**A Report**

**on**

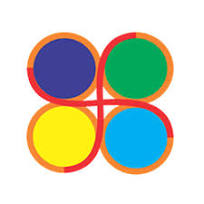
**Micro Project**

**(NAME OF MICRO-PROJECT)**

**Subject: Scripting Language Python**

**Subject Code: 4330701**

**Branch: Computer Engineering**

****

**Government Polytechnic, Ahmedabad**

**Prepared by:**

**Name: Kukshi Shah**

**Enrollment No.: 226170307190**

**Mobile no.: 8160401245**

**Semester: 3rd**

**Batch: c1**

**Detailed Description: Face Recognition Attendance System**

**Objective:**

* **The project aims to implement a face recognition attendance system using Python and popular libraries such as OpenCV, NumPy, and face\_recognition. The system captures real-time video from a webcam, detects faces, recognizes them based on pre-encoded face data, and logs attendance with timestamps.**
* **This project demonstrates practical application of computer vision and face recognition techniques.**
* **It involves image processing, real-time video analysis, and integration of external libraries.**
* **It serves as a foundation for learning about attendance management systems and biometric identification.**

**Libraries Used:**

* **OpenCV (cv2)**
* **NumPy (np)**
* **face\_recognition**
* **os**
* **datetime**

**Software Versions:**

* **Python: 3.x**
* **OpenCV: Version 4.x**
* **NumPy: Latest version available during the project development**
* **face\_recognition: Latest version available during the project development**

**Functionality:**

**Image Loading and Encoding:**

* **The system reads images from the 'images' directory, where each image represents a known individual.**
* **Images are loaded using OpenCV, and corresponding names are extracted using the os module.**
* **findEncodings function encodes face features from the images using the face\_recognition library.**

**Real-time Face Recognition:**

* **The system uses the webcam (cv2.VideoCapture) to capture real-time video frames.**
* **Face locations and encodings are detected using the face\_recognition library.**
* **The detected faces are compared with the pre-encoded face data to recognize individuals.**

**Attendance Logging:**

* **When a recognized face is detected, the system logs the attendance by writing the person's name and timestamp to a CSV file ('attendance.csv').**
* **The markAttendance function checks if the person's name is already in the attendance log to avoid duplicate entries.**

**User Interface:**

* **The system displays the real-time video stream with bounding boxes around recognized faces.**
* **It also overlays the name of the recognized person and logs their attendance.**

**Termination:**

* **The system can be terminated by pressing the 'q' key, which releases the webcam and closes the OpenCV windows.**

**Code:**

import cv2

import numpy as np

import face\_recognition

import os

from datetime import datetime

path = 'images'

images = []

classNames = []

myList = os.listdir(path)

# print(myList)

for cls in myList:

    curImg = cv2.imread(f'{path}/{cls}')

    images.append(curImg)

    classNames.append(os.path.splitext(cls)[0])

# print(classNames)

def findEncodings(images):

    encodeList = []

    for img in images:

        img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

        face\_encodings = face\_recognition.face\_encodings(img)

        if face\_encodings:

            encode = face\_encodings[0]

            encodeList.append(encode)

    return encodeList

def markAttendance(name):

    with open('attendance.csv', 'r+') as f:

        mydataList = f.readlines()

        nameList = []

        for line in mydataList:

            entry = line.split(',')

            nameList.append(

                entry[0].strip()

            )  # strip() removes leading/trailing whitespaces

        if name not in nameList:

            now = datetime.now()

            dtstring = now.strftime('%H:%M:%S')

            f.writelines(f'\n{name} , {dtstring}')

encodeListKnown = findEncodings(images)

print('Encoding Complete')

cap = cv2.VideoCapture(0)

while True:

    success, img = cap.read()

    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)

    imgS = cv2.cvtColor(imgS, cv2.COLOR\_BGR2RGB)

    facesCurframe = face\_recognition.face\_locations(imgS)

    encodesCurrFrame = face\_recognition.face\_encodings(imgS, facesCurframe)

    for encodeFace, faceLoc in zip(encodesCurrFrame, facesCurframe):

        matches = face\_recognition.compare\_faces(encodeListKnown, encodeFace)

        faceDis = face\_recognition.face\_distance(encodeListKnown, encodeFace)

        # print(faceDis)

        matchIndex = np.argmin(faceDis)

        if matches[matchIndex]:

            name = classNames[matchIndex].upper()

            # print(name)

            y1, x2, y2, x1 = faceLoc

            y1, x2, y2, x1 = y1 \* 4, x2 \* 4, y2 \* 4, x1 \* 4

            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)

            cv2.putText(

                img,

                name,

                (x1 + 6, y2 - 6),

                cv2.FONT\_HERSHEY\_COMPLEX,

                1,

                (255, 255, 255),

                2,

            )

            markAttendance(name)

    cv2.imshow('Webcam', img)

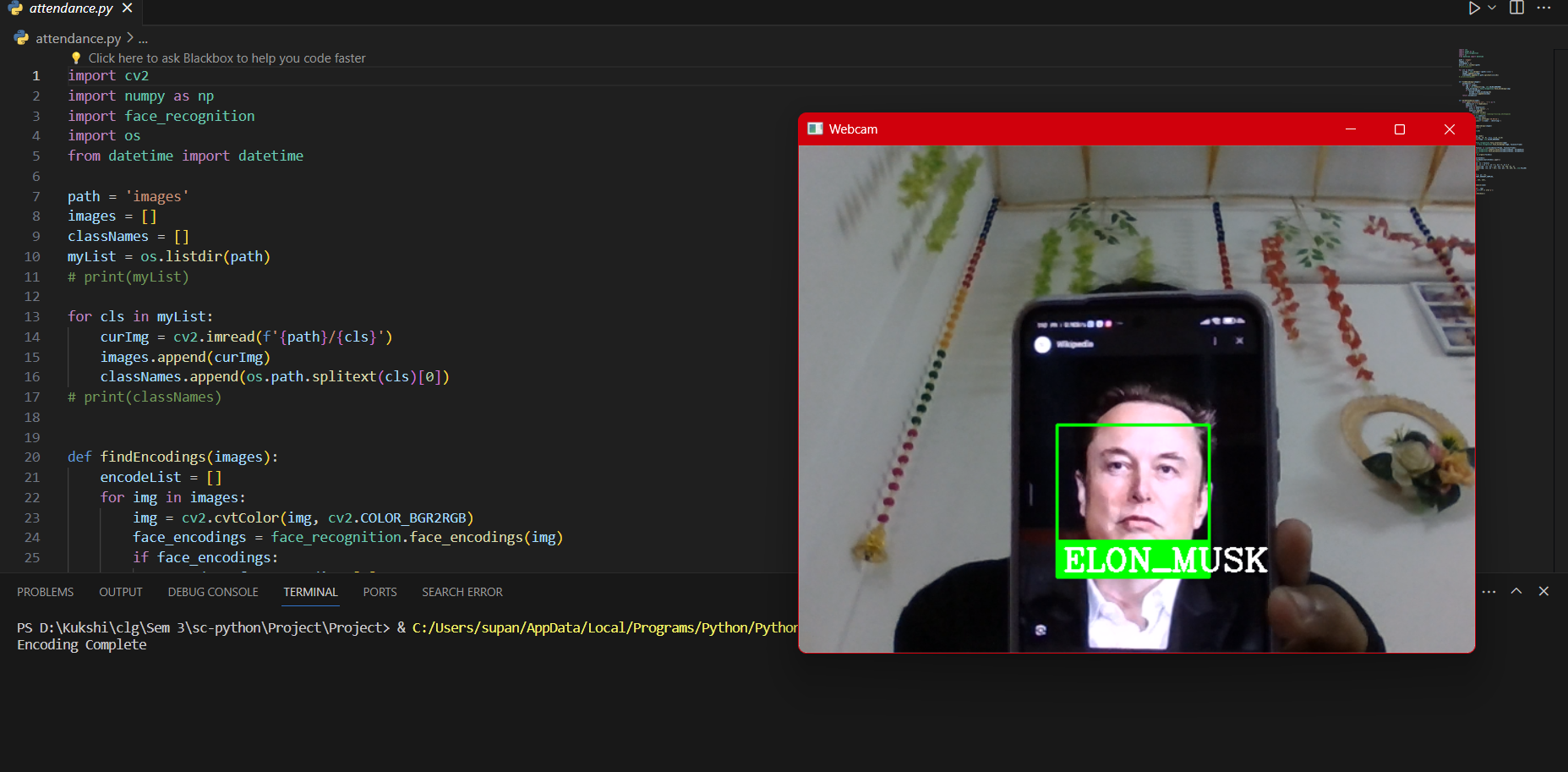
    if cv2.waitKey(1) & 0xFF == ord('q'):

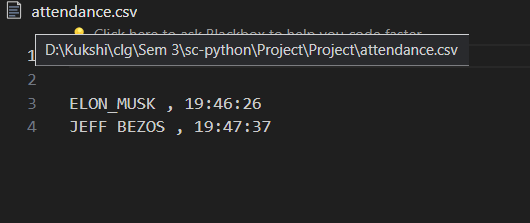
        cap.release()

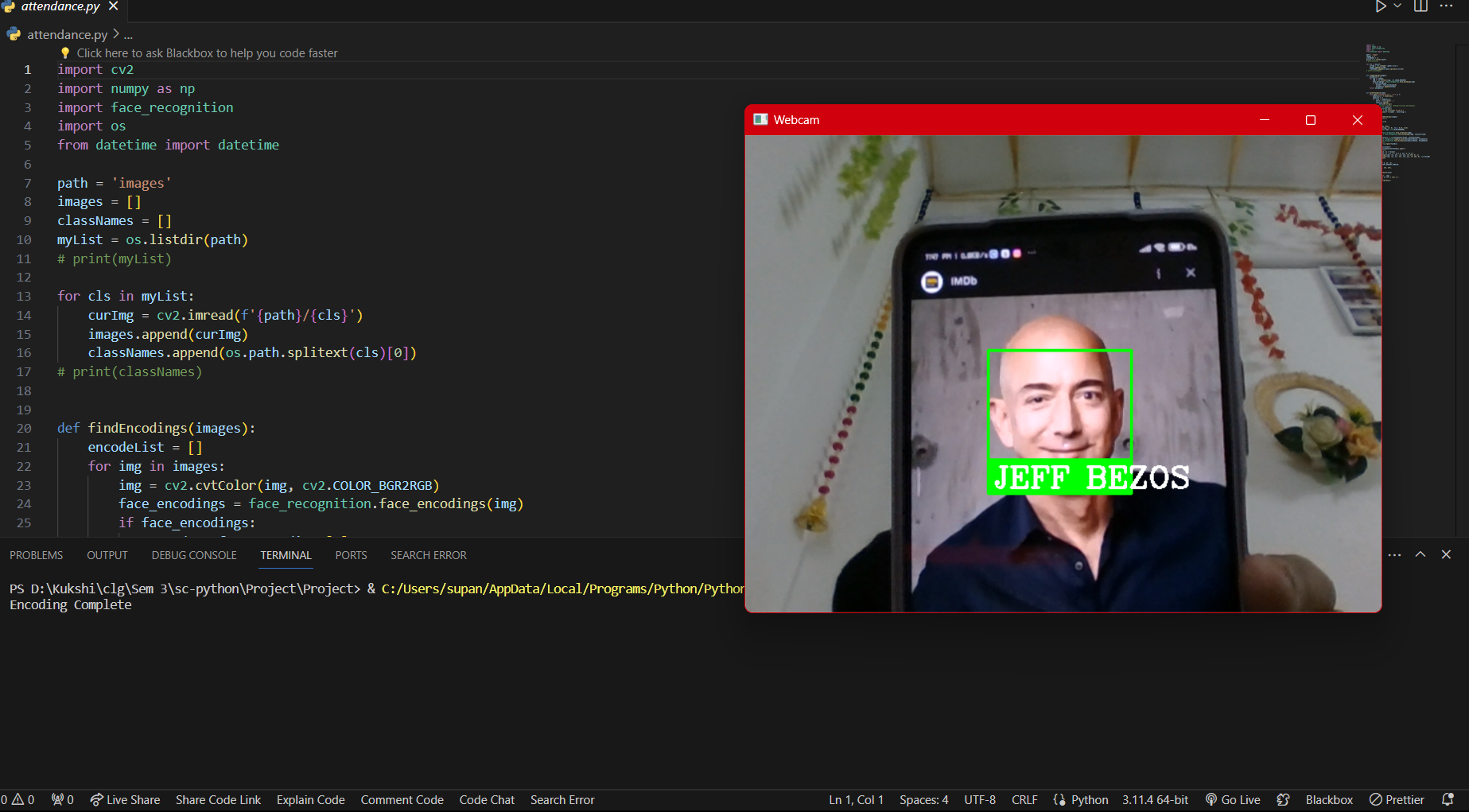
        cv2.destroyAllWindows()

        break

**Output:**

****

****

****