

CRITERIA C

IMPORTS:

detect_faces_IP.py


















```
from imutils.video import VideoStream
import numpy as np
from ssl import SSLContext, PROTOCOL_TLSv1
from urllib.request import urlopen
import argparse
import imutils
import time
import cv2
```

Temp(temperature file)

```
import sys, string, os
import csv
```

FILES:

Initial files:

Name	Date modified	Type	Size
 real_time_object_detection_new	9/3/2020 4:27 PM	PY File	4 KB
 MobileNetSSD_deploy_new.caffemodel	9/3/2020 4:05 PM	CAFFEMODEL File	22,606 KB
 MobileNetSSD_deploy_new.prototxt	9/3/2020 4:05 PM	Text Document	31 KB
 MobileNetSSD_deploy.caffemodel	8/15/2020 1:32 PM	CAFFEMODEL File	22,606 KB
 real_time_object_detection	8/15/2020 1:31 PM	PY File	4 KB
 MobileNetSSD_deploy.prototxt	8/15/2020 1:31 PM	Text Document	31 KB
 webcam_model	8/4/2020 5:35 PM	PY File	3 KB
 main	8/4/2020 5:18 PM	PY File	4 KB
 res10_300x300_ssd_iter_140000.caffemodel	8/4/2020 4:47 PM	CAFFEMODEL File	10,417 KB
 deploy.prototxt	8/4/2020 4:46 PM	Text Document	28 KB
 webcam_in	7/24/2020 4:51 PM	PY File	1 KB
 detect_faces_IP	7/24/2020 4:28 PM	PY File	4 KB
 output	7/21/2020 3:43 PM	Video Clip	6 KB
 webcam_in	7/20/2020 5:06 PM	Text Document	1 KB
 detection_ideal	7/17/2020 5:14 PM	PY File	2 KB
 detection ideal	7/17/2020 5:13 PM	Text Document	0 KB
 webcam	7/17/2020 5:07 PM	PY File	1 KB

Face Detection:

Title	File type
deploy.prototxt.txt	Prototxt File
detect_faces_IP.py	Python File
res10_300x300_ssd_iter_140000.caffemodel	Caffemodel File

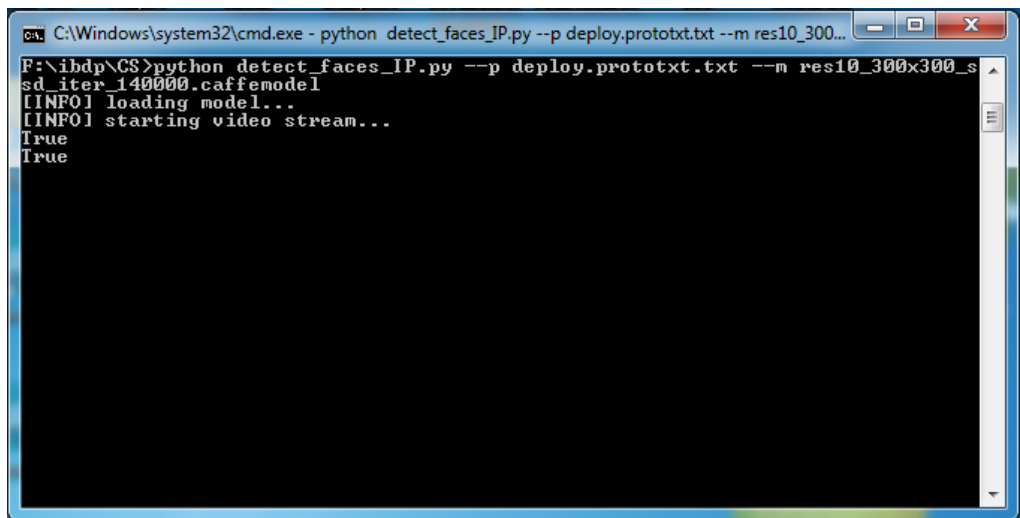
Measuring temperature

Title	File type
MlxCIRT.exe	Application
mlxcirt3_log.csv	CSV File
temppcsv.py	Python File

Dashboard

Title	File type
index(1).html	HTML File
script.js	JavaScript File
style.css	CSS File
graduate.png	PNG Image File

EXECUTION



```

C:\Windows\system32\cmd.exe - python detect_faces_IP.py --p deploy.prototxt.txt --m res10_300...
F:\ibdp\CS>python detect_faces_IP.py --p deploy.prototxt.txt --m res10_300x300_ssd_iter_140000.caffemodel
[INFO] loading model...
[INFO] starting video stream...
True
True

```

--p deploy.prototxt.txt

PROTOTXT files are serialized using Google's Protocol Buffers serialization library, and they contain:

- A list of the neural network layers a model contains
- Each layer's parameters, including its name, type, input dimensions, and output dimensions
- Specifications for connections between layers

Instance of layer and section with similarities:

```

layer {
  name: "conv9_2_mbox_loc_perm"
  type: "Permute"
  bottom: "conv9_2_mbox_loc"
  top: "conv9_2_mbox_loc_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}

layer {
  name: "conv9_2_mbox_loc_flat"
  type: "Flatten"
  bottom: "conv9_2_mbox_loc_perm"
  top: "conv9_2_mbox_loc_flat"
  flatten_param {
    axis: 1
  }
}

layer {
  name: "conv9_2_mbox_conf"
  type: "Convolution"
  bottom: "conv9_2_h"
  top: "conv9_2_mbox_conf"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 8 # 84
    pad: 1
    kernel_size: 3
    stride: 1
  }
}

```

```

        weight_filler {
            type: "xavier"
        }
        bias_filler {
            type: "constant"
            value: 0
        }
    }
}

```

detect_faces_IP.py

Arguments

These arguments are entered into the Common Prompt while entering the code

```

ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
    help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

```

This evidenced in the CMD input

```

F:\ibdp\CS>python detect_faces_IP.py --p deploy.prototxt.txt --m res10_300x300_s
sd_iter_140000.caffemodel

```

Initialising video stream

```

url1="rtsp://admin:1234567a@192.168.1.65:554/doc/page/preview.asp"
cap = cv2.VideoCapture(url1)
print(cap.isOpened())

```

Here, the url1 is the variable which is set to the webpage location of the video stream from the IP Camera. cap is a method that captures this video stream.

When the model has been loaded and the video stream has been started, the print lines are executed in the Command Prompt, as evidenced.

```

INFO1 loading model...
INFO1 starting video stream...

```

```

while(cap.isOpened()):

```

This is a while loop that ensures that the program runs as long as the stream is open.

Identifying faces

```
ret, frame = cap.read()
```

```
if ret == False:  
    break
```

The use of the if loop ensures that the next steps in the project only begin when a face is identified in the stream. If `ret == False` (a Boolean operator), the process goes forward/ Else, it breaks.

When a face has been identified, it also shows 'True' on the Command Prompt, as shown here.

```
True  
True
```

Blob

```
(h, w) = frame.shape[:2]  
blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 1.0,  
    (300, 300), (104.0, 177.0, 123.0))  
net.setInput(blob)  
detections = net.forward()
```

When the frame is detected in `frame = imutils.resize(frame, width=400)`, the frame is converted to a blob. A blob is a data type that can store binary data. As a result, it can store images or multimedia files.



This is a frame, which is shown as a blob.

```
net.setInput(blob)
detections = net.forward()
```

This connects the blob to other devices and sends data across the network. It also predicts whether there is a person in the frame or not.

Confidence of image

```
for i in range(0, detections.shape[2]):
```

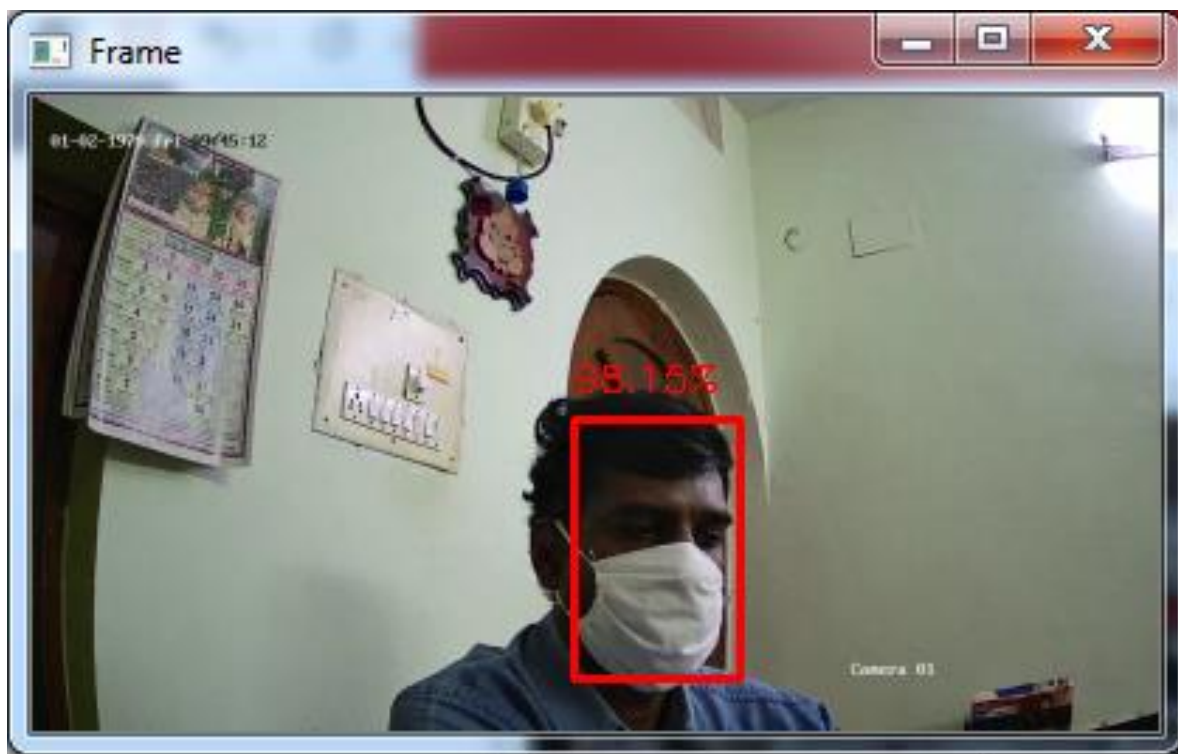
This is a for loop which means that the process below the loop repeatedly occurs from 0 to the higher limit, which in this is *detections.shape[2]*.

```
confidence = detections[0, 0, i, 2]
```

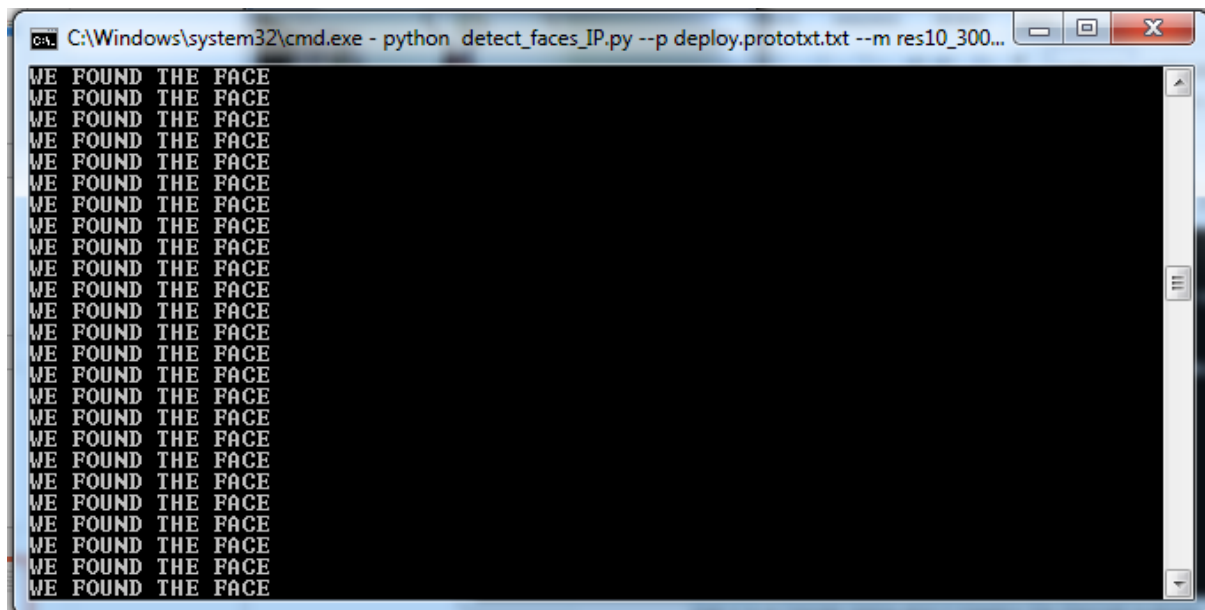
Confidence is the probability of the image in the video stream being an actual person. This is associated with the prediction performed earlier.

```
if confidence < args["confidence"]:  
    continue
```

The weak or unclear detections can be eliminated by ensuring that only frame with the confidence level greater than the minimum confidence level. Hence, it requires an if-else statement. The minimum confidence level used here is 92%. If the confidence level is higher than 92%, then the phrase “WE FOUND THE FACE” is shown in the Command Prompt.



The confidence is the percentage in red colour above the red box around the face. Here, it is 98.15%. Since 98.15% is greater than 92%, then the face has been identified. Hence, “WE FOUND THE FACE” is shown in the Command Prompt, as shown below.



```
C:\Windows\system32\cmd.exe - python detect_faces_IP.py --p deploy.prototxt.txt --m res10_300...
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
WE FOUND THE FACE
```

Bounding Box

```
box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
(startX, startY, endX, endY) = box.astype("int")

text = "{:.2f}%".format(confidence * 100)
y = startY - 10 if startY - 10 > 10 else startY + 10
cv2.rectangle(frame, (startX, startY), (endX, endY),
              (0, 0, 255), 2)
cv2.putText(frame, text, (startX, y),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
```

The bottom-left quadrant of the dashboard is a close up of the customer after the stream is frozen and their image is captured. This piece of the code computes the (x, y)-coordinates of the box. Then, the box around the face is drawn.



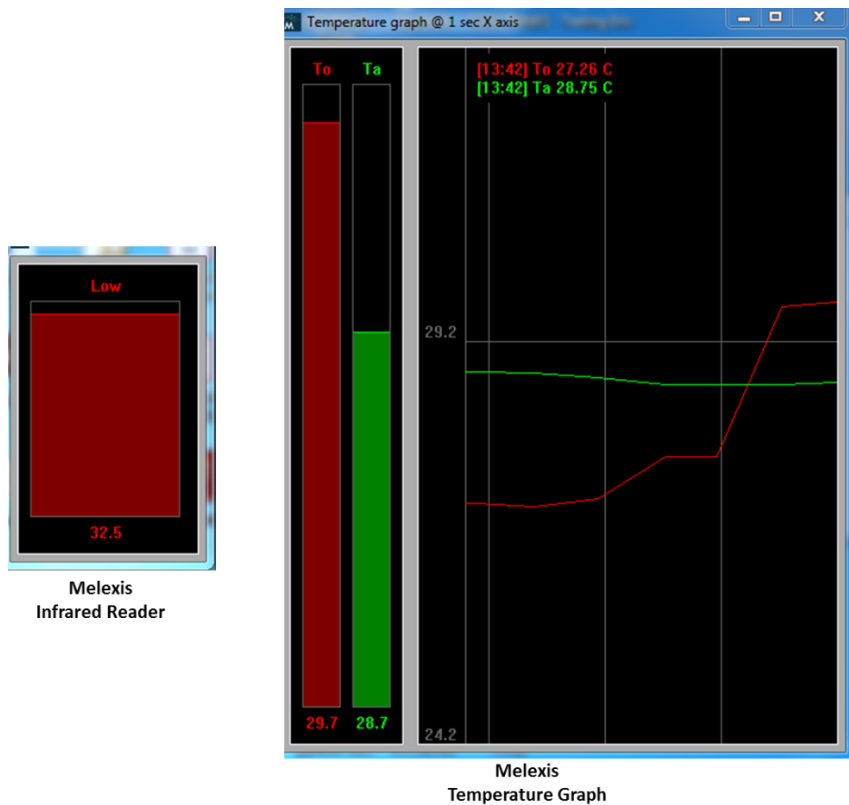
This is an example of a bounding box as it focuses on the face alone and ignores the rest of the frame.

tempcsv.py

Importing files

```
os.system("C:\Program Files (x86)\Melexis\MlxCI RT 90632\MlxCI RT.exe")
```

This line invokes the Melexis app. Then, it runs the applications causing the temperatures to be stored.



These are the temperature readings that can be shown when the Melexis are opened. This data is then stored on the automatically created CSV File on the Desktop of the laptop.

```
with open(r'C:\Users\computer\Desktop\Testing Env\mlxcirt3_log.csv')  
as csv_file:
```

As mentioned earlier, the temperatures are stored every few milliseconds in a CSV File, titled 'mlxcirt3_log.csv'. This line invokes the CSV file to access the temperature readings.

	A	B	C	D	E	F	G	H	I
190	19:13:36	90632-521	28.83	29.07	31.7	23	21	22792	23320
191	19:13:37	90632-521	28.83	29.07	31.7	23	21	22792	23320
192	19:13:37	90632-521	28.83	29.23	31.89	39	37	22792	23320
193	19:13:38	90632-521	28.83	29.09	31.73	26	21	22793	23318
194	19:13:38	90632-521	28.81	29.29	31.97	44	43	22795	23318
195	19:13:39	90632-521	28.81	29.29	31.97	44	43	22795	23318
196	19:13:39	90632-521	28.81	28.63	31.19	-16	-19	22795	23318
197	19:13:40	90632-521	28.81	29.87	32.64	99	97	22795	23318
198	19:13:41	90632-521	28.83	27.19	29.5	-151	-154	22793	23317
199	19:13:41	90632-521	28.83	27.19	29.5	-151	-154	22793	23317
200	19:13:41	90632-521	28.81	27.15	29.46	-152	-155	22793	23321
201	19:13:42	90632-521	28.75	27.25	29.59	-137	-140	22788	23326
202	19:13:43	90632-521	28.67	27.77	30.21	-82	-85	22783	23326
203	19:13:43	90632-521	28.67	27.77	30.21	-82	-85	22783	23326
204	19:13:44	90632-521	28.67	29.63	32.39	91	89	22780	23329
205	19:13:44	90632-521	28.69	29.69	32.45	94	94	22780	23328
206	19:13:45	90632-521	28.67	29.45	32.18	75	72	22783	23328
207	19:13:45	90632-521	28.65	29.21	31.9	52	52	22783	23329
208	19:13:46	90632-521	28.67	27.21	29.56	-132	-136	22781	23329
209	19:13:46	90632-521	28.65	27.19	29.54	-134	-137	22781	23331
210	19:13:47	90632-521	28.63	27.13	29.48	-135	-139	22777	23331
211	19:13:47	90632-521	28.61	27.15	29.5	-132	-135	22777	23334
212	19:13:48	90632-521	28.57	27.17	29.54	-128	-130	22773	23334
213	19:13:48	90632-521	28.55	27.17	29.54	-127	-128	22773	23338
214	19:13:49	90632-521	28.51	27.19	29.57	-121	-123	22769	23338
215	19:13:49	90632-521	28.49	27.17	29.55	-121	-123	22769	23341
216	19:13:50	90632-521	28.47	27.15	29.53	-120	-122	22765	23341
217	19:13:50	90632-521	28.45	27.17	29.56	-115	-119	22765	23344

This is an example of the CSV file that could be created from running the Melexis thermal scanner.

Conversion to array

```
csv_reader = csv.reader(csv_file, delimiter=',')
```

This converts the CSV files into an array.

Maximum temperature

```
answer = 0
for column in csv_reader :
    i = float(column[4])
    if i > answer:
        answer = i
```

In *for column in csv_reader*, the CSV file is divided into columns and each column is analysed separately.

In *i = float(column[4])*, the float integer 'i' is assigned to the values from the 5th column in the CSV File.

The purpose of the temperature detection is to find the maximum temperature measured. The 'if' statement *if i > answer: answer = i*, assigns answer to the value of i. It continues to increment the index value and compares to the next value in column[4]. Answer is assigned to the next i value if i > answer. Hence, it will only present the maximum value.

	A	B	C	D	E	F	G	H	I
190	19:13:36	90632-521	28.83	29.07	31.7	23	21	22792	23320
191	19:13:37	90632-521	28.83	29.07	31.7	23	21	22792	23320
192	19:13:37	90632-521	28.83	29.23	31.89	39	37	22792	23320
193	19:13:38	90632-521	28.83	29.09	31.73	26	21	22793	23318
194	19:13:38	90632-521	28.81	29.29	31.97	44	43	22795	23318
195	19:13:39	90632-521	28.81	29.29	31.97	44	43	22795	23318
196	19:13:39	90632-521	28.81	28.63	31.19	-16	-19	22795	23318
197	19:13:40	90632-521	28.81	29.87	32.64	99	97	22795	23318
198	19:13:41	90632-521	28.83	27.19	29.5	-151	-154	22793	23317
199	19:13:41	90632-521	28.83	27.19	29.5	-151	-154	22793	23317
200	19:13:41	90632-521	28.81	27.15	29.46	-152	-155	22793	23321
201	19:13:42	90632-521	28.75	27.25	29.59	-137	-140	22788	23326
202	19:13:43	90632-521	28.67	27.77	30.21	-82	-85	22783	23326
203	19:13:43	90632-521	28.67	27.77	30.21	-82	-85	22783	23326
204	19:13:44	90632-521	28.67	29.63	32.39	91	89	22780	23329
205	19:13:44	90632-521	28.69	29.69	32.45	94	94	22780	23328
206	19:13:45	90632-521	28.67	29.45	32.18	75	72	22783	23328
207	19:13:45	90632-521	28.65	29.21	31.9	52	52	22783	23329
208	19:13:46	90632-521	28.67	27.21	29.56	-132	-136	22781	23329
209	19:13:46	90632-521	28.65	27.19	29.54	-134	-137	22781	23331
210	19:13:47	90632-521	28.63	27.13	29.48	-135	-139	22777	23331
211	19:13:47	90632-521	28.61	27.15	29.5	-132	-135	22777	23334
212	19:13:48	90632-521	28.57	27.17	29.54	-128	-130	22773	23334
213	19:13:48	90632-521	28.55	27.17	29.54	-127	-128	22773	23338
214	19:13:49	90632-521	28.51	27.19	29.57	-121	-123	22769	23338
215	19:13:49	90632-521	28.49	27.17	29.55	-121	-123	22769	23341
216	19:13:50	90632-521	28.47	27.15	29.53	-120	-122	22765	23341
217	19:13:50	90632-521	28.45	27.17	29.56	-115	-119	22765	23344

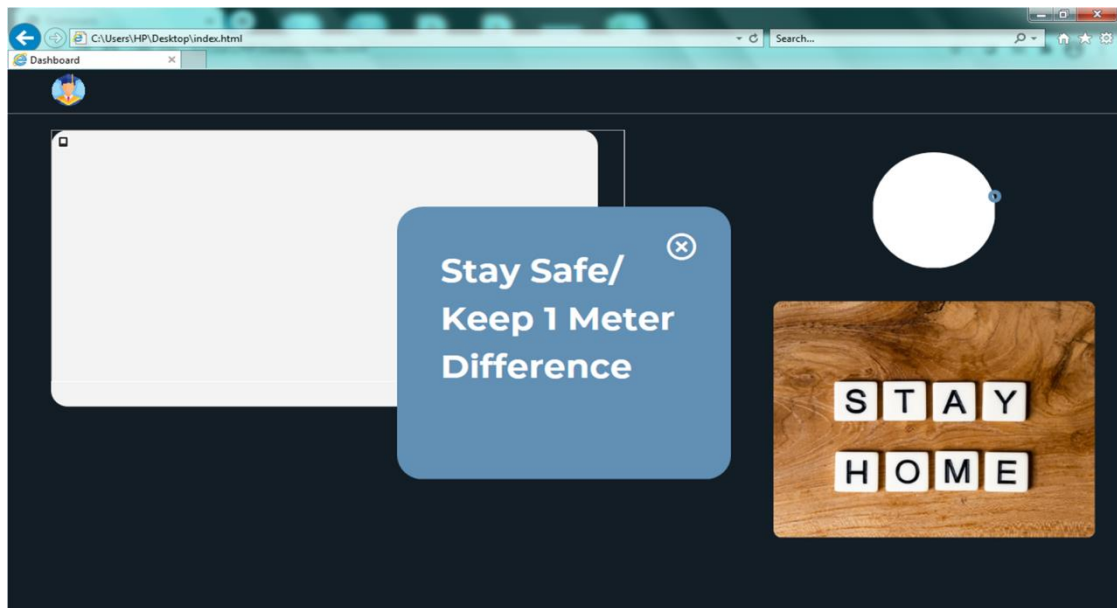
The column selected is the one rounded in black. Since the highest temperature is chosen, the temperature chosen for the HTML Dashboard is the one in red.

index(1).html

Opening of HTML file

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dashboard</title>
  <link rel="stylesheet" href="style.css">
```

This is a standard introduction to an HTML document. It includes the <!DOCTYPE html> and <head>. The title, shown as <title>Dashboard</title>, reflects the intended title of Dashboard. The stylesheet chosen was style.css, a Cascading Style Sheets file stored on the computer.



<link href="https://cdn.jsdelivr.net/npm/remixicon@2.5.0/fonts/remixicon.css" rel="stylesheet">

Remix Icon is a set of open source neutral style system symbols for users.

<https://cdn.jsdelivr.net/npm/remixicon@2.5.0/fonts/remixicon.css> is another CSS fonts sheet with specific formatting.

Start of CSS Page:

```
/*
 * Remix Icon v2.5.0
 * https://remixicon.com
 * https://github.com/Remix-Design/RemixIcon
 *
 * Copyright RemixIcon.com
 * Released under the Apache License Version 2.0
 *
 * Date: 2020-05-23
 */
@font-face {
  font-family: "remixicon";
  src: url('remixicon.eot?t=1590207869815'); /* IE9*/
  src: url('remixicon.eot?t=1590207869815#iefix') format('embedded-opentype'), /* IE6-IE8 */
  url("remixicon.woff2?t=1590207869815") format("woff2"),
  url("remixicon.woff?t=1590207869815") format("woff"),
  url('remixicon.ttf?t=1590207869815') format('truetype'), /* chrome, firefox, opera, Safari, Android, iOS 4.2+*/
  url('remixicon.svg?t=1590207869815#remixicon') format('svg'); /* iOS 4.1- */
  font-display: swap;
}

[class^="ri-"], [class*=" ri-"] {
  font-family: 'remixicon' !important;
  font-style: normal;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

.ri-lg { font-size: 1.3333em; line-height: 0.75em; vertical-align: -.0667em; }
.ri-xl { font-size: 1.5em; line-height: 0.6666em; vertical-align: -.075em; }
.ri-xxs { font-size: .5em; }
.ri-xs { font-size: .75em; }
.ri-sm { font-size: .875em; }
.ri-1x { font-size: 1em; }
.ri-2x { font-size: 2em; }
.ri-3x { font-size: 3em; }
.ri-4x { font-size: 4em; }
.ri-5x { font-size: 5em; }
.ri-6x { font-size: 6em; }
.ri-7x { font-size: 7em; }
.ri-8x { font-size: 8em; }
.ri-9x { font-size: 9em; }
.ri-10x { font-size: 10em; }
```

End of CSS Page:

```
.ri-wechat-fill:before { content: "\f2b5"; }
.ri-wechat-line:before { content: "\f2b6"; }
.ri-wechat-pay-fill:before { content: "\f2b7"; }
.ri-wechat-pay-line:before { content: "\f2b8"; }
.ri-weibo-fill:before { content: "\f2b9"; }
.ri-weibo-line:before { content: "\f2ba"; }
.ri-whatsapp-fill:before { content: "\f2bb"; }
.ri-whatsapp-line:before { content: "\f2bc"; }
.ri-wheelchair-fill:before { content: "\f2bd"; }
.ri-wheelchair-line:before { content: "\f2be"; }
.ri-wifi-fill:before { content: "\f2bf"; }
.ri-wifi-line:before { content: "\f2c0"; }
.ri-wifi-off-fill:before { content: "\f2c1"; }
.ri-wifi-off-line:before { content: "\f2c2"; }
.ri-window-2-fill:before { content: "\f2c3"; }
.ri-window-2-line:before { content: "\f2c4"; }
.ri-window-fill:before { content: "\f2c5"; }
.ri-window-line:before { content: "\f2c6"; }
.ri-windows-fill:before { content: "\f2c7"; }
.ri-windows-line:before { content: "\f2c8"; }
.ri-windy-fill:before { content: "\f2c9"; }
.ri-windy-line:before { content: "\f2ca"; }
.ri-wireless-charging-fill:before { content: "\f2cb"; }
.ri-wireless-charging-line:before { content: "\f2cc"; }
.ri-women-fill:before { content: "\f2cd"; }
.ri-women-line:before { content: "\f2ce"; }
.ri-wubi-input:before { content: "\f2cf"; }
.ri-xbox-fill:before { content: "\f2d0"; }
.ri-xbox-line:before { content: "\f2d1"; }
.ri-xing-fill:before { content: "\f2d2"; }
.ri-xing-line:before { content: "\f2d3"; }
.ri-youtube-fill:before { content: "\f2d4"; }
.ri-youtube-line:before { content: "\f2d5"; }
.ri-zcool-fill:before { content: "\f2d6"; }
.ri-zcool-line:before { content: "\f2d7"; }
.ri-zhihu-fill:before { content: "\f2d8"; }
.ri-zhihu-line:before { content: "\f2d9"; }
.ri-zoom-in-fill:before { content: "\f2da"; }
.ri-zoom-in-line:before { content: "\f2db"; }
.ri-zoom-out-fill:before { content: "\f2dc"; }
.ri-zoom-out-line:before { content: "\f2dd"; }
.ri-zzz-fill:before { content: "\f2de"; }
.ri-zzz-line:before { content: "\f2df"; }
```

When copied into a Microsoft Word document with the original formatting, the code occupies 39 pages, revealing how elaborate and detailed it is.

Static message

```

<body>
  <div id="background">
    <div id="message">
      <div id="content">
        <i class="ri-close-circle-line"></i>
        <p>stay safe/ keep 1 meter Difference</p>
      </div>
    </div>
  </div>

```

This shows the rounded box in the front of the HTML page that instructs the user to “Stay Safe” and “Keep 1 meter Difference”. Since this message and the shape of the text box are constant, they were inserted directly into the code.



Navigation Bar

```

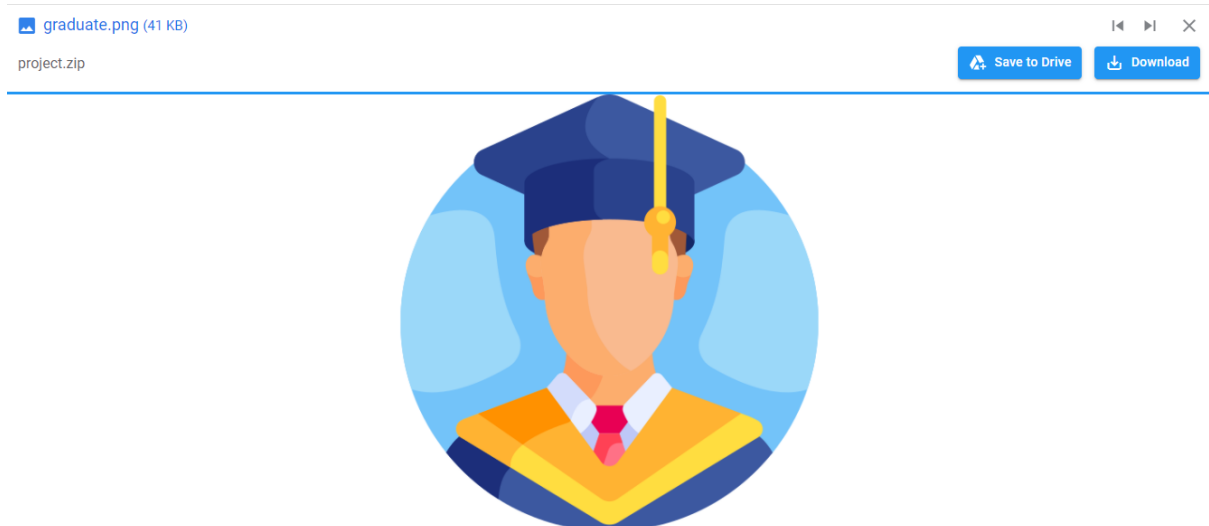
<nav>
  <div class="logo"></div>
</nav>

```

This shows the navigation bar. For the sake of convenience, the image of a graduate was used as a placeholder. This could be replaced by the logo of the owner’s shop.



The file *graduate.png* is an additional file in the same folder.



IP Camera Video Stream

```
<div id="video">  
  <iframe width="560" height="315" src="http://192.168.1.65/doc/page/preview.asp"  
  frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-  
  in-picture" allowfullscreen></iframe>
```

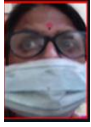
This shows the constant Video Stream feed from the IP Camera.



Bounding Box

```
<div id="F:\idbp\CS\ROI.png"></div>
```

The bounding box is, evidently, placed under the IP Camera Video Stream. It is a PNG file that keeps getting updated every time a consumer enters the shop. The bounding box from the python file is saved as ROI.png and it shown on the HTML Dashboard. When a new person walks in, the bounding box part of the python file runs again and updates the ROI.png file. As a result, the bounding box on the HTML dashboard is updated.



TOTAL:

Dashboard

File | C:/Users/user/Desktop/project/index(1).html

01-03-1970 Sat 03:49:55

Camera 01

98.5°

STAY HOME

Type here to search

13:10 23-11-2020

The image shows a web browser window displaying a dashboard. The browser's address bar shows the file path C:/Users/user/Desktop/project/index(1).html. The dashboard has a dark background. On the left, there is a large video feed of a person wearing a green patterned top, a pink shawl, and a white face mask. Above the video feed, the text '01-03-1970 Sat 03:49:55' is displayed. Below the video feed, there is a smaller, square video feed of the same person. To the right of the video feeds, there is a circular temperature display showing '98.5°' and a rectangular area with a wood-grain background containing the text 'STAY HOME' in white block letters. The Windows taskbar is visible at the bottom, showing the search bar, task view button, and various application icons. The system tray on the right shows the time as 13:10 and the date as 23-11-2020.