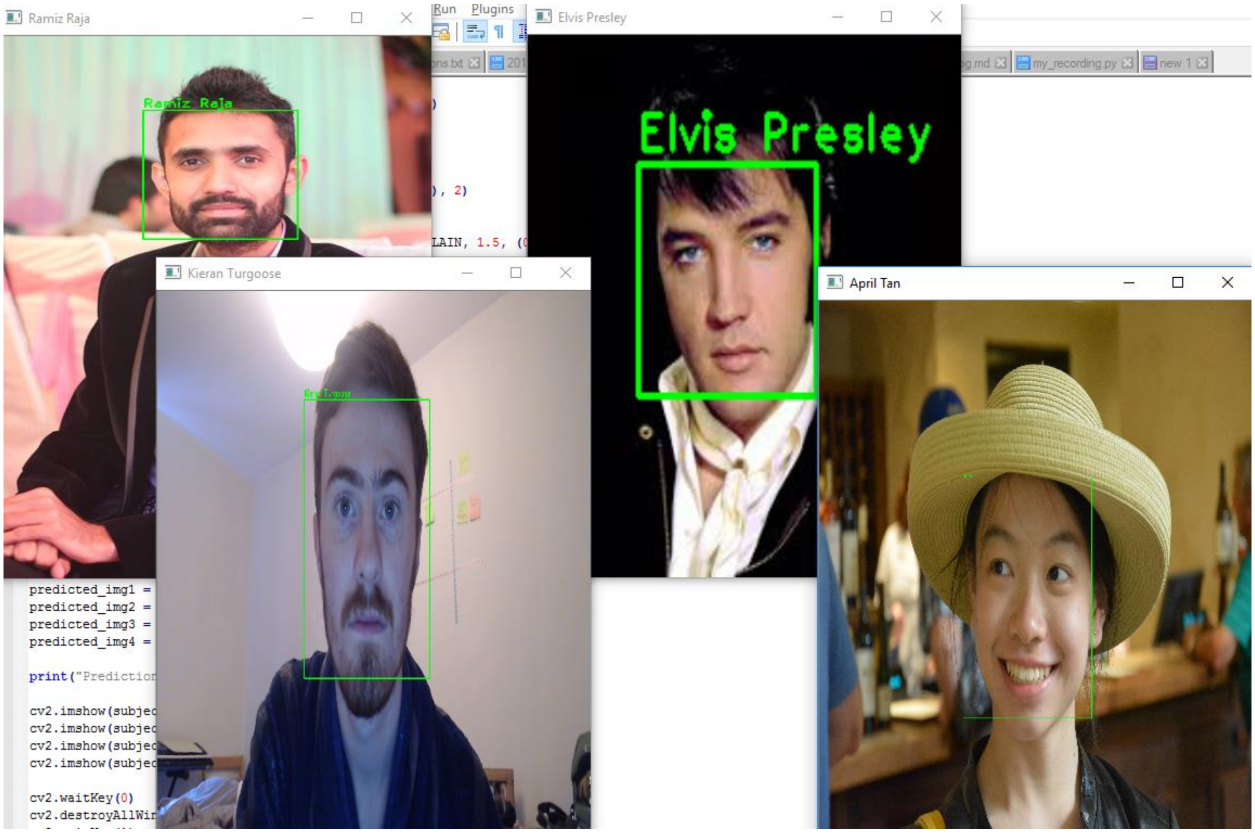
Testing

Facial Recognition Accuracy Tests



```
), 2)
LAIN, 1.5, (
```

```
predicted_img1 =
predicted_img2 =
predicted_img3 =
predicted_img4 =

print("Prediction

cv2.imshow(subjec
cv2.imshow(subjec
cv2.imshow(subjec
cv2.imshow(subjec

cv2.waitKey(0)
cv2.destroyAllWin
```

Results for facial recognition algorithm on test-set:

All four trained users correctly identified as can be seen by the names of the image windows above. This boasts a 100% accuracy for still images, albeit on a relatively small sample size. However, due to the nature of this project, PiMirror is a product that is intended for households. This will mean that the average sample size will often be around 4/5 inhabitants. Therefore, I look at these results with confidence that my system can be used as intended.

Climate-Based Testing for Python

TEST CASES FOR FACIAL RECOGNITION
General Prerequisites: - Software fully installed - User has a config file saved with module configuration - User group have their faces trained on the system - Internet connection - Webcam connected to the Raspberry Pi - Facial capture python program running

<table>
<tr><th colspan="3">Recognise Face in well-lit room</th></tr>
<tr><td colspan="3">Prerequisites: Well-lit room</td></tr>
<tr><td>Steps:</td><td>Expected Result</td><td>Actual Result</td></tr>
<tr><td>1. User A steps in front of the webcam</td><td>1. User A sees their name correctly display beside their face</td><td>Pass</td></tr>
<tr><th colspan="3">Recognise Face in dimly-lit room</th></tr>
<tr><td colspan="3">Prerequisites: Dimly-lit room</td></tr>
<tr><td>Steps:</td><td>Expected Result</td><td>Actual Result</td></tr>
<tr><td>1. User A steps in front of the webcam</td><td>1. User A sees their name correctly display beside their face</td><td>Pass</td></tr>
<tr><th colspan="3">Recognise Two separate users correctly</th></tr>
<tr><td colspan="3">Prerequisites: Well-lit room, both users' faces are trained on the system</td></tr>
<tr><td>Steps:</td><td>Expected Result</td><td>Actual Result</td></tr>
<tr><td>1. User A steps in front of the webcam
2. User A steps away from the webcam
3. User B steps in front of the webcam</td><td>1. User A sees their name correctly display beside their face
2. User B sees their name correctly display beside their face</td><td>Pass</td></tr>
<tr><th colspan="3">Ensure untrained user is not recognised</th></tr>
<tr><td colspan="3">Prerequisites: Well-lit room, User A's face is trained, User B's face is not trained</td></tr>
<tr><td>Steps:</td><td>Expected Result</td><td>Actual Result</td></tr>
<tr><td>1. User A steps in front of the webcam
2. User A steps away from the webcam
3. User B steps in front of the webcam</td><td>1. User A sees their name correctly display beside their face
2. User B sees no name display beside their face</td><td>Pass</td></tr>
</table>

Results for Climate-Based Facial Recognition Testing:

All designed test cases were passed, which is good. These results show that the facial recognition also works well in real-life and real-time scenarios. It is able to handle well-lit rooms, and also dimly-lit rooms. However, it was inconsistent when treated with a bright light from one side, so that half of the user's face was unrecognizable. I believe this is forgivable due to this projects main focus is that of a mirror, whereby it will often be utilized in a bathroom, and will therefore often be well-lit from the front.

Scenario-Based Testing for Python

TEST CASES FOR PYTHON CODE
General Prerequisites: - Software fully installed

- User group have a config file saved with module configuration - User group have their faces trained on the system - Internet connection - Webcam connected to the Raspberry Pi - Facial capture python program running		
Trained user stands in-front of mirror		
Prerequisites: Well-lit room		
Steps:	Expected Result	Actual Result
1. User A steps in front of the webcam	1. User A sees their name correctly display beside their face 2. Correct face count is shown to be increasing in console 3. Correct config file is loaded and displayed as mirror	Pass
Untrained user stands in front of mirror		
Prerequisites: Well-lit room		
Steps:	Expected Result	Actual Result
1. User A steps in front of the webcam	1. User A sees no name display beside their face 2. No mirror displays, just black background	Pass
Trained second user stands in front of the mirror		
Prerequisites: Well-lit room, both users' faces are trained on the system, User A has mirror loaded		
Steps:	Expected Result	Actual Result
1. User A step away from the webcam 2. User B steps in front of the webcam	1. User B sees their name correctly display beside their face 2. Correct face count is shown to be increasing in the console 3. User A mirror shuts off 4. User B mirror is displayed	Pass
Mirror turns off automatically		
Prerequisites: Well-lit room, User A configuration displayed on mirror		
Steps:	Expected Result	Actual Result
1. User A steps away from the webcam	1. After configured number of seconds' pass, mirror shuts off	Pass
User attempts to restart mirror		
Prerequisites: Well-lit room, User A's face is trained on the system, Mirror turns off automatically		
Steps:	Expected Result	Actual Result
1. User A step back in front of the webcam Branch:	1. User A sees their name correctly display beside their face 2. Correct face count is shown to be increasing in the console 3. User A mirror is displayed	Pass

1.B User B steps in front of the webcam	1. User B sees their name correctly display beside their face 2. Correct face count is shown to be increasing in the console 3. User B mirror is displayed	Pass
Second user stands adjacent to user		
Prerequisites: Well-lit room, both users' faces are trained on the system, User A has mirror loaded		
Steps:	Expected Result	Actual Result
1. User B steps in front of the webcam	1. User B should not be recognized enough for the mirror to switch configurations	Pass

Results:

All of these tests passed, and so I am happy that each of the potential scenarios in which this system could be used have been tested and found successful. This proves that the system is well-designed and well implemented to a high-standard, so that any scenario will be handled comfortably without the system failing or producing a drastically wrong outcome.

Scenario-Based Testing for Android Code

TEST CASES FOR ANDROID APPLICATION			
General Prerequisites: - Raspberry Pi powered on with connected webcam and monitor display. - Raspberry Pi has trained users xml file and Mirror software installed - Both the Pi and the Phone are connected to the same internet connection			
Sign In			
Steps:	Expected Result:	Result (Samsung):	Result (Pixel):
1. Click "Sign In" button 2. Select an account	1. Main Menu activity loads	Pass	Pass
Connect To Pi			
Prerequisites:			
Steps:	Expected Result:	Result (Samsung):	Result (Pixel):
1. Click "Connect To Pi" 2. Enter valid IP address + password 3. Click "Save"	1. Main Menu activity loads 2. Toast with "Connection Successful" displayed	Pass	Pass
Branch: 2B. Enter invalid login credentials.	1. Main Menu activity loads?	Pass	Pass

<table>
<tr><td></td><td>2. Toast with “Connection Unsuccessful” displayed.</td><td></td><td></td></tr>
<tr><td colspan="4"><b>Turn on Mirror w/no modules selected</b></td></tr>
<tr><td colspan="4">Prerequisites: Sign In, Connect To Pi, no weather API saved, no modules selected</td></tr>
<tr><td>Steps:</td><td>Expected Result:</td><td>Result (Samsung):</td><td>Result (Pixel):</td></tr>
<tr><td>1. Click “Save”
2. Click “Turn On Mirror”</td><td>1. Mirror starts up and shows blank display.
2. Config file started is correct signed-in user.</td><td>Pass</td><td>Pass</td></tr>
<tr><td colspan="4"><b>Turn off Mirror</b></td></tr>
<tr><td colspan="4">Prerequisites: Sign In, Connect To Pi, Mirror display is on</td></tr>
<tr><td>Steps:</td><td>Expected Result:</td><td>Result (Samsung):</td><td>Result (Pixel):</td></tr>
<tr><td>1. Click “Turn off Mirror”</td><td>1. Mirror display disables</td><td>Pass</td><td>Pass</td></tr>
<tr><td colspan="4"><b>Save Weather API</b></td></tr>
<tr><td colspan="4">Prerequisites: Sign In</td></tr>
<tr><td>Steps:</td><td>Expected Result:</td><td>Result (Samsung):</td><td>Result (Pixel):</td></tr>
<tr><td>1. Click “Weather API”
2. Enter API key from “openweathermap.org”
3. Click “Save”</td><td>1. API activity loads
2. Toast with “API saved” displayed
3. Main menu activity loads</td><td>Pass</td><td>Pass</td></tr>
<tr><td colspan="4"><b>Save Modules</b></td></tr>
<tr><td colspan="4">Prerequisites: Sign In, Connect To Pi, Save Weather API, No modules selected</td></tr>
<tr><td>Steps:</td><td>Expected Result:</td><td>Result (Samsung):</td><td>Result (Pixel):</td></tr>
<tr><td>1. Enter a module in each of the 6 grid spaces
2. Enter additional information on modules which require it
3. Click “Save”
4. Click “Turn on Mirror”</td><td>1. Toast with “Configuration saved” displayed
2. Toast with “Mirror enabled” displayed
3. Correct modules displayed on Mirror display</td><td>Pass</td><td>Pass</td></tr>
<tr><td colspan="4"><b>Sign Out</b></td></tr>
<tr><td colspan="4">Prerequisites: Sign In</td></tr>
<tr><td>Steps:</td><td>Expected Result:</td><td>Result (Samsung):</td><td>Result (Pixel):</td></tr>
<tr><td>1. Click “Sign Out”</td><td>1. Toast with “Signed out” displays
2. Login Activity loads</td><td>Pass</td><td>Pass</td></tr>
<tr><td colspan="4"><b>Display Saved Settings on Login</b></td></tr>
<tr><td colspan="4">Prerequisites: Saved Modules, Sign Out</td></tr>
<tr><td>Steps:</td><td>Expected Result:</td><td>Result (Samsung):</td><td>Result (Pixel):</td></tr>
<tr><td>1. Click “Sign In”
2. Select an account</td><td>1. Main menu activity loads
2. Previously saved module configuration is shown</td><td>Pass</td><td>Pass</td></tr>
</table>

<table>
<tr><th colspan="4">Reload mirror with a different configuration</th></tr>
<tr><td colspan="4">Prerequisites: Sign In, Connect To Pi, Saved Modules, Mirror Display on, Saved Weather API</td></tr>
<tr><td>Steps:</td><td>Expected Result:</td><td>Result (Samsung):</td><td>Result (Pixel):</td></tr>
<tr><td>1. Edit selected modules to a different set of modules.
2. Click “Save”
3. Click “Turn Off Mirror”
4. Click “Turn On Mirror”</td><td>1. Toast with “Configuration Saved” displayed
2. Toast with “Mirror disabled” displayed
3. New Mirror configuration correctly displayed</td><td>Pass</td><td>Pass</td></tr>
<tr><th colspan="4">Turn Mirror on when in-use</th></tr>
<tr><td colspan="4">Prerequisites: Sign In, Connect To Pi, Mirror Display on w/2nd user’s displays, Saved Weather API</td></tr>
<tr><td>Steps:</td><td>Expected Result:</td><td>Result (Samsung):</td><td>Result (Pixel):</td></tr>
<tr><td>1. Click “Turn On Mirror”</td><td>1. Same mirror display reloads
2. New user’s config is not shown as mirror is in use by 2nd user.</td><td>Pass</td><td>Pass</td></tr>
<tr><th colspan="4">Show Graphs</th></tr>
<tr><td colspan="4">Prerequisites: Sign In, Connect To Pi, Mirror Display Crypto graphs displayed, Saved Weather API</td></tr>
<tr><td>Steps:</td><td>Expected Result:</td><td>Result (Samsung):</td><td>Result (Pixel):</td></tr>
<tr><td>1. Click “Show Graphs”
2. Click “Save”
3. Click “Turn off Mirror”
4. Click “Turn on Mirror”</td><td>1. Checkbox is unchecked
2. Mirror interface is turned off
3. New display is started, Crypto module has default layout, no graphs</td><td>Pass</td><td>Pass</td></tr>
</table>

Results:

All of these tests passed, and so I am happy that the Android code is well designed and robust enough to handle all scenarios that may occur. To add another level of testing, I installed the app on another phone with a different version of Android, and re-tested all of these tests. This is shown above as there are two results columns, one for Samsung and one for the Google Pixel.