regarding the application, how to use it and tips to take a good ID photo.
2	ID Photo button which directs to the ID Photo screen when tapped. This screen contains functionality to capture or upload a photo and perform face detection on a photo.
3	Settings button which directs to the Settings screen when tapped. This screen contains a switch to swap between Developer Mode or User Mode.
4	A placeholder image view which will be replaced by an actual photo when a photo is captured or uploaded from the device image library.
5	Choose Photo button which allows a user to select a photo from their image library currently stored on their device to use as their student ID photo.
6	Capture Photo button which brings up the device camera so that a user can take a photo of themselves to use for their student ID photo.
7	Submit button which uploads the photo currently sitting on the image view to a remote webserver so that it can be retrieved later on.
8	Process button which scans the photo sitting on the image view with face detection algorithms in order to detect a face and each facial landmark. The scanning also compares the photo with the industry standards for ID photos.
9	Home button which appears on every screen and directs the user back to the Home screen, containing the list of menu items, when tapped.
10	Developer Mode switch which when toggled to true will switch the app to Developer Mode and when toggled to false will switch the app to User Mode. NOTE: The Functionality of this switch has not been implemented and is merely there as a proof of concept.
11	Overlay prompt indicating that an image is currently being submitted to the webserver.
12	Error message which has appeared due to a face not being detected or the photo not meeting the industry standards set for ID photos.

Steps for Photo Submission and Face Detection:

1. Starting from the Home screen, tap the 'ID Photo' button from the menu.
2. Once on the ID Photo screen, tap the 'Capture Photo' button to take a photo of yourself to use as a student ID photo. Alternatively, tap the 'Choose Photo' button to select a photo which already exists within the image library of the mobile device.
3. Once a photo appears within the image view, tap the 'Process' button to run the face detection scan over the photo. If the photo passes the scan, a red box and labels corresponding to each facial landmark will appear on the photo. If the photo fails detection, an error message informing the user that the submission has failed and outlining the reasoning as to why it has failed, will be displayed.
4. If the photo has passed detection, tap the 'Submit' button to upload the photo to a web server for later retrieval. At this point, the photo submission is complete.

1.1 Making Sense of the Output

When a face is detected, a red box is drawn around the face to indicate the detected area. The numbers which appear on the face correspond to the detection of facial landmarks. Each facial landmark is given an ID in the form of a number which is saved into an array. The array is searched to see if all of the ID's are present. If an ID is not present then the overall confidence level of the detection is lowered and an error message will appear based on the missing landmark.



Facial Landmark	Unique ID
Left Eye	4
Right Eye	10
Left Cheek	1
Right Cheek	7
Nose Tip	6
Left Mouth Corner	5
Right Mouth Corner	11
Bottom of Mouth	0
Left Ear	3
Left Ear Tip	2
Right Ear	9
Right Ear Tip	8

NOTE: The application is designed to reject images that are not between -9 and 9 degrees as this means that the face is not completely frontal. The following table shows what landmarks will be detected based on the angle that the face is orientated. If the left ear (ID: 3) or left ear tip (ID: 2) and right ear (ID: 9) or right ear tip (ID: 8) are detected then the application will reject the image as this is not a frontal face.

Euler Y Angle	Detectable Landmarks
Less than -36 degrees	left eye, left mouth, left ear, nose base, left cheek
-36 degrees to -9 degrees	left mouth, nose base, bottom mouth, right eye, left eye, left cheek, left ear tip
-9 degrees to 9 degrees	right eye, left eye, nose base, left cheek, right cheek, left mouth, right mouth, bottom mouth
9 degrees to 36 degrees	right mouth, nose base, bottom mouth, left eye, right eye, right cheek, right ear
Greater than 36 degrees	right eye, right mouth, right ear, nose base, right cheek

2.0 Developer Guide

This is a Developer Guide which entails further detail into the development tools for the project and how to run all of the code for both the initial OpenCV prototype algorithms and Android application. There will also be some explanation as to how some of the code works for the Android application. Please note that after progressing through the project it was found that the initial OpenCV based face detection prototypes were not good enough to use based on the use case of the project. However, if a situation arises meaning that these algorithms need to be revisited, then an explanation into the tools used to develop and test them are outlined.

2.1 Development Tools and Links

The development tools used throughout the project are listed here and are broken up between the tools used for designing the initial OpenCV face detection prototypes and developing the Android application.

2.1.1 Project Repository and Tools

GitHub repository contains all of the code used for the project. To access any software requiring a username and password, contact the project supervisor to grant you access. The face database contains folders full of faces which were used for testing prototype algorithms.

GitHub Repository: <https://github.com/higproject/HiQ-Photo-Uploading-and-Face-Detection>

Project Management Software: Atlassian's Jira - <https://higproject.atlassian.net/wiki/discover/all-updates>

Project Collaboration Software: Atlassian's Confluence - <https://higproject.atlassian.net/secure/Dashboard.jspa>

Face Database: <http://cswww.essex.ac.uk/mv/allfaces/index.html>

Project Email: hig_project@outlook.com

2.1.2 OpenCV Based Initial Face Detection Prototype Development Tools

Programming Language: Python 3.6 - <https://www.python.org/downloads/release/python-360/>

Computer Vision Framework: OpenCV - <https://opencv.org/releases.html>

Integrated Development Environment (IDE): Spyder - <https://pythonhosted.org/spyder/>

2.1.3 Android Application and Vision API Development Tools

Programming Language: Java (Business Logic), XML (User Interface)

Integrated Development Environment: Android Studio -

<https://developer.android.com/studio/index.html>

Computer Vision API: Google Mobile Vision -

<https://developers.google.com/vision/introduction>

Webserver: 000Webhost - <https://www.000webhost.com/>

The 000Webhost service is a free webhosting service which stores the photos uploaded from the face detection application.

Email: hiq_project@outlook.com

Password: Hiqproject

2.2 How to Execute Project Code

The following section outlines how to go about executing the code for both the initial face detection prototypes and the final Android application. **NOTE:** It is assumed that the necessary development tools as stated in the previous section have been downloaded and installed.

2.2.1 OpenCV Based Initial Face Detection Prototypes

The Viola-Jones and Cascade Classifier algorithms were originally prototyped using Python 3.6 and OpenCV 3. Within the repository, these prototypes can be found within the 'OpenCV Face Detection Prototype Algorithms' directory. This folder contains the Python code for the viola-jones algorithm (viola_jones.py) and the cascade classifier algorithm (cascade_classifier.py).

To execute the viola_jones.py code, perform the following:

1. Open a command line terminal
2. Navigate to the path which contains the viola_jones.py file, the haarcascade_frontalface_default.xml file and the image files you want to test the code with
3. In the command line window type: `python viola_jones.py Images/{name_of_image_file}.{file type} haarcascade_frontalface_default.xml`
4. Example: `python viola_jones.py Images/2.jpg haarcascade_frontalface_default.xml`

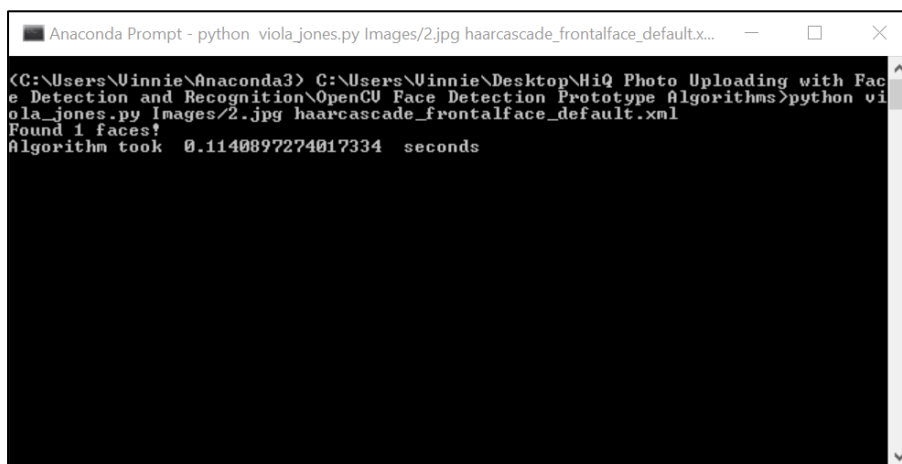
To execute the cascade_classifier.py code, perform the following:

1. Open a command line terminal

2. Navigate to the path which contains the cascade_classifier.py file, the haarcascade_frontalface_default.xml file, the haarcascade_eye.xml and the image files you want to test the code with
3. In the command line window type: python cascade_classifier.py Images/{name_of_image_file}.{file type} haarcascade_frontalface_default.xml haarcascade_eye.xml
4. Example: python cascade_classifier.py Images/2.jpg haarcascade_frontalface_default.xml haarcascade_eye.xml

The screenshots below display the command line window and sample output when the viola-jones.py file and the cascade_classifier.py file is run.

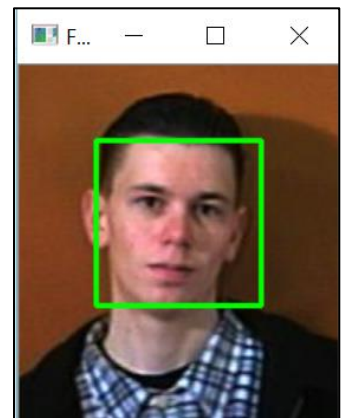
Execution of the viola_jones.py file:



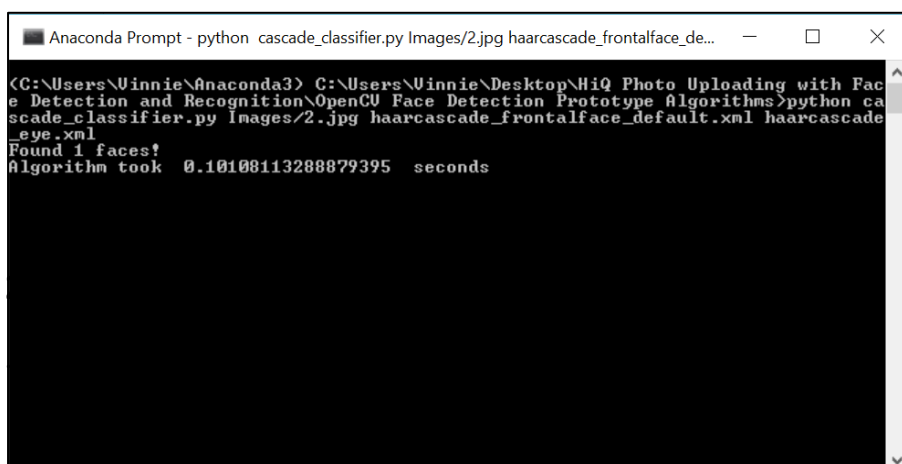
```

Anaconda Prompt - python viola_jones.py Images/2.jpg haarcascade_frontalface_default.x...
<C:\Users\Vinnie\Anaconda3> C:\Users\Vinnie\Desktop\HiQ Photo Uploading with Face Detection and Recognition\OpenCV Face Detection Prototype Algorithms>python viola_jones.py Images/2.jpg haarcascade_frontalface_default.xml
Found 1 faces!
Algorithm took 0.1140897274017334 seconds

```



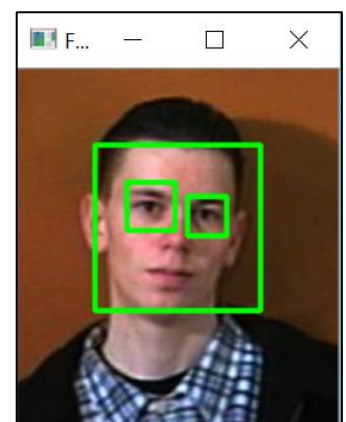
Execution of the cascade_classifier.py file:



```

Anaconda Prompt - python cascade_classifier.py Images/2.jpg haarcascade_frontalface_de...
<C:\Users\Vinnie\Anaconda3> C:\Users\Vinnie\Desktop\HiQ Photo Uploading with Face Detection and Recognition\OpenCV Face Detection Prototype Algorithms>python cascade_classifier.py Images/2.jpg haarcascade_frontalface_default.xml haarcascade_eye.xml
Found 1 faces!
Algorithm took 0.10108113288879395 seconds

```



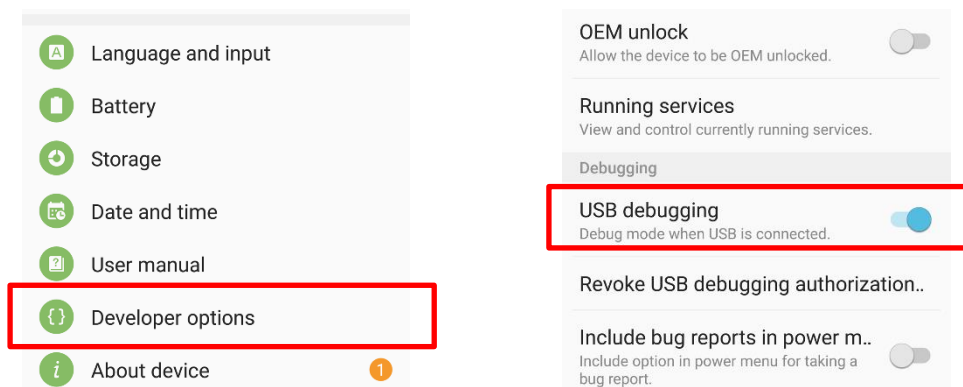
2.2.2 Face Detection Android Application

The Face Detection Android Application was developed using the Java programming language and the Android Studio IDE.

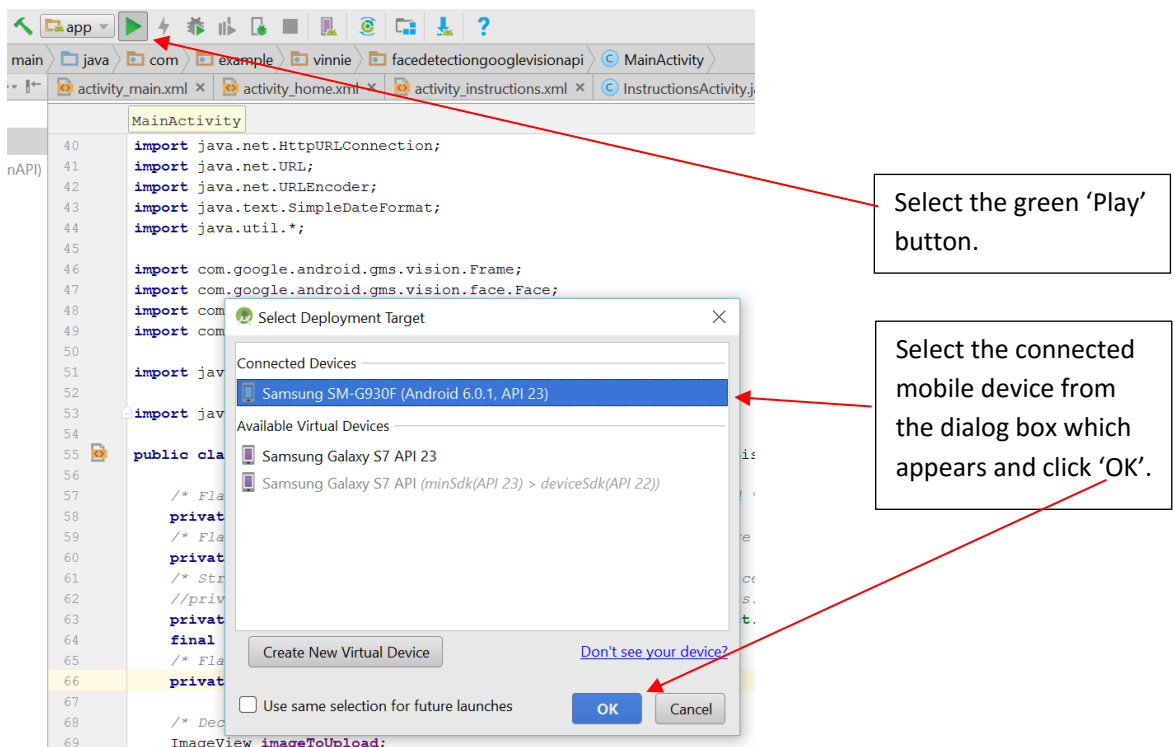
NOTE: The application was designed and tested on a Samsung Galaxy S7 running the Android 6.0 (Marshmallow) Operating System. This application was not tested on any other Android device or any other Android Operating system meaning that there may be issues when this application is run on any other device.

Within the repository, the application can be found within the 'FaceDetectionGoogleVisionAPI' directory. This folder contains all of the source code and libraries which make up the entirety of the application. Assuming that Android Studio has been downloaded and installed, import this folder into the Android Studio workspace.

In order to run the application on an Android mobile device, connect a microUSB cable from the computer to the mobile device. On the Android device go to Settings > Developer Options > USB Debugging. Make sure that the USB Debugging is toggled on. The Developer Options may not appear within the device settings automatically. In this case, research online how to have Developer Options appear in the Android device settings as the process can be different for each mobile device. The images below illustrate the Developer Options and the USB Debugging toggle on a Samsung Galaxy S7.

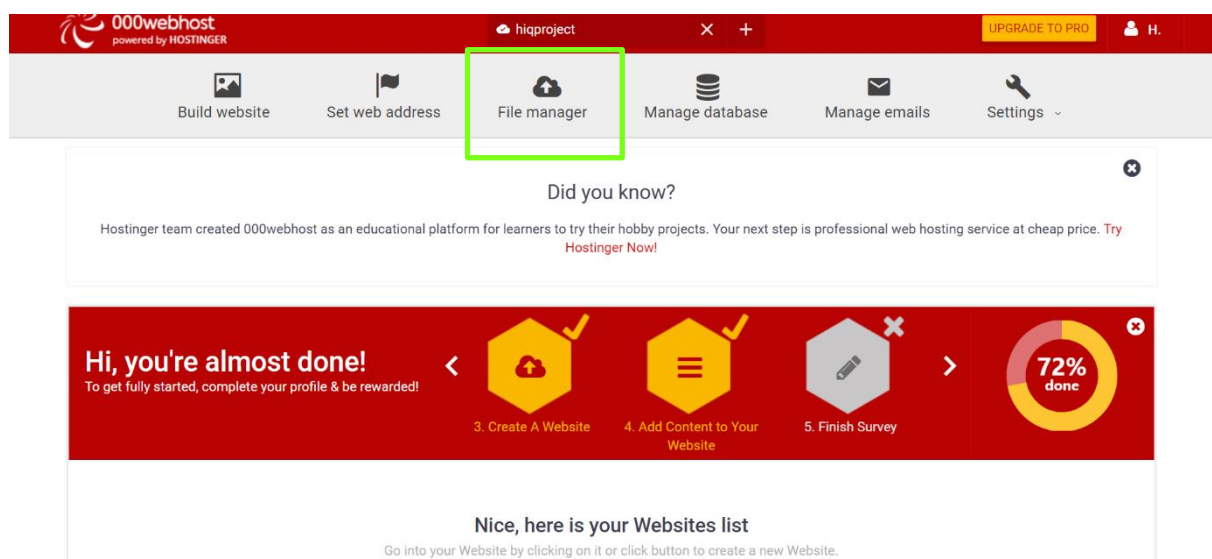


Within Android Studio and with the project currently open, select the green Play button on the toolbar. If the mobile device is currently connected to the computer and USB Debugging is currently activated then the name of the device will appear in a dialog box. Select the mobile device and selection 'OK'. At this point, the Face Detection Application should successfully load onto the device and an icon for the application will appear amongst the list of applications on the device. The image below depicts the process of running the application and selecting the connected mobile device from the dialog box.

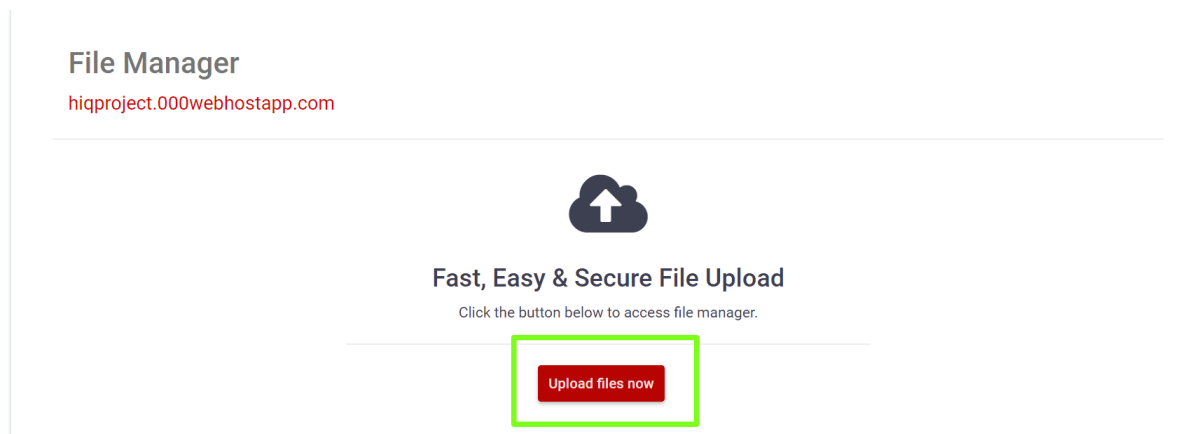


2.2.3 How to View Uploaded Photos in 000Webhost

When photos are uploaded from the application, they are sent to an online webserver called 000Webhost. Navigate to the following website: <https://www.000webhost.com/>. Login using the credentials as seen in Section 2.1.3. Once logged in, select the 'File Manager' tab near the top of the webpage and as seen in the image below.



Then select the 'Upload files now' button as seen in the screenshot below.

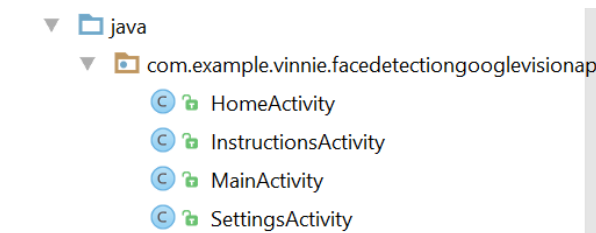


At this point, the browser will redirect and show the webserver file storage. Select the 'pictures' folder to view the images uploaded from the application as illustrated below.

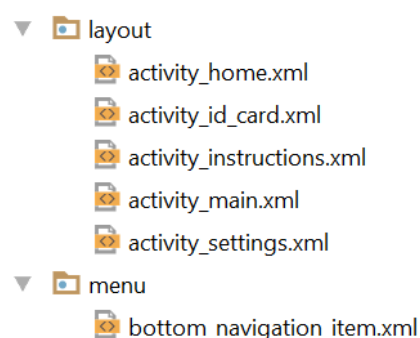


2.3 Face Detection Android Application Structure

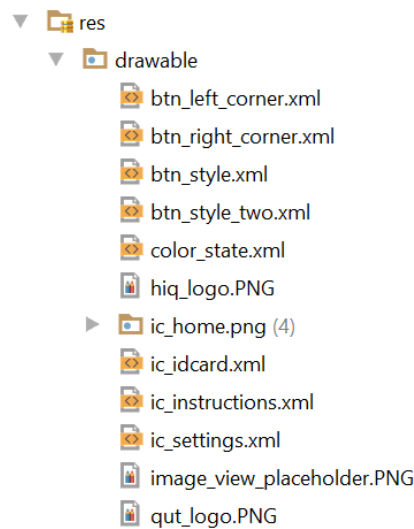
This section explains the overall structure of the Face Detection Android Application. The application has been split up into four activities which correspond to each different screen in the app. Namely the Home, Instructions, ID Photo and Settings screens. The image below illustrates the structure of these Java files which can be seen within the Project Structure sidebar in Android Studio. It should be noted that the 'MainActivity.java' file contains all of the functionality related to the ID Photo screen.



Each Java activity file comes equipped with an associated Layout XML file which contains all of the code corresponding to the design of the user interface for that activity. The XML language is used to create user interfaces in this project, however Android Studio comes equipped with a GUI builder so buttons and containers can be dragged onto the screen which automatically generates the associated XML code. The XML code within these files is relatively straight forward and self-explanatory and corresponds to every button, image view, toolbar etc. present on an activity's screen. The image below illustrates the file structure of the 'layout' folder containing the XML files for each activity. A separate 'menu' file is also pictured which contains the XML code used to design the bottom navigation toolbar containing the 'Home' button on each screen within the application.



A separate 'drawable' folder is used to store all of the XML code used to design specific shapes and colours for the buttons in the application. Furthermore, image files for the QUT and HiQ logos are also stored in this directory which can be seen in the image below.



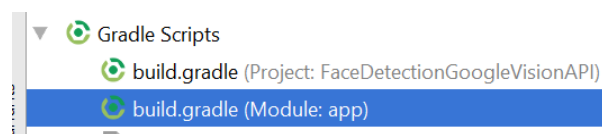
To perform the functionality in the application to use the device camera and save or upload photos from the device storage, certain permissions are required. The Face Detection Android Application project contains an 'AndroidManifest.xml' file. This file contains the lines of code which request these permissions. Lines 5-8 in the screenshot below show the permissions requested for the Face Detection Android Application. These include permissions to access the device camera, read from the device external storage and to access the internet. No other code in this file should need to be changed.

```

5      <uses-feature android:name="android.hardware.camera" />
6
7      <uses-permission android:name="android.permission.INTERNET" />
8      <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

```

The Face Detection Android Application relies on a number of dependencies in order to implement its functionality. For example, as the application uses the Google Vision API, a dependency has been added so that the API can be used. Each dependency can be seen in the 'App build.gradle' file which can be seen highlighted in blue within the image below.



Whilst there is no need to touch most of the code in this file, if dependencies change or need to be added or deleted then simply add the dependency within the 'dependencies' brace as seen within lines 26-37 in the image below.

```

26      dependencies {
27          compile fileTree(include: ['*.jar'], dir: 'libs')
28          androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
29              exclude group: 'com.android.support', module: 'support-annotations'
30          })
31          //compile project(':openCVLibrary320')
32          compile 'com.android.support:appcompat-v7:25.3.1'
33          compile 'com.android.support.constraint:constraint-layout:1.0.2'
34          compile 'com.google.android.gms:play-services-vision:9.8.0'
35          compile 'com.android.support:design:25.1.1'
36          testCompile 'junit:junit:4.12'
37      }

```

2.4 Face Detection Android Application Code

All of the code within the project has been clearly commented so future developers should be able to read and understand the purpose of each function. The most important function which will be covered in this document is the 'onClick()' function displayed within the 'MainActivity.java' file between lines 134-215. This function contains a switch case statement corresponding to each button within the ID Photo screen.

The functionality for the camera is began upon pressing the 'Capture Photo' button. An Android Camera Intent is used here to call upon the existing camera application to capture a photo. The photo is placed into a file and saved within the mobile device storage with the name being the current device date and time.

Similarly, when uploading a photo from the image library present on the device, an Android Gallery Intent is used to access the gallery application in order to choose an image. This is done when a user presses the 'Choose Photo' button.

The functionality to upload an image is began when a user presses the 'Submit' button. First the ImageView is checked to see if it contains an image. If it does, the image is scaled down to the size of an ID photo and passed into a separate class to handle uploading the photo to a webserver. Line 63 within the 'MainActivity.java' file holds a server address as depicted below:

```
63      private static final String SERVER_ADDRESS = "https://hiqproject.000webhostapp.com/";
```

If the connection to the webserver is to be changed so that the image is uploaded to a different server, then this constant string will need to be modified with the address for the new server.

Finally the face detection is began once a user selects the 'Process' button. This part of the switch case statement calls upon a number of helper functions. The 'performFaceDetection()' function calls upon the Google Vision API to execute the face detection. Here detected faces are saved into an array of Face objects. This array is passed into the 'checkFacialLandmarks()' helper function which scans each face in the array for each facial landmark present in the photo. If any of the expected landmarks are missing, then a custom error prompt is displayed to the user. Within this helper function, the 'storeFacialData()' function is called which stores all of the facial data attained from the detected face and may output further error messages based on this data. The image below shows some of the values which have been trial and errored to be used as thresholds for the probability of left and right eyes being open, facial orientation and face sizes. It is recommended that these values be experimented with further to find the most ideal threshold values.

```

337
338      /* Error messages based on the returned data */
339      if (leftEyeProb < 0.6 || rightEyeProb < 0.6) {
340          confidence -= 0.25;
341          Toast.makeText(MainActivity.this, "Submission Failed!", Toast.LENGTH_SHORT).show();
342          createDialog("Submission Failed!", "- Make sure both left and right eyes are open and visible in the image");
343      } else if (eulerY > 9 || eulerZ > 9 || eulerY < -9 || eulerZ < -9 ) {
344          confidence -= 0.25;
345          Toast.makeText(MainActivity.this, "Submission Failed!", Toast.LENGTH_SHORT).show();
346          createDialog("Submission Failed!", "- Make sure that your face is completely frontal and not tilted in any direction");
347      }
348
349      /*Error messages based on face size */
350      if (faceHeight < 600 || faceHeight > 1000) {
351          Toast.makeText(MainActivity.this, "Submission Failed!", Toast.LENGTH_SHORT).show();
352          createDialog("Submission Failed!", "Make sure your face in the center of the frame with the camera positioned 25-35cm away");
353      } else if (faceWidth < 500 || faceHeight > 800) {
354
355      }

```

Finally the facial data is printed to the console using the 'printFacialData()' function. The sample output of some facial data can be seen in the image below.

```

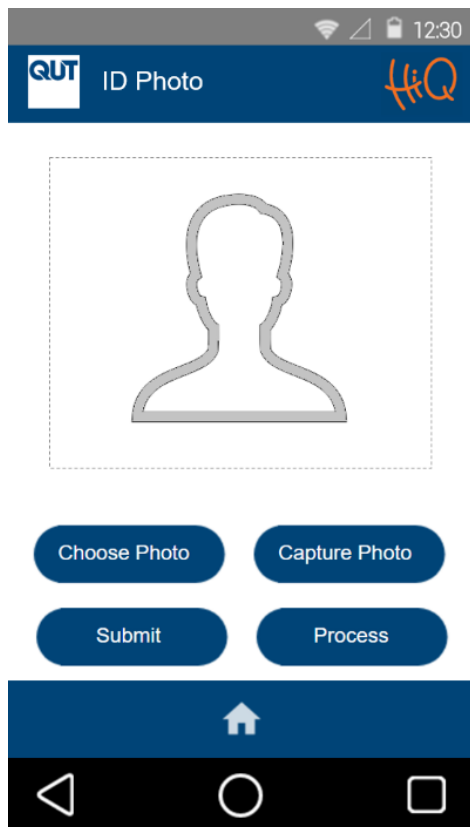
I/System.out: Height:2592
I/System.out: 2228
I/System.out: Width:1944
I/System.out: Face Height: 711.51434
I/System.out: Face Width: 569.2115
I/System.out: EulerY: 6.9030643
I/System.out: EulerZ 4.135371
I/System.out: Left Eye Probability: 0.9916368
I/System.out: Right Eye Probability: 0.9862527
I/System.out: Smiling Probability: 0.09607188
I/System.out: Confidence Level: 1.0

```

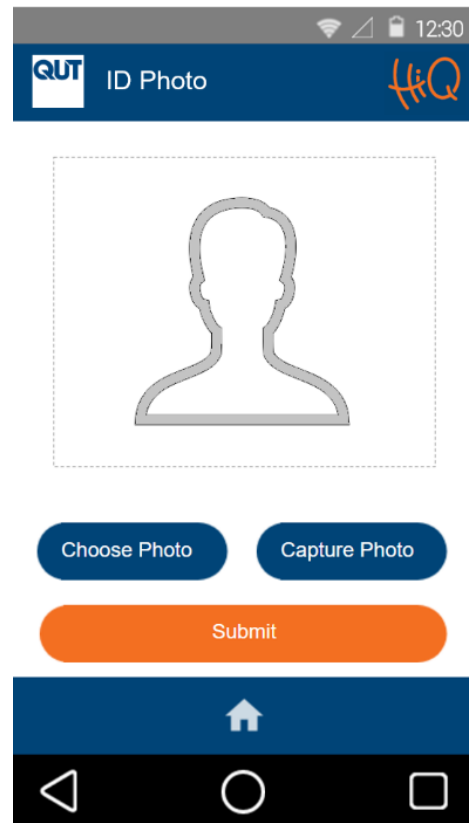
2.5 Current Limitations and Future Work

The following is a list of known limitations and issues within the current application. This also includes some future work which may need to be done.

- The application has only been developed and tested using a Samsung Galaxy S7 running a 64-bit Android 6.0 (Marshmallow) Operating System. As such issues may arise when ran on another Android device.
- Whilst the functionality should work on different Android devices, there is no guarantee that the layout and styling of the application will be consistent on different device screens. (i.e. buttons may appear out of position, image views may appear too large etc.)
- If a new update to the Android Operating System is released, this may cause some functionality to break. If this is the case, it is recommended to Google the new Android API for the new operating system and make the necessary changes to the code. It is likely that either no changes or a minor change will need to be implemented if an update to the Android OS causes certain features to stop working.
- Future development of this project may require that instead of uploading a photo to a web server hosted by 000Webhost, it may need to be sent to a server within HiQ at QUT. If this is the case, HiQ will need to be consulted to provide the address to their server and to explain how to connect to their server.
- A number of thresholds of acceptance for facial characteristics were hard coded into the Android application code base based on trial and error when testing a number of photos. Due to a lack of testing conducted on other Android devices, there are no guarantees that these thresholds may be the same when cameras of different qualities are used. Specifically, accepting or rejecting a photo based on the size of a face in an image or facial orientation may be affected due to the difference in pixels output by different cameras.
- Currently, toggling the Developer Mode switch in the Settings menu does not actually do anything. The Settings menu was merely placed within the application as a proof of concept. The idea was to have a toggle so that general users who are not developers of the application can use the app in User Mode. Within User Mode, the functionality to upload a photo to a webserver and conduct face detection should be combined. For example, when a user in User Mode selects the 'Submit' button on the ID Photo screen, the face detection should first be executed and only after passing should it begin uploading to the webserver. The idea behind this flow is so that users aren't actually aware of the face detection taking place. Furthermore, the red box and labels which normally appear on a detected face in Developer Mode will not be drawn onto the photo in User Mode as general users should not see this data. The mock ups below depict the intended difference between User Mode and Developer Mode.



Developer Mode – ID Photo Screen



User Mode – ID Photo Screen