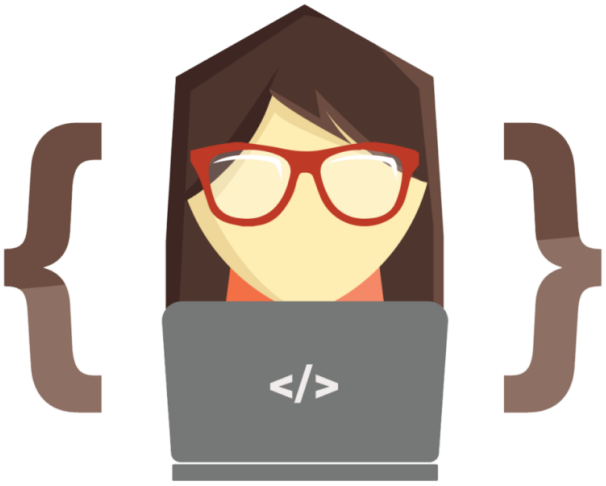


FACIAL RECOGNITION AND TRACKING

A DAY WISE PROJECT REPORT

GIRL SCRIPT DEVELOPER TECH CAMP
HACK-IN PROJECT

Hack –In is a week-long coding challenge in which the participants build a small-scale project using new technology.

BY SHUBHAM CHHIMPA
Marudhar Engineering College

ABSTRACT

The growing interest in computer vision of the past decade. Fueled by the steady doubling rate of computing power every 13 months, face detection and recognition has transcended from an esoteric to a popular area of research in computer vision and one of the better and successful applications of image analysis and algorithm based understanding. Because of the intrinsic nature of the problem, computer vision is not only a computer science area of research, but also the object of neuro-scientific and psychological studies, mainly because of the general opinion that advances in computer image processing and understanding research will provide insights into how our brain work and vice versa. Because of general curiosity and interest in the matter, the author has proposed to create an application that would allow user access to a particular machine based on an in-depth analysis of a person's facial features.

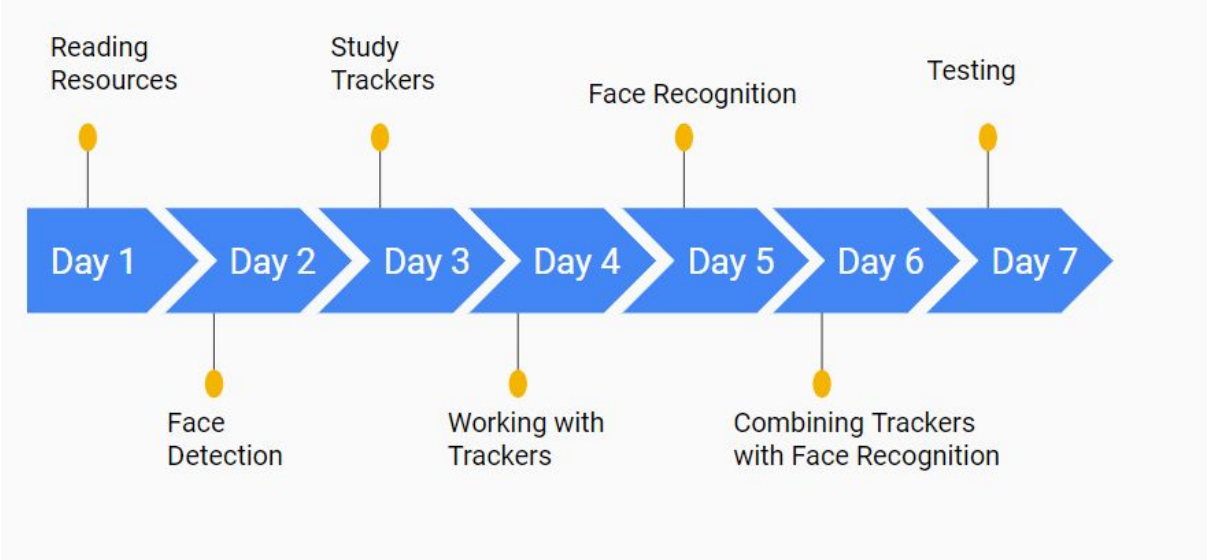
TECHNOLOGY STACK

1. OPENCV 4
2. PYTHON 3
3. PYCHARM IDE
4. WINDOW 10 OS
5. PIP
6. IMUTILS
7. NUMPY
8. PILLOW

DAY 1: CREATING TIMELINE

- Objective: To create a timeline and reading Resources for the face recognition project.

- Timeline Image:



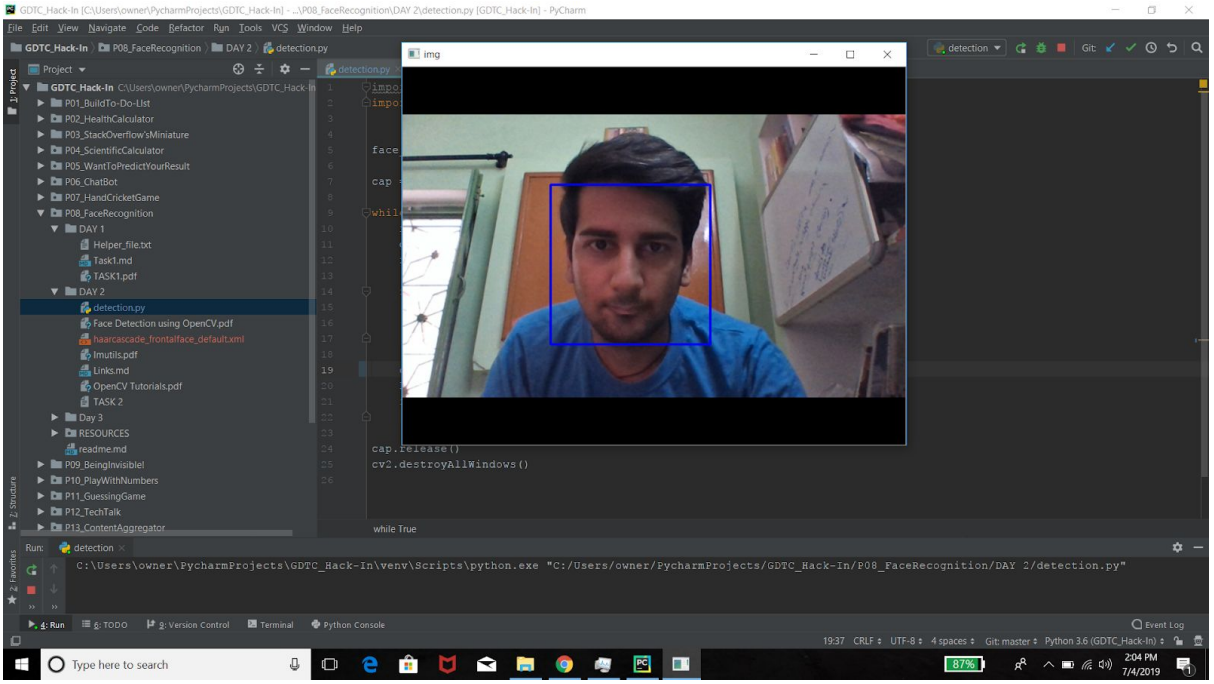
DAY 2: FACE DETECTION

- Objective: To detect fraces from webcam video feed using opencv.

- Code :

```
import numpy as np
import cv2


face_cascade                                                    =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)

while True:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
        roi_gray = gray[y:y + h, x:x + w]
        roi_color = img[y:y + h, x:x + w]

    cv2.imshow('img', cv2.flip(img,1))
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

- Sample Output Screenshots:



[fig : face detected by opencv haarcascade]

DAY 3: STUDY TRACKERS

❖ Objective: to study opencv object trackers

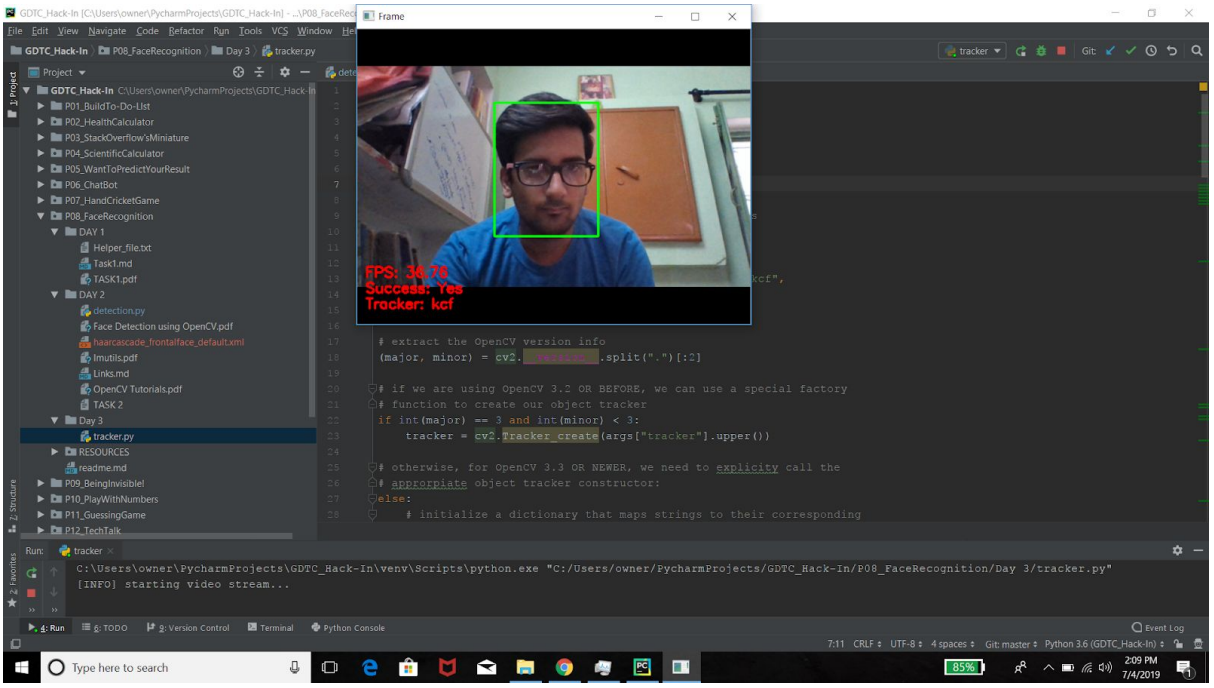
❖ Code :

```
# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import argparse
import imutils
import time
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", type=str,
                help="path to input video file")
ap.add_argument("-t", "--tracker", type=str, default="kcf",
                help="OpenCV object tracker type")
args = vars(ap.parse_args())

# extract the OpenCV version info
(major, minor) = cv2.__version__.split(".")[:2]

# if we are using OpenCV 3.2 OR BEFORE, we can use a special factory
# function to create our object tracker
if int(major) == 3 and int(minor) < 3:
   tracker = cv2.Tracker_create(args["tracker"].upper())

# otherwise, for OpenCV 3.3 OR NEWER, we need to explicity call the
# approrpiate object tracker constructor:
else:
   # initialize a dictionary that maps strings to their corresponding
   # OpenCV object tracker implementations
   OPENCV_OBJECT_TRACKERS = {
       "csrt": cv2.TrackerCSRT_create,
       "kcf": cv2.TrackerKCF_create,
       "boosting": cv2.TrackerBoosting_create,
       "mil": cv2.TrackerMIL_create,
       "tld": cv2.TrackerTLD_create,
       "medianflow": cv2.TrackerMedianFlow_create,
```

```
        "mosse": cv2.TrackerMOSSE_create
    }

    # grab the appropriate object tracker using our dictionary of
    # OpenCV object tracker objects
    tracker = OPENCV_OBJECT_TRACKERS[args["tracker"]]()

# initialize the bounding box coordinates of the object we are going
# to track
initBB = None

# if a video path was not supplied, grab the reference to the web cam
if not args.get("video", False):
    print("[INFO] starting video stream...")
    vs = VideoStream(src=0).start()
    time.sleep(1.0)

# otherwise, grab a reference to the video file
else:
    vs = cv2.VideoCapture(args["video"])

# initialize the FPS throughput estimator
fps = None

# loop over frames from the video stream
while True:
    # grab the current frame, then handle if we are using a
    # VideoStream or VideoCapture object
    frame = vs.read()
    frame = frame[1] if args.get("video", False) else frame

    # check to see if we have reached the end of the stream
    if frame is None:
        break

    # resize the frame (so we can process it faster) and grab the
    # frame dimensions
    frame = imutils.resize(frame, width=500)
    (H, W) = frame.shape[:2]

    # check to see if we are currently tracking an object
    if initBB is not None:
        # grab the new bounding box coordinates of the object
        (success, box) = tracker.update(frame)

        # check to see if the tracking was a success
        if success:
            (x, y, w, h) = [int(v) for v in box]
            cv2.rectangle(frame, (x, y), (x + w, y + h),
                          (0, 255, 0), 2)
```

```
        # update the FPS counter
        fps.update()
        fps.stop()

        # initialize the set of information we'll be displaying on
        # the frame
        info = [
            ("Tracker", args["tracker"]),
            ("Success", "Yes" if success else "No"),
            ("FPS", "{:.2f}".format(fps.fps())),
        ]

        # loop over the info tuples and draw them on our frame
        for (i, (k, v)) in enumerate(info):
            text = "{}: {}".format(k, v)
            cv2.putText(frame, text, (10, H - ((i * 20) + 20)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the 's' key is selected, we are going to "select" a bounding
    # box to track
    if key == ord("s"):
        # select the bounding box of the object we want to track (make
        # sure you press ENTER or SPACE after selecting the ROI)
        initBB = cv2.selectROI("Frame", frame, fromCenter=False,
                               showCrosshair=True)

        # start OpenCV object tracker using the supplied bounding box
        # coordinates, then start the FPS throughput estimator as well
        tracker.init(frame, initBB)
        fps = FPS().start()
    # if the `q` key was pressed, break from the loop
    elif key == ord("q"):
        break

# if we are using a webcam, release the pointer
if not args.get("video", False):
    vs.stop()

# otherwise, release the file pointer
else:
    vs.release()

# close all windows
cv2.destroyAllWindows()
```

❖ Sample Output Screenshots:



[fig : face tracked by opencv KCF tracker]

DAY 4: WORKING WITH TRACKERS

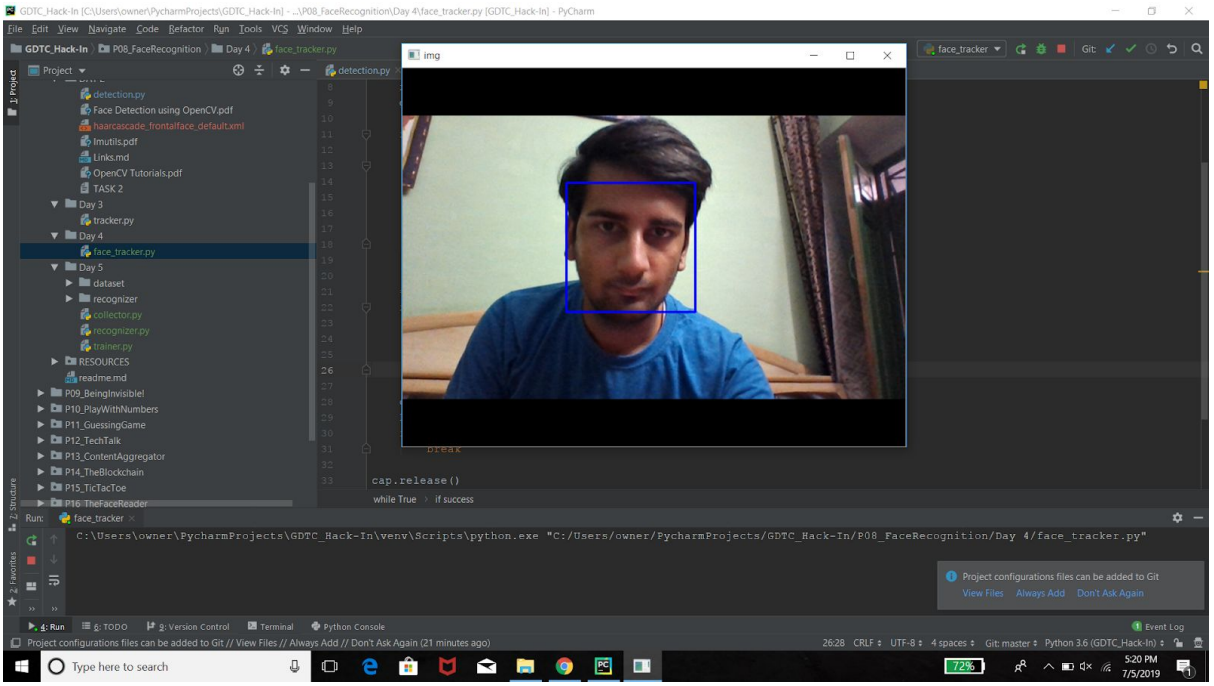
- Objective: to track face detected by opencv haarcascade classifier.

- Code :

```
    import cv2

face_cascade = cv2.CascadeClassifier('../day
2/haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)
frame_num = 1
while True:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    if frame_num == 1:
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
            roi_gray = gray[y:y + h, x:x + w]
            roi_color = img[y:y + h, x:x + w]
            tracker = cv2.TrackerCSRT_create()
            tracker.init(img, (x, y, w, h))
    (success, box) = tracker.update(img)

    # check to see if the tracking was a success
    if success:
        frame_num = frame_num + 1
        (x, y, w, h) = [int(v) for v in box]
        cv2.rectangle(img, (x, y), (x + w, y + h),
                      (255, 0, 0), 2)

    cv2.imshow('img', cv2.flip(img, 1))
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

cap.release()
```

```
cv2.destroyAllWindows()
```

❖ Sample Output Screenshots:



[fig : face detected and then tracked by opencv]

DAY 5: FACE RECOGNITION

- Objective: to recognize face using opencv LBPH face recognizer.

- Code :

1. Data collector

```
import cv2
import imutils


# Create the haar cascade
faceCascade = cv2.CascadeClassifier('../day
2/haarcascade_frontalface_default.xml')

id = input("input id of user to be detected")
sampleNum = 0
cap = cv2.VideoCapture(0)

while True:
    ret, image = cap.read()
    image = imutils.resize(image, height=300)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Detect faces in the image
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.2,
        minNeighbors=5,
        minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE
    )

    print("Found {0} faces!".format(len(faces)))

    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        sampleNum = sampleNum + 1
        cv2.imwrite("dataset/user." + str(id) + "." +
str(sampleNum) + ".jpg", gray[y:y + h, x:x + w])
```

```
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0),
    2)
            cv2.waitKey(100)

        cv2.imshow("Faces found", image)
        cv2.waitKey(1)
        if (sampleNum > 40):
            break
    cv2.destroyAllWindows()
```

2. Trainer

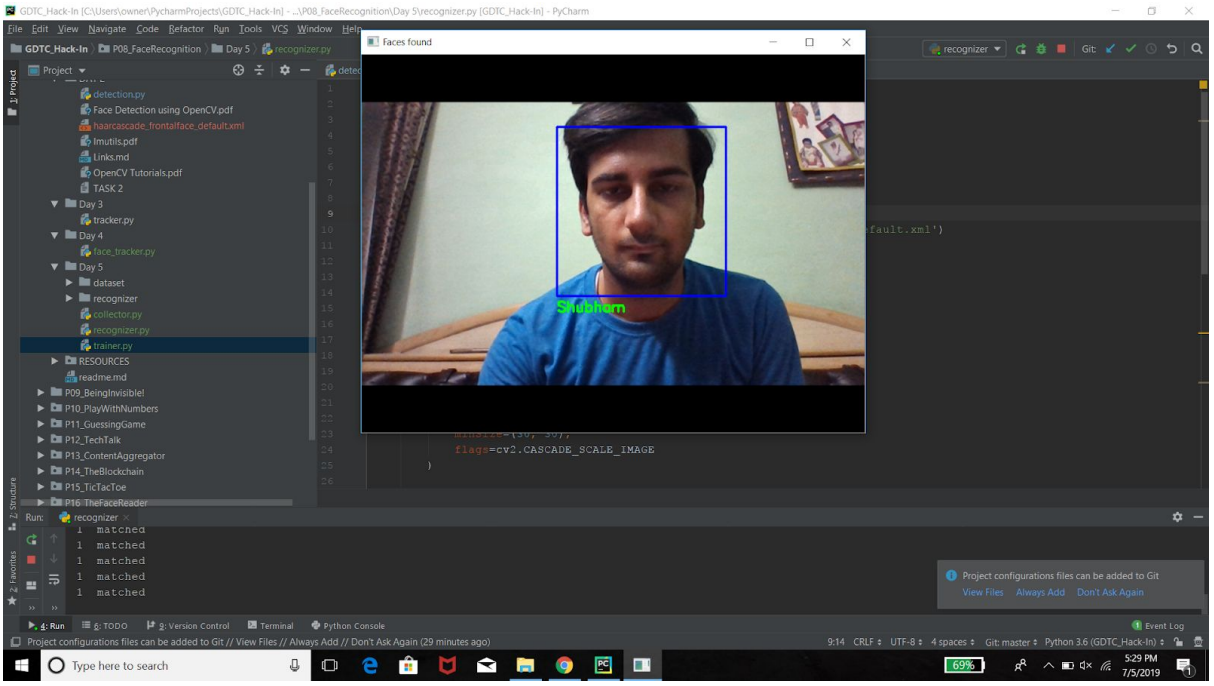
```
import os
import numpy as np
import cv2
from PIL import Image

recognizer = cv2.face.LBPHFaceRecognizer_create()
path = 'dataset'


def getImageWithID(path):
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faces = []
    IDs = []
    for imagePath in imagePaths:
        faceImg = Image.open(imagePath).convert('L')
        faceNp = np.array(faceImg, 'uint8')
        ID = int(os.path.split(imagePath)[-1].split('.')[1])
        faces.append(faceNp)
        IDs.append(ID)
        cv2.imshow("traning", faceNp)
        cv2.waitKey(10)
    return IDs, faces


Ids, faces = getImageWithID(path)
recognizer.train(faces, np.array(Ids))
recognizer.save('recognizer/trainningData.yml')
cv2.destroyAllWindows()
```

3. Recognizer

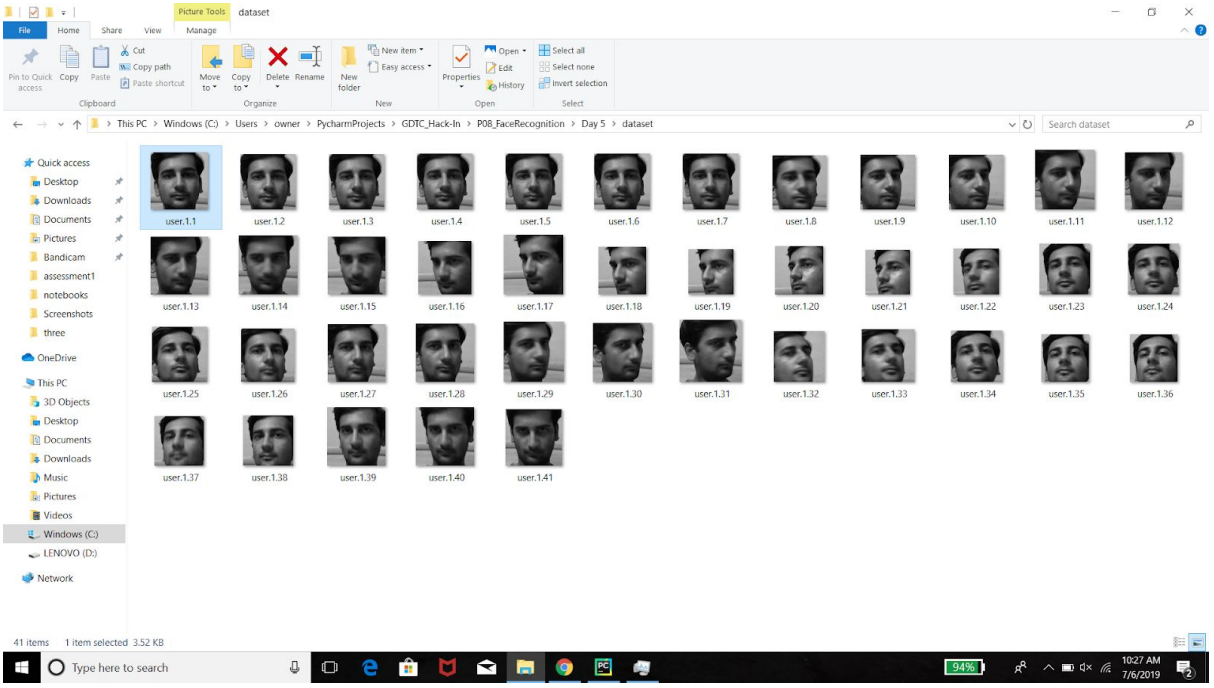
```
import cv2

recognizer = cv2.face.LBPHFaceRecognizer_create()
```

```
recognizer.read('recognizer\\trainningData.yml')
id = 0
font = cv2.FONT_HERSHEY_SIMPLEX
names = ["", "Shubham", "Vishal", "Vinit"]
cap = cv2.VideoCapture(0)
frame_num = 1
faceCascade = cv2.CascadeClassifier('../day
2/haarcascade_frontalface_default.xml')

while True:
    ret, image = cap.read()
    # Create the haar cascade
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    if frame_num == 1:

        # Detect faces in the image
        faces = faceCascade.detectMultiScale(
            gray,
            scaleFactor=1.2,
            minNeighbors=5,
            minSize=(30, 30),
            flags=cv2.CASCADE_SCALE_IMAGE
        )

        print("Found {0} faces!".format(len(faces)))

        # Draw a rectangle around the faces
        for (x, y, w, h) in faces:
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
            tracker = cv2.TrackerCSRT_create()
            tracker.init(image, (x, y, w, h))

    (success, box) = tracker.update(image)
    # check to see if the tracking was a success
    if success:
        frame_num = frame_num + 1
        (x, y, w, h) = [int(v) for v in box]
        cv2.rectangle(image, (x, y), (x + w, y + h),
                      (255, 0, 0), 2)
        id, conf = recognizer.predict(gray[y:y + h, x:x + w])
        print(str(id) + "  matched")

        cv2.putText(image, names[int(id)], (x, y + h + 20), font, .6,
(0, 255, 0), 2)

    cv2.imshow("Faces found", image)
    if cv2.waitKey(1) == 27:
        break
cv2.destroyAllWindows()
```

❖ Sample Output Screenshots:



[fig : face recognised by opencv LBPH face recogniser]

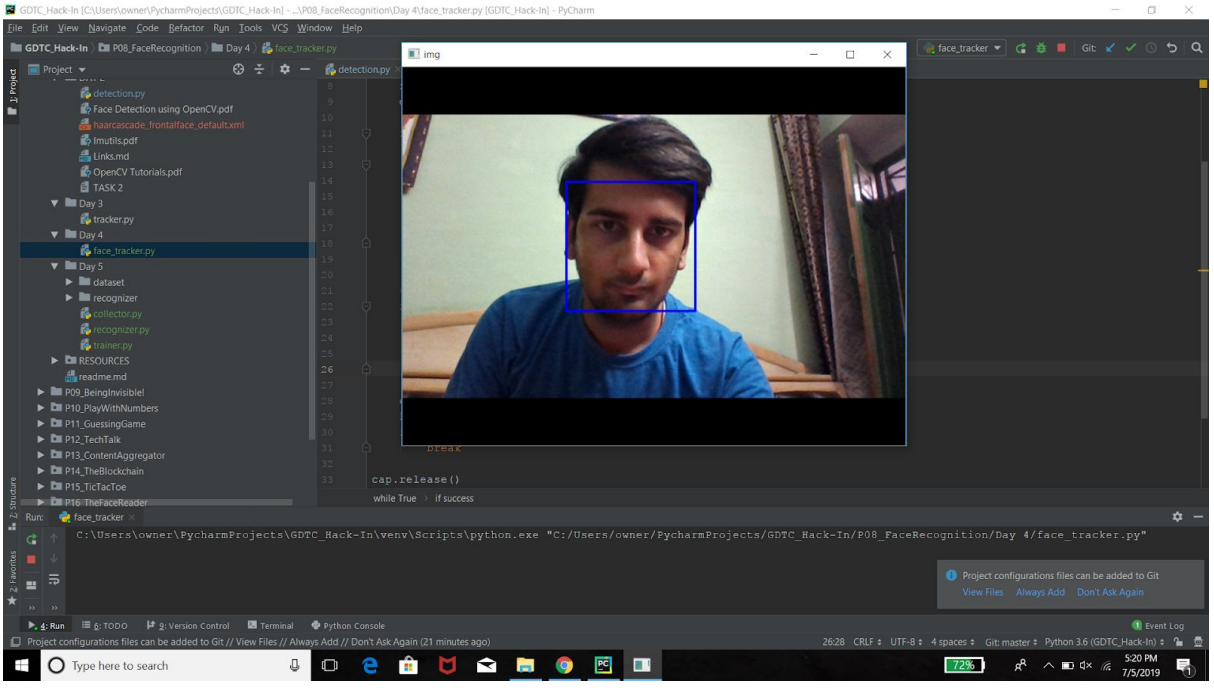


[fig : face dataset for training LBPH face recogniser]

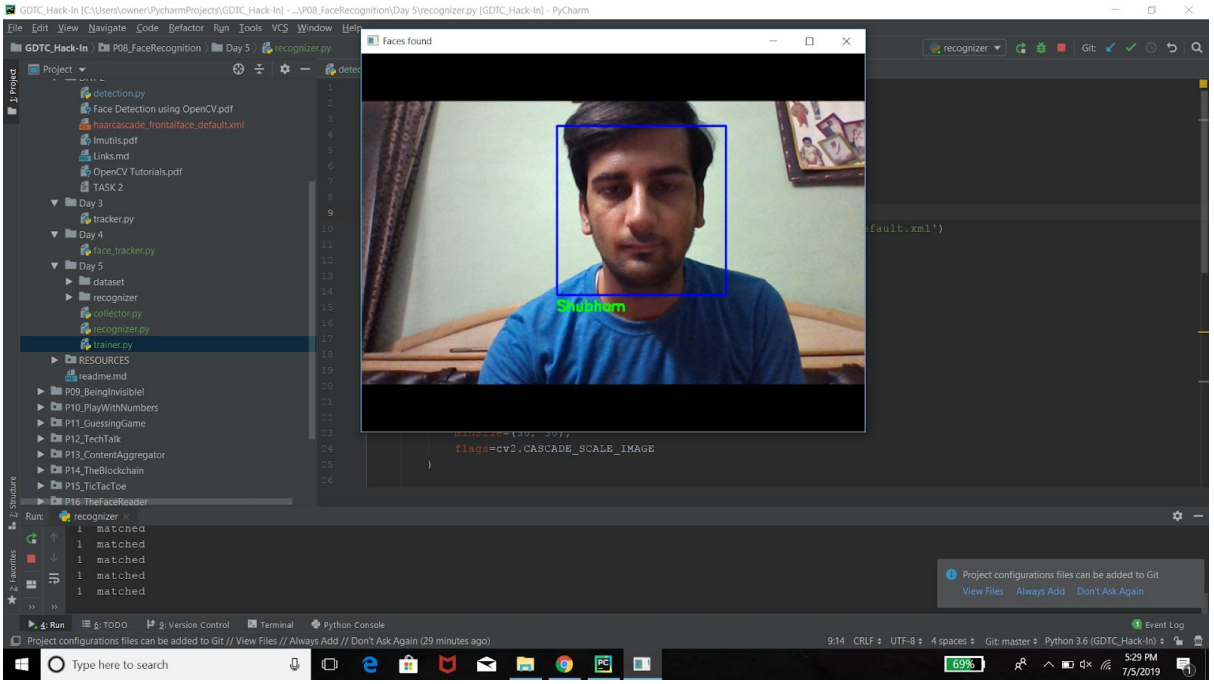
DAY 6: TESTING

- Objective: to test the face recognition and its working.

- Sample Output Screenshots:



[fig : face detected and tracked by recognise.py]



[fig : face recognised by recogniser.py]

FUTURE SCOPE

1. Can be used with Raspberry pi with web cam to detect robbery.
2. Can be used with Quadrone to track real time person.
3. Can be used to detect over speeding vehicles in smart traffic management system

REFERENCES

1) OpenCV

https://www.pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/

https://docs.opencv.org/2.4/doc/tutorials/introduction/table_of_content_introduction/table_of_content_introduction.html#table-of-content-introduction

2) Face Detection using Haar Cascades

https://docs.opencv.org/2.4/doc/tutorials/introduction/table_of_content_introduction/table_of_content_introduction.html#table-of-content-introduction

https://docs.opencv.org/trunk/db/d28/tutorial_cascade_classifier.html

3) Open CV Trackers

https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/

4) face Recognition

http://hanzratech.in/2015/02/03/face-recognition-using-opencv.html