



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Final Viva

By Samuel Groves

Content

- ▶ Research Question
- ▶ Aims
- ▶ Objectives
- ▶ Literature Review Findings
 - ▶ Secondary Research Findings
 - ▶ Primary Research Findings
- ▶ Application Demonstration
- ▶ Development Justification
- ▶ Test Strategies
- ▶ Application results/conclusions
- ▶ Recommendations for future work
- ▶ Any Questions?



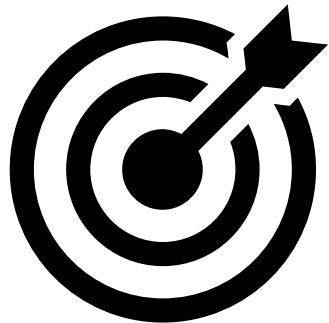
Research Question

‘The use of facial recognition software to improve security
within an academic/educational setting’



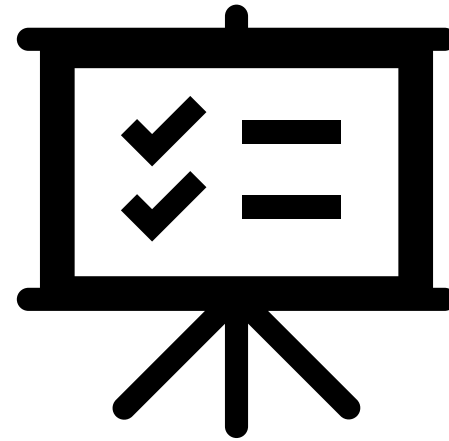
Aims

- Assess how FR software can be used to improve security of current systems within an Academic setting.
- Test and evaluate a designed solution within a suitable setting and obtain feedback where necessary.

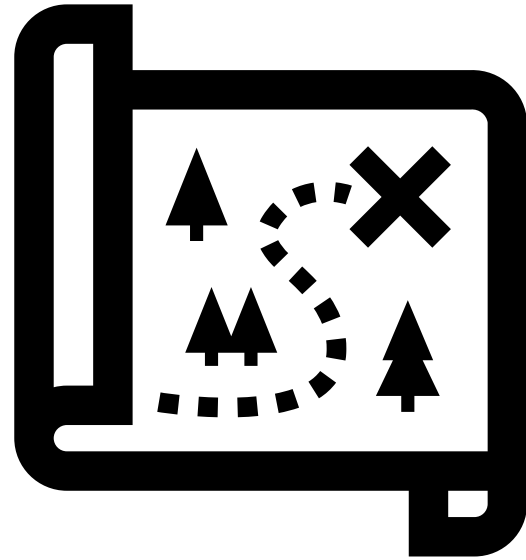


Objectives

- ▶ Ethical Considerations
- ▶ Security Considerations
- ▶ Data generated
- ▶ First-hand statistics
- ▶ Benefits & Drawbacks of FR
- ▶ Design, Evaluate & Test solution



Research Findings





UNITED STATES

More than **50%** of all Americans are currently in police facial recognition databases.



BELGIUM

Belgium is so far the only country to find facial recognition in breach of national law.



TURKEY

The Turkish military recently purchased **30 kamikaze drones** with facial recognition capabilities.



CHINA

China will account for an estimated **45%** of the facial recognition technology market by 2023.



UGANDA

The Ugandan government recently spent **\$126 million** on facial recognition CCTV technology from Huawei.



BRAZIL

Facial recognition technology is currently implemented in at least **37 cities** throughout Brazil.

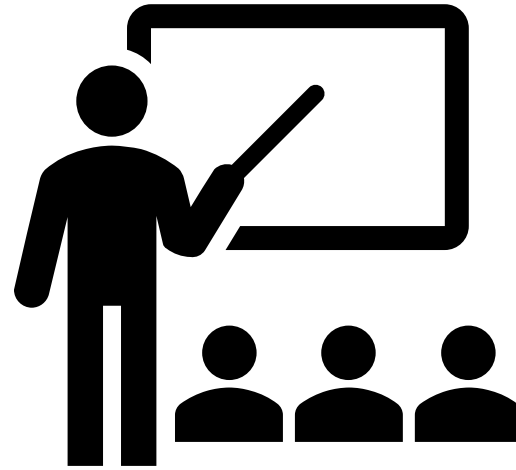


Literature Review Findings

Secondary Research

Facial Recognition within education.

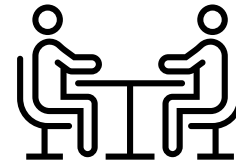
- ▶ United States Early Adopter
- ▶ Evidence of Use within the UK
- ▶ Controversy within the UK



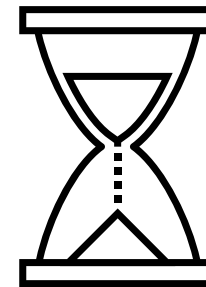
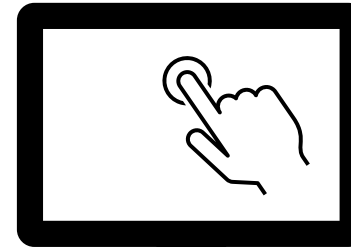
Literature Review Findings

Primary Research

Interviews - College Safety & Security Manager



- ▶ New touch screen sign-in system
- ▶ Long time scale
 - ▶ Must follow legislation
 - ▶ Should be able to integrate with other technologies.
- ▶ New technology - large range of ethical considerations
- ▶ Only Students & Contractors - NOT STAFF!

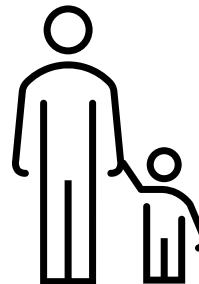
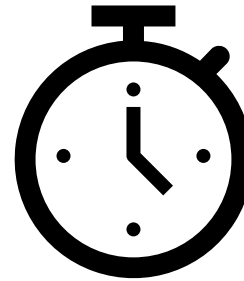


Literature Review Findings

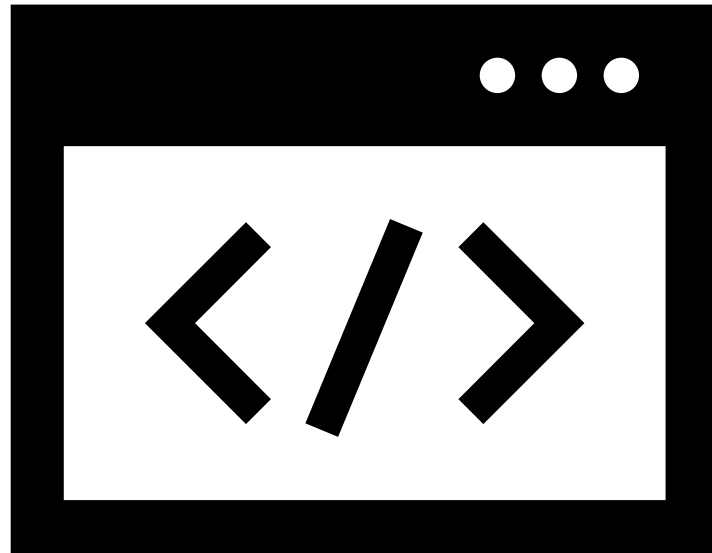
Concerns

Concerns regarding FR technology - Primary and secondary research findings

- ▶ Intrusion
 - ▶ Staff feeling clock-watched
- ▶ Ethical Concerns
 - ▶ Restrictive & Controlling
- ▶ Legislation
 - ▶ Data protection
 - ▶ The Equality Act 2010
- ▶ Student Age Related Concerns



Application/Solution



Facial Recognition Application Demo

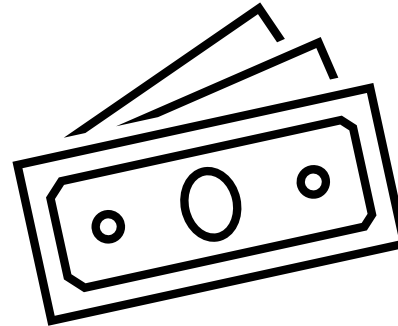
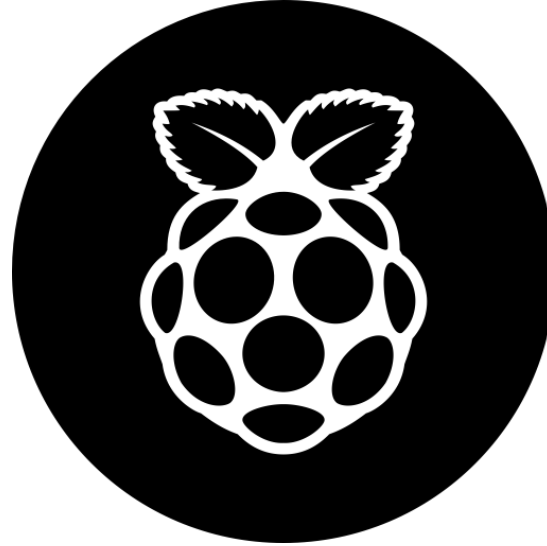
By Sam Groves



Development Justification

Hardware

- ▶ Why the Raspberry Pi 4gb?
 - ▶ Inexpensive
 - ▶ Relatively powerful for cost
 - ▶ Pre-packaged operating system
 - ▶ Arduino?



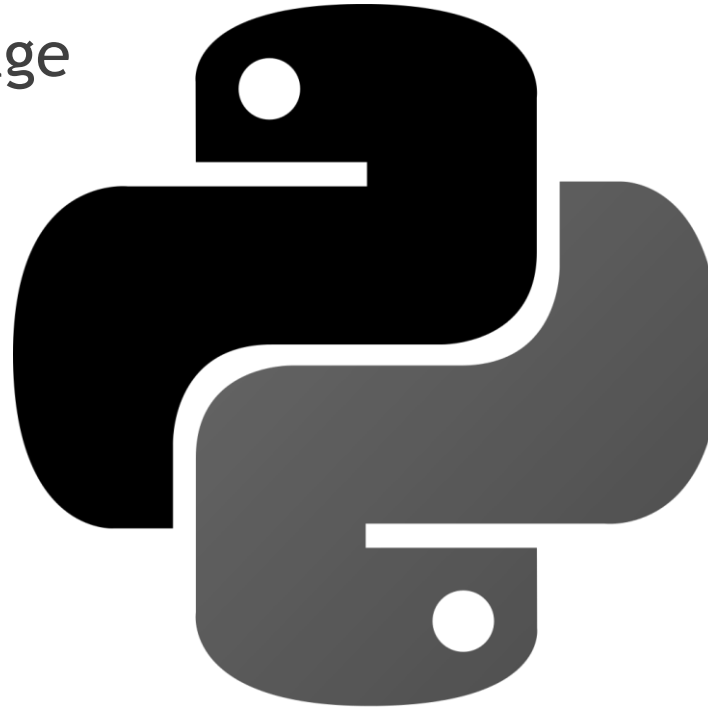
Development Justification

Software & Programming Language

- ▶ Why Python?

Frameworks & Paradigms

- ▶ PyQt
- ▶ OpenCV
- ▶ Dlib
- ▶ Facial Recognition Algorithm



Application Testing Strategies

What was tested?

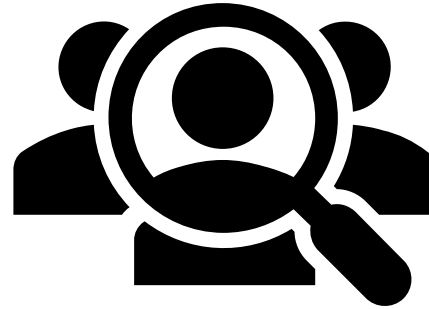
- ▶ Users
- ▶ Accuracy of Facial Recognition Algorithm

Intrinsic Factors	Extrinsic Factors
Aging	Occlusion
Facial Surgery	Background Noise
Facial Expression	Illumination
	Pose Variation
	Resolution

Application Testing Strategies - Users

Who are the users?

- ▶ Broad range of users
 - ▶ Age
 - ▶ Skin Colour
 - ▶ Facial Expression
 - ▶ Facial Surgery
- ▶ Students
- ▶ Contractors
- ▶ Staff (Initially)



Application Testing Strategies

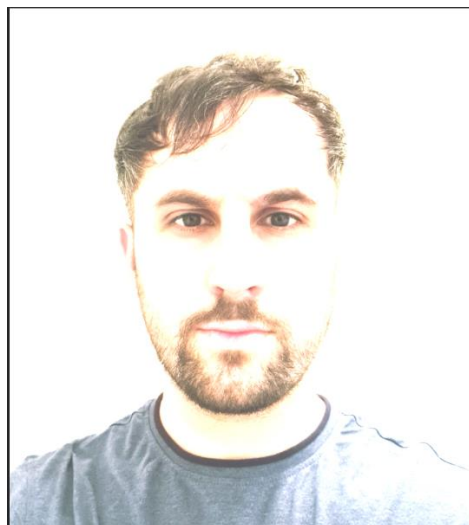
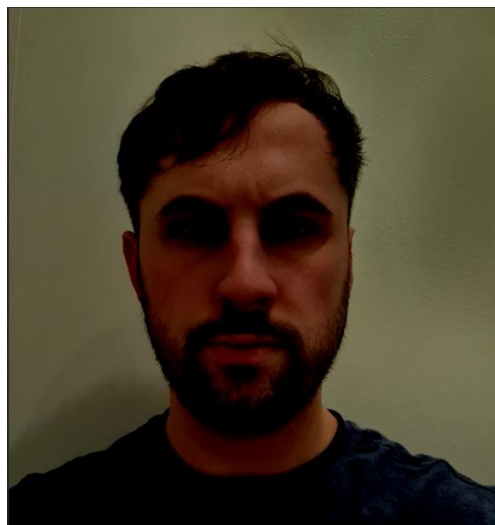
Example Test

1.



Resolution of known
user image Test

2.

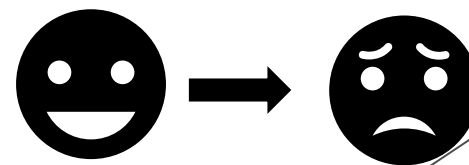
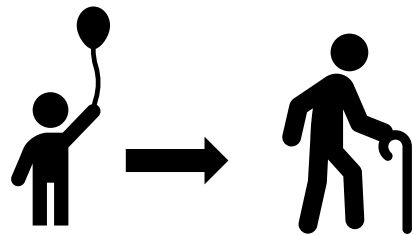


Illumination Test

Application Results

What were the results of the **Intrinsic** tests conducted?

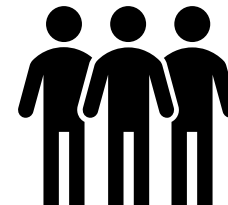
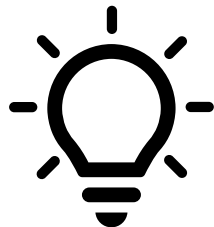
Intrinsic Factors	Result
Aging	Unaffected (recommended to keep up-to-date photos of known users)
Facial Surgery	N/A unable to test
Facial Expression	Did not affect accuracy



Application Results

What were the results of the **Extrinsic** tests conducted?

Extrinsic Factors	Result
Occlusion	User Unidentified
Background Noise	Huge difficulty (incorrect user logged)
Illumination	Highly illuminated image unrecognised
Pose Variation	Issues occurred, recommendations made
Resolution	Image Under 720p proved problematic



Application Results

A range of other features were tested...

- ▶ Functionalities

- ▶ Enter button within application
- ▶ Correct GUI Display

- ▶ Logging of user

- ▶ Ensure log file is correct.

- ▶ Hardware testing

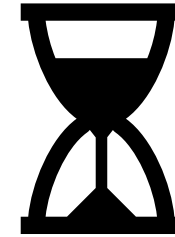
- ▶ Test Raspberry Pi identifies USB camera

```
pi@raspberrypi: ~/Facial_Recognition_App
File Edit Tabs Help
pi@raspberrypi:~ $ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 046d:0892 Logitech, Inc. OrbiCam
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Application Conclusions

Testing

- ▶ Errors found throughout tests were identified and addressed.
- ▶ Long load time with Application due to hardware

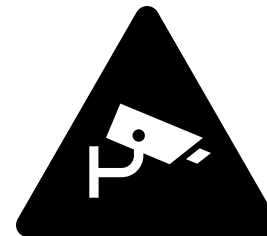


Implementation

- ▶ Simulated - Logistical issues

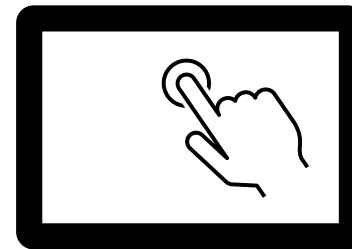
Does it improve security in Academic settings?

- ▶ Improves security under specific set of conditions (correct implementation)



Recommendations for Future Work

- ▶ Translate the application to Welsh or other languages (for specific environment)
- ▶ Implement application in real environment
 - ▶ Thorough legislation testing
- ▶ Assess upgrading Raspberry Pi 4
- ▶ Interactive Touchscreen Application (theme)
- ▶ Integration Testing



Any Questions?



References

1. Ghosh I. Mapped: The State of Facial Recognition Around the World [Internet]. Visual Capitalist. 2020 [cited 28 March 2022]. Available from: <https://www.visualcapitalist.com/facial-recognition-world-map/>
2. Groves S. The Use Of Facial Recognition Within Academia[Document]. Visual Capitalist. 2020 [cited 1 April 2022].

Code Examples - Script 1

```
1 #Facial Recognition Application for NPTC by Samuel Groves
2 #Imports Below
3
4 import sys
5 from PyQt5.uic import loadUi
6 from PyQt5 import QtWidgets
7 from PyQt5.QtCore import pyqtSlot
8 from PyQt5.QtWidgets import QApplication, QDialog
9 import resource
10 from out_window import Ui_OutputDialog
11
12 class Ui_Dialog(QDialog):
13
14     # Initialise loading of qt Main menu/window interface (Log in screen)
15
16     def __init__(self):
17         # Defined function for initialisation (Script Start)
18         super(Ui_Dialog, self).__init__()
19         loadUi("mainwindow.ui", self)
20         self.logolabel.show()
21
22         # Login/enter button below will run the defined function runCameraWindow
23
24         self.enterBtn.clicked.connect(self.runCameraWindow)
25         # Once button is clicked, load live camera feed based on defined function (line 35)
26
27         self._new_window = None
28         self.Videocapture_ = None
29         # Ensure single window is shown to user
30         # Ensure camera does not start within this script
31
32     def refreshAll(self):
33         # Defined function to reset video camera
34
35         self.Videocapture_ = "0"
36         # Direction of USB Camera port connected to Raspberry Pi
37
38     @pyqtSlot()
39     def runCameraWindow(self):
40
41         print("Student has Logged in")
42         self.refreshAll()
43         self.hide()
44         self.outputWindow_()
45         # Guidance for command prompt when user has clicked login/enter button
46         # Restart USB Camera (refresh)
47         # Main Menu is hidden here
48         # Create and open new output window (Live camera feed)
49
50     def outputWindow_(self):
51
52         self._new_window = Ui_OutputDialog()
53         self._new_window.show()
54         self._new_window.startVideo(self.Videocapture_)
55         print("Live Camera Feed Has Started")
56         # Connection to secondary script, initialises live camera feed interface
57         # Live camera feed window shows
58         # Start the live camera feed
59         # Guidance for command prompt when live camera feed has started
```

Code Examples - Script 2 (Initial)

```
23 class Ui_OutputDialog(QDialog):
24     def __init__(self):
25         super(Ui_OutputDialog, self).__init__()
26         loadUi("./outputwindow.ui", self)
27
28         now = QDate.currentDate()
29         current_date = now.toString('ddd dd MMMM yyyy')
30         current_time = datetime.datetime.now().strftime("%I:%M %p")
31
32         @pyqtSlot()
33
34
35
36     def startVideo(self, camera_name):
37
38         if len(camera_name) == 1:
39             self.capture = cv2.VideoCapture(int(camera_name))
40         else:
41             self.capture = cv2.VideoCapture(camera_name)
42         self.timer = QTimer(self)
43         path = 'FaceDatabase'
44         if not os.path.exists(path):
45             os.mkdir(path)
46         images = []
47         self.class_names = []
48         self.encode_list = []
49         self.TimeList1 = []
50         self.TimeList2 = []
51         students_list = os.listdir(path)
52
53
54
55         for cl in students_list:
56             cur_img = cv2.imread(f'{path}/{cl}')
57             images.append(cur_img)
58             self.class_names.append(os.path.splitext(cl)[0])
59         for img in images:
60             img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
61             boxes = face_recognition.face_locations(img)
62             encodes_cur_frame = face_recognition.face_encodings(img, boxes)[0]
63             encode = face_recognition.face_encodings(img)[0]
64             self.encode_list.append(encodes_cur_frame)
65         self.timer.timeout.connect(self.update_frame)
66         self.timer.start(10)
```

```
# Class which connects to initial script (main menu script)
# What to run when second script starts (live camera feed script)
#
# Load second GUI window live camera feed
#
# Variable 'now' is changed to the current date for future use within log file.
# Format for date when added to log file is defined here
# Format for time when added to log file is defined here
#
# Allows python to read specific elements within PyQt interface design (live camera feed label)
#
# Defined function to start video/live camera feed
#
# OpenCV configuration
# Locates USB camera port (directs to the USB path)
#
#
# Locates directory for images of known faces
#
#
# Configuration of images to match
# Disregard image name until required
# Further OpenCV configuration
#
#
# Ensures directory is found
#
# Directory found, Read images of known faces
#
# Allow images to be readable/understood by FR algorithm
#
# Camera configuration required by OpenCV
# Ensure camera presents RGB colour scheme (OpenCV uses BGR by default which cannot be read by algorithm)
# Location of box
# Creates frame for box
# Places box around specific facial features
#
# Initialises the update of each frame
#
```

Code Examples - Script 2 (Middle)

```
76 faces_cur_frame = face_recognition.face_locations(frame) # Connects Box to specific user
77 encodes_cur_frame = face_recognition.face_encodings(frame, faces_cur_frame) # Box location
78 for encodeFace, faceLoc in zip(encodes_cur_frame, faces_cur_frame): # Face recognition algorithm starts
79     match = face_recognition.compare_faces(encode_list_known, encodeFace, tolerance=0.50) # Defines the match variable. Tolerance relates to accuracy of algorithm
80     face_dis = face_recognition.face_distance(encode_list_known, encodeFace) # Estimate and understand distance of user
81     name = "unknown" # User name unknown till user identified
82     best_match_index = np.argmin(face_dis) # Find best match based on user distance
83
84
85
86
87 if match[best_match_index] and isActive == False: # If statement implemented for Users who are Identified
88     name = class_names[best_match_index].upper() # Name equals identified user image name
89     y1, x2, y2, x1 = faceLoc # Co-ordinates of face location
90     cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2) # OpenCV frame and box styles/properties
91     cv2.rectangle(frame, (x1, y2 - 20), (x2, y2), (0, 0, 255), cv2.FILLED) #
92     cv2.putText(frame, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 255, 255), 1) #
93     self.close() # Close live camera feed window as User is identified
94     print("Face has been found/recognised. User is accepted") # Guidance for command prompt to show a user has been accepted
95     loadUi("Accepted.ui", self) # Load PyQt GUI interface 'Accepted.ui' for identified user
96     isActive = True # isActive variable changed to true so that script stops updating
97     self.show() # Show 'Accepted.ui' interface
98
99
100
101 f = open("./logInfo.txt", 'w') # Defining log file information and path/directory
102 with open('logInfo.txt', 'w') as f: #
103     date_time_string = datetime.datetime.now().strftime("%d/%m/%y %H:%M:%S") # Add date and time to log file
104     f.writelines(f'\n {name},{date_time_string}') # Write to log file with name (ID) of accepted user (Name of image)
105
106 break # Break code here to ready restart
107
108
109
110 elif count > 20 and isActive: # Restart script after X seconds of accepted display shown.
111     os.execv(sys.executable, ['python'] + sys.argv) # Code to restart the entire application
112
113 elif count > 20 and not isActive: # If count is over 20, no face has been identified
114     self.close() # Close live camera feed window
115     loadUi("Declined.ui", self) # Load PyQt GUI interface 'Declined.ui' as no user is found
116     self.show() # Show 'Declined.ui' interface
117     self.timer.stop() # Stop live camera feed updating
118
119 return frame #
```

Code Examples - Script 2 (End)

```
121 def update_frame(self):
122     global count
123
124     ret, self.image = self.capture.read()
125     self.displayImage(self.image, self.encode_list, self.class_names, 1)
126     count = count + 1
127     #print(count)
128
129
130 def displayImage(self, image, encode_list, class_names, window=1):
131     image = cv2.resize(image, (640, 480))
132     try:
133         image = self.face_rec_(image, encode_list, class_names)
134     except Exception as e:
135         print(e)
136     qformat = QImage.Format_Indexed8
137     if len(image.shape) == 3:
138         if image.shape[2] == 4:
139             qformat = QImage.Format_RGBA8888
140         else:
141             qformat = QImage.Format_RGB888
142     outImage = QImage(image, image.shape[1], image.shape[0], image.strides[0], qformat)
143     outImage = outImage.rgbSwapped()
144
145     if window == 1:
146         self.imgLabel.setPixmap(QPixmap.fromImage(outImage))
147         self.imgLabel.setScaledContents(True)
```

```
# Defined function to update every frame of live camera feed when active

# Connection to Face recognition algorithm to keep 'scanning' for faces
# Further connection obtain image information for Face Recognition algorithm
# Each frame that updates, +1 is added to the count (over X seconds, produce outcome)
# Direction for command prompt during testing & Bug fixing

# Defined function to display live camera feed with PyQt GUI interface
# Set camera feed label size
#
# Exception if error occurs and camera feed cannot be displayed
# Guidance for command prompt when this error occurs
# Formatting of live camera feed label
#
# Ensure image is readable/formatted correctly
#
# Ensure live camera feed colour is correct

# Attach live camera feed to label
# Import QPixmap for images within resources
# Scale camera feed label to size property allocated
```

Initial Design - Main Menu (Initial)

**Welcome to the NPTC Facial
Recognition Scanner!**

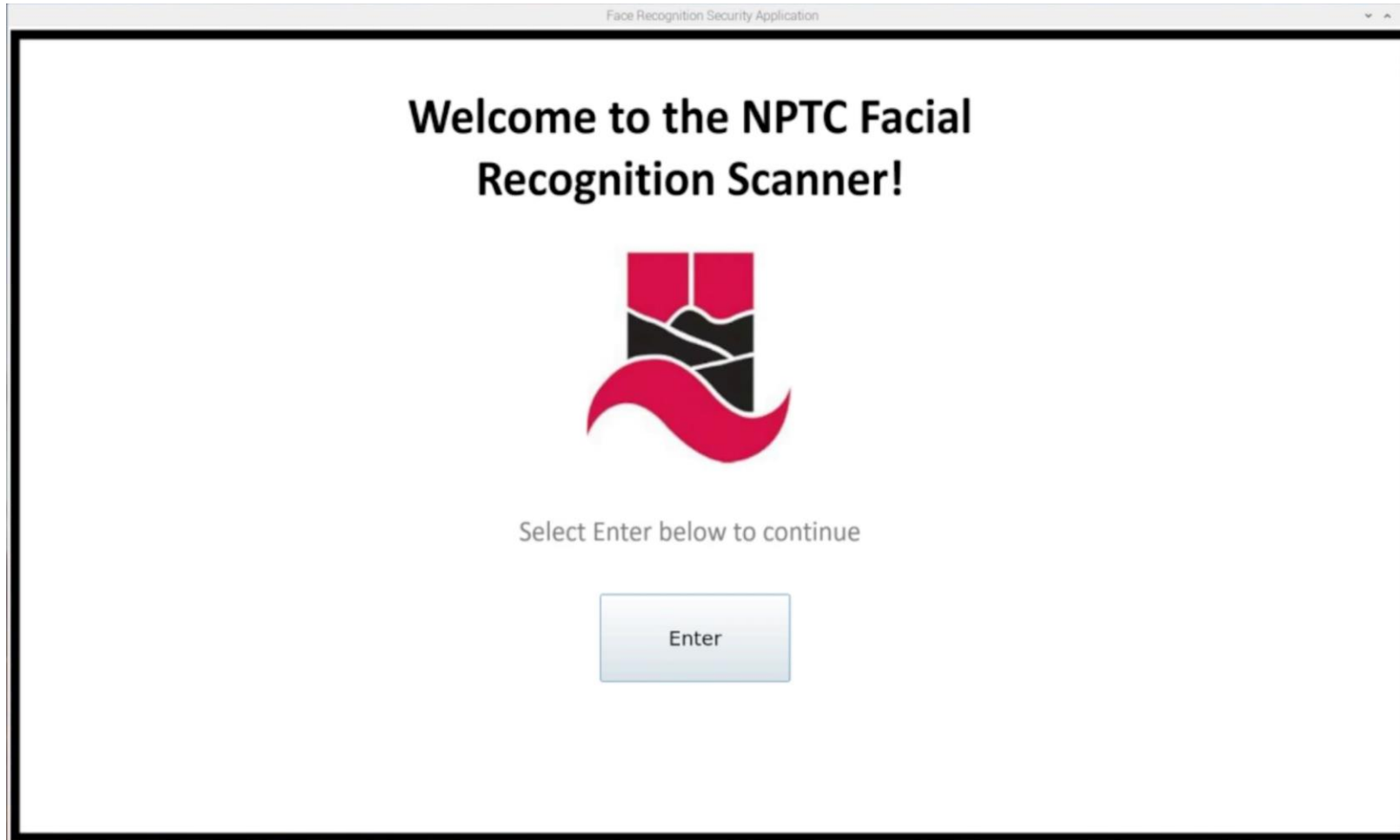


Please enter your student number below

Student Number:

Enter

Initial Design - Main Menu (Real)



Initial Design - Live Camera Feed (Initial)

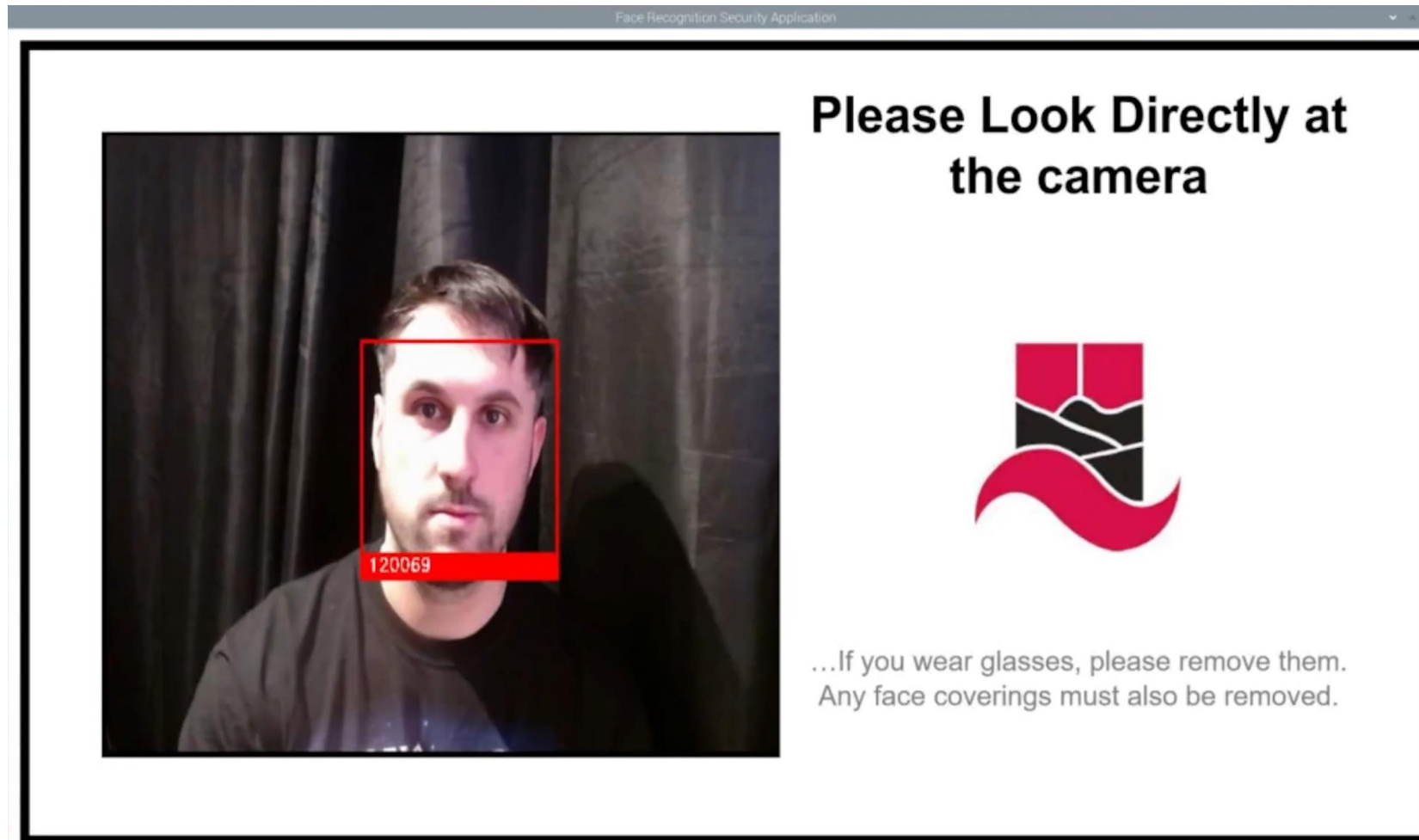


**Please Look Directly at the
camera**



...If you wear glasses, please remove them. Any
face coverings must also be removed.

Initial Design - Live Camera Feed (Real)



Initial Design - Accepted/Declined

You have been accepted



Welcome!

**Hmm... I cannot seem to recognise
you.**



**Please contact a nearby staff
member.**