VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Certificate of Merit

This is to certify that the following team of students of B.Tech (CSE) studying in the School of Computer Science and Engineering, VIT, Chennai Campus have successfully completed the project work entitled

Facial Recognition Attendance Management System

…………………………………………………………………..

between 25th April 2023 and 5th July 2023.

The work was found commendable and good.

Team Members:

1. 21BCE1163 Sandip Datta
2. 21BCE5806 S.V. Rohith Kumar

Duly Examined and Certified by

INDEX

.

Project Name: Facial Recognition Attendance Management System

Introduction:

Face recognition systems have revolutionized the way we interact with technology, enabling computers to identify and verify individuals based on their unique facial features. A face recognition system is a sophisticated technology that analyzes facial patterns and compares them against a database of known faces to determine a person's identity. This technology has found widespread applications in various fields, including security, law enforcement, access control, and personal device authentication.

The primary objective of a face recognition system is to accurately match a captured face image with a corresponding identity in a database. This process involves multiple stages, including face detection, feature extraction, and face matching. First, the system locates and isolates the face within an image or video frame using advanced algorithms and techniques. Next, it extracts distinctive features from the face, such as the distance between the eyes, the shape of the nose, and the contours of the face.

These extracted features are then compared with the features stored in a database, which may contain a large number of known individuals' faces. The system applies sophisticated matching algorithms to compute the similarity or dissimilarity between the captured face and the database entries. Based on this analysis, a face recognition system can determine if a match exists and provide an identification result.

Purpose:

The purpose of this project is to take attendance of the student using face recognition system so that manual error can be reduced and accuracy can be increased and also it will consume less time because system is very fast and efficient. To increase more accuracy and make it antispoofing we will integrate liveness detection.

Scope of the project:

1.**Automated Attendance Tracking Process:** The primary objective of a face recognition attendance system in school or colleges is to automate the process of tracking student attendance. By leveraging facial recognition technology, the system can accurately and efficiently record the presence of students in classrooms or other designated areas.

2.**Time Efficiency:** Traditional manual attendance methods, such as taking roll calls or using paper-based systems, can be time-consuming and prone to errors. A face recognition attendance system eliminates the need for manual entry and saves valuable instructional time by quickly and accurately identifying students as they enter or exit a classroom.

3.**Increased Accuracy:** Face recognition technology provides a high level of accuracy in identifying individuals. It reduces the chances of errors or discrepancies that may occur with manual attendance tracking methods, such as mistaken identity, forged signatures, or proxy attendance.

4.**Real-Time Monitoring:** The system allows for real-time monitoring of attendance data, providing immediate feedback on the presence or absence of students. This information can be valuable for instructors, administrators, and parents who need to keep track of students' attendance patterns and identify potential issues or irregularities.

5.**Improved Accountability:** A face recognition attendance system promotes accountability among students by ensuring that they are physically present during classes. It discourages unauthorized absences, encourages regular attendance, and holds students responsible for their attendance records. Overall, implementing a face recognition attendance system in colleges aims to enhance efficiency, accuracy, accountability, and security in attendance tracking while providing valuable data for analysis and decision-making.

The Main Drawbacks of Face recognition attendance system:

Accuracy and Reliability:

Although face recognition technology has made significant advancements, it is not flawless. Accuracy can be affected by various factors such as lighting conditions, pose variations, occlusions, and image quality. The system may encounter challenges in accurately recognizing individuals, resulting in false positives or false negatives, leading to inaccurate attendance records.

Differentiating between 3d and 2d objects:

In face recognition if you show the camera some photo of some person then it accepts and gives authorization so this is very critical situation so any can bypass the security and can get the attendances for their friends who are not even physically present.

How we are going to approach the problems:

For more accuracy and reliability, we are going to train our model with large number of datasets so that it gives us correct results all the time.

For differentiating between 3d and 2d objects we are going to use one component which is called liveness detection. Liveness detection is an important component of face recognition systems that aims to verify whether a detected face is from a live human or a spoofing attempt using a fake representation of a face, such as a photograph, mask, or video. Liveness detection technology helps enhance the security and reliability of face recognition systems by mitigating the risk of spoofing attacks.

The primary objective of liveness detection is to differentiate between a genuine, live human face and a non-living, spoofed representation. By analyzing various physiological and behavioral cues, liveness detection algorithms aim to identify signs of vitality and ensure that the face being presented is actively generated by a real person.

There are several techniques and approaches used for liveness detection, including:

1.Motion Analysis: This technique examines the presence of natural, spontaneous movements that are typically associated with a living person. It analyzes subtle facial movements, such as eye blinks, head tilts, or facial expressions, to determine the authenticity of the face.

2.Texture Analysis: By analyzing the fine-grained texture details on a face, liveness detection algorithms can identify inconsistencies or anomalies that are indicative of a spoofing attempt. This may involve analyzing features like skin texture, pores, micro-movements, or reflections on the face.

3.Depth Information: Utilizing depth-sensing technologies, such as infrared sensors or structured light, liveness detection algorithms can capture depth information and analyze the geometric properties of the face. This helps distinguish between a flat 2D representation and a three-dimensional, live human face.

4.Challenge-Response Tests: Liveness detection can involve presenting challenges to the user that require a genuine, live response. For example, the system may ask the user to follow specific instructions, perform certain actions, or respond to random prompts. The responses are then analyzed to verify liveness.

System Requirements:

Product perspective:

This face recognition attendance management system will take care of the attendance of any class or meeting automatically and at last it will generate a .csv file where the names of the attendees will be recorded.

Software Requirement:

1. Python programming language;
2. OpenCV;
3. Operating System: Microsoft Windows 10 or Windows 11;

Hardware Requirements:

1. 4 GB RAM (Minimum)
2. 80 GB HDD
3. Processor: i3,i5 and greater versions;
4. Input devices: Key board and mouse;

Functional Requirement:

1. **Face Detection:**

The system should accurately detect and locate human faces within the video stream.

It should utilize the Haar cascade classifier for face detection.

2. **Face Recognition:**

The system should recognize individuals by comparing the detected face with pre-registered facial embeddings.

It should utilize the K-Nearest Neighbors (KNN) algorithm for face recognition.

The system should load pre-trained face data from the "names.pkl" and "faces_data.pkl" files.

3. **Enrollment:**

The system should allow users to enroll by providing their name and capturing facial images.

It should store the captured facial images and associated user information in the "names.pkl" and "faces_data.pkl" files, respectively.

4. **Attendance Capture:**

The system should continuously capture the video stream from the camera.

It should detect faces in the captured frames using face detection algorithms.

The system should perform blink detection to verify liveness and prevent spoofing.

5. **Blink Detection:**

The system should detect blinks by analyzing eye movements or changes in eye state.

It should calculate the eye aspect ratio (EAR) using the landmarks from the shape predictor.

If a blink is detected, the system should initiate the attendance marking process.

6. **Attendance Tracking:**

The system should record the attendance of recognized individuals.

It should store the attendance records in CSV format, with the "Attendance_" prefix and the date as the filename.

7. **User Interaction:**

The system should display the recognized name on the video frame.

It should provide feedback when a blink is detected or attendance is taken.

8. **File Management:**

The system should check if the "names.pkl" and "faces_data.pkl" files exist.

If the files exist, the system should load the data and update them with newly enrolled faces.

If the files do not exist, the system should create new files and store the enrolled faces and names.

9. **Attendance Logging:**

The system should create and update a CSV file for each date to maintain attendance records.

The CSV file should include columns for name and timestamp.

10. **Termination:**

The system should be terminated by pressing the 'q' key.

Nonfunctional requirements:

1. **Accuracy:**

The system should have a high level of accuracy in face detection and recognition to minimize false positives and negatives.

The blink detection mechanism should accurately distinguish between blinks and other eye movements.

2. **Speed and Real-time Performance:**

The system should process video frames in real-time to provide a seamless user experience.

Face detection, recognition, and blink detection should be performed efficiently to ensure timely attendance capture.

3. **Scalability:**

The system should be capable of handling a large number of enrolled individuals and attendance records without significant performance degradation.

It should support scalability as the number of enrolled individuals increases over time.

4. **Usability:**

The user interface should be intuitive and easy to use for both enrollment and attendance capture.

Clear instructions and feedback should be provided to guide users through the process.

5. **Security:**

The system should implement appropriate security measures to protect the stored facial data and attendance records.

Access to sensitive data should be restricted to authorized personnel only.

6. **Privacy:**

The system should adhere to privacy regulations and respect the privacy of individuals using the system.

Facial data and attendance records should be handled with strict confidentiality.

7. **Robustness:**

The system should be robust and able to handle variations in lighting conditions, facial poses, and expressions.

It should provide accurate results even in challenging scenarios.

8. **Reliability:**

The system should be reliable and stable, operating without unexpected crashes or failures.

It should handle exceptions and errors gracefully, providing appropriate error messages or fallback mechanisms.

9. **Maintainability:**

The codebase should be well-structured, modular, and easily maintainable to support future updates or enhancements.

Proper documentation and code commenting should be provided for better understanding and maintainability.

10. Performance:

The system should have reasonable hardware and resource requirements to ensure smooth operation on target platforms.

The system should efficiently utilize system resources, such as CPU and memory.

DESIGN SPECIFICATION:

In this project there are two main modules:

i. **capture_facedata.py**

the capture_facedata module captures faces from the webcam video feed, detects and crops them using a Haar cascade classifier, and stores the captured faces for enrollment in a facial recognition attendance management system. It prompts the user to enter their name, associates it with the captured faces, and saves the data in separate files. The captured faces are resized to a fixed dimension and stored in 'faces_data.pkl', while the user's name is stored in 'names.pkl'. This module serves as the initial step in the system, allowing users to enroll by capturing their facial data, which will later be used for face recognition and attendance tracking.

Let's analyze the module in depth ,

```
import cv2
import pickle
import numpy as np
import os
```

so for running this code first we need to get all our libraries which are needed to build the code,'

OpenCV (cv2):

OpenCV is a popular open-source computer vision library.

It provides various functions and algorithms for image and video processing, including face detection, image manipulation, and video capture.

In the code, it is used for video capture (VideoCapture), grayscale conversion (cvtColor), face detection (CascadeClassifier), drawing overlays on frames (putText, rectangle), and displaying frames (imshow).

Additionally, OpenCV is used to save and load face data and names using pickle and file operations (os).

Pickle:

Pickle is a module in Python's standard library used for object serialization and deserialization.

It allows objects to be converted into a byte stream and stored in a file or transferred over a network.

In the code, it is used to save and load the captured face data (faces_data) and associated names (names) into separate pickle files ('faces_data.pkl' and 'names.pkl').

NumPy (np):

NumPy is a fundamental library for scientific computing in Python.

It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

In the code, NumPy is used to convert the faces_data list into a NumPy array (np.asarray) and reshape it to a specific dimension (reshape).

NumPy is also used to append the captured face data to the existing face data and perform array operations (np.append).

OS:

The os module provides a way to interact with the operating system in a portable manner.

It provides functions for file and directory operations, such as checking file existence, listing directory contents, and creating directories.

In the code, the os module is used to check the existence of specific files ('names.pkl', 'faces_data.pkl') in the 'data/' directory (os.listdir('data/')).

The commands to install the libraries:

OpenCV installation:

https://www.geeksforgeeks.org/how-to-install-opencv-for-python-in-windows/

pickle installation:

https://stackoverflow.com/questions/68582382/how-to-pip-install-pickle-under-python-3-9-in-windows

NumPy installation:

https://numpy.org/install/

OS installation:

os is a standard python module its already there.

```
video=cv2.VideoCapture(0)
facedetect=cv2.CascadeClassifier('data/haarcascade_frontalface_default.xml')

faces_data=[]

i=0

name=input("Enter Your Name: ")

while True:
    ret,frame=video.read()
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces=facedetect.detectMultiScale(gray, 1.3 ,5)
    for (x,y,w,h) in faces:
        crop_img=frame[y:y+h, x:x+w, :]
        resized_img=cv2.resize(crop_img, (50,50))
        if len(faces_data)<=100 and i%10==0:
            faces_data.append(resized_img)
        i=i+1
        cv2.putText(frame, str(len(faces_data)), (50,50), cv2.FONT_HERSHEY_COMPLEX, 1,
(50,50,255), 1)
        cv2.rectangle(frame, (x,y), (x+w, y+h), (50,50,255), 1)
    cv2.imshow("Frame",frame)
    k=cv2.waitKey(1)
    if k==ord('q') or len(faces_data)==100:
        break
video.release()
cv2.destroyAllWindows()
```

This part of the code is responsible to capture faces from a video feed through the webcam and it'll store those for the enrollment in the facial recognition attendance management system.

Video capture:

video=cv2.VideoCapture(0) this starts to video capture object to access the default webcam which is the pc's webcam that's why in brackets the index is 0. It allows capturing frames from the webcam.

Face Detection and capture loop:

The code enters a continuous loop(while true) because it will capture frames from the video feed.

Each of the t=frames are captured using video.read() and assigned to frame and then the frame is converted to grayscale using cv2.cvtcolor(frame,cv2.color_bgr2gray).face detection is performed on the grayscale image using facedetect.detectMultiScale(gray,1.3,5).

facedetect.detectMultiScale(gray, 1.3 ,5) detects the faces in the grayscale image using the detectMultiScale function of the CascadeClassifier object, and returns the coordinates of the detected faces.

for (x,y,w,h) in faces: loops through the detected faces and extracts the region of interest (ROI) containing the face from the original color image.

cv2.rectangle(frame, (x,y), (x+w, y+h), (50,50,255), 1) draws a rectangle around the detected face in the original color image.

Frame Display and Exit Condition:

cv2.imshow("Frame",frame) displays the original color image with the detected faces.

cv2.waitKey(1) waits for a key press for 1 millisecond.

if k==ord('q') or len(faces_data)==100: breaks the while loop either when the user presses the 'q' key or when the program has captured 100 face images.

faces_data.append(resized_img) adds the resized face image (50x50 pixels) to a list of face images, which can be used for face recognition.

Video Capture Cleanup:

Video.release() it will release the video capture object and will free the webcam.

Cv2.destroyALLWindows() it will close all the OpenCV windows.

ii. rld.py module:

This module is designed to create a facial recognition attendance management system. It utilizes computer vision and machine learning techniques to capture face images from a webcam feed, detect faces using the Haar cascade classifier, and enroll them for recognition. The captured face images are resized and stored in a database. The system then employs the K-Nearest Neighbors algorithm to perform real-time face recognition on the video feed, comparing the detected faces with the enrolled faces. Additionally, it incorporates blink detection by analyzing the eye aspect ratio based on landmarks. Attendance is tracked by counting the number of blinks detected, and the system saves the attendance records with recognized names and timestamps in a CSV file. Text-to-speech feedback is provided to indicate the successful attendance capture or the absence of blinks.

Lets analyze in depth:

```
from sklearn.neighbors import KNeighborsClassifier
import cv2
import pickle
import numpy as np
import os
import csv
import time
from datetime import datetime

from win32com.client import Dispatch
import dlib
from scipy.spatial import distance as dist
```

so for running this code first we need to get all our libraries which are needed to build the code

sklearn.neighbors:

Library for performing various nearest neighbor-based algorithms.

KNeighborsClassifier is a class used for k-nearest neighbors classification, which is utilized for face recognition in this code.

csv:

Python library for reading and writing CSV (comma-separated values) files.

Used for recording attendance records by writing recognized names and timestamps to a CSV file.

time:

Python library for working with time-related functions.

Used for timestamp generation for attendance records.

datetime:

Python library for manipulating dates and times.

Used for obtaining the current date and time for attendance records.

win32com.client:

Python library used for interfacing with the Windows COM (Component Object Model) services.

Utilized for text-to-speech functionality, providing auditory feedback on attendance status.

dlib:

Dlib library offers various computer vision algorithms and tools.

Used for face detection and shape prediction, enabling the calculation of eye aspect ratio for blink detection.

scipy.spatial.distance:

SciPy library provides various functions for scientific computing.

The dist module within scipy.spatial.distance is used to calculate Euclidean distances for blink detection.

sklearn.neighbors installation:

https://scikit-learn.org/stable/install.html

CSV installation:

https://pypi.org/project/python-csv/

Time installation:

Python's standard utility model.

Datetime installation:

Python's standard utility model.

win32com.client installation:

https://superuser.com/questions/609447/how-to-install-the-win32com-python

dlib installation:

https://www.geeksforgeeks.org/how-to-install-dlib-library-for-python-in-windows-10/

Utility Functions:

```
def shape_to_landmarks(shape):
    landmarks = []
    for i in range(0, 68):
        x = shape.part(i).x
        y = shape.part(i).y
        landmarks.append((x, y))
    return landmarks
```

```
def speak(str1):
    speak = Dispatch(("SAPI.SpVoice"))
    speak.Speak(str1)

def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear
```

Three useful functions are defined in the code:

shape_to_landmarks(shape): creates a list of (x, y) coordinates from a list of face landmark points in dlib shape format.

speak(str1): speaks the supplied text out loud using the Windows COM interface.

eye_aspect_ratio(eye): Determines the eye's aspect ratio using the landmarks' coordinates.

Initialization:

```
video = cv2.VideoCapture(0)
facedetect = cv2.CascadeClassifier('data/haarcascade_frontalface_default.xml')

with open('data/names.pkl', 'rb') as w:
    LABELS = pickle.load(w)
with open('data/faces_data.pkl', 'rb') as f:
    FACES = pickle.load(f)

print('Shape of Faces matrix --> ', FACES.shape)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(FACES, LABELS)



COL_NAMES = ['NAME', 'TIME']
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 2
COUNTER = 0
TOTAL = 0
attendance_taken = False

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('data/shape_predictor_68_face_landmarks.dat')
```

Initialization of variables and objects for the attendance system, including cv2 video capture, is done in the code.Create an instance of KNeighborsClassifier from sklearn.neighbors, load the Haar cascade classifier for face detection, load the face data and labels from pickle files, define constants and variables for blink detection and attendance recording.

Main Loop:

```
while True:
    ret, frame = video.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = facedetect.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        crop_img = frame[y:y+h, x:x+w, :]
        resized_img = cv2.resize(crop_img, (50, 50)).flatten().reshape(1, -1)
        output = knn.predict(resized_img)

        rects = detector(gray, 0)
        for rect in rects:
            shape = predictor(gray, rect)

            landmarks = shape_to_landmarks(shape)
            left_eye = landmarks[36:42]
            right_eye = landmarks[42:48]
            ear_left = eye_aspect_ratio(left_eye)
            ear_right = eye_aspect_ratio(right_eye)

            if not attendance_taken:
                if ear_left < EYE_AR_THRESH or ear_right < EYE_AR_THRESH:
                    COUNTER += 1
                else:
                    if COUNTER >= EYE_AR_CONSEC_FRAMES:
                        TOTAL += 1
                        print("Blink detected")
                        attendance_taken = True
                    COUNTER = 0

        ts = time.time()
        date = datetime.fromtimestamp(ts).strftime("%d-%m-%Y")
        timestamp = datetime.fromtimestamp(ts).strftime("%H:%M-%S")
        exist = os.path.isfile("Attendance/Attendance_" + date + ".csv")

        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 1)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (50, 50, 255), 2)
        cv2.rectangle(frame, (x, y-40), (x+w, y), (50, 50, 255), -1)
        cv2.putText(frame, str(output[0]), (x, y-15), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 
255, 255), 1)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (50, 50, 255), 1)
        attendance = [str(output[0]), str(timestamp)]


    cv2.imshow("Frame", frame)
```

For the purpose of processing videos and tracking attendance, the code starts a never-ending loop.

The Haar cascade classifier is used to recognise faces as it loops around the video stream, capturing frames, converting them to grayscale, and detecting faces.

In order to estimate the person's identity, it pulls the cropped face image from each identified face, resizes it to a set size (50x50 pixels), and then feeds it into the trained KNeighborsClassifier model.

It then computes the eye aspect ratio for blink detection, updates blink-related variables, and utilises dlib to identify face landmarks.

Additionally, it keeps track of attendance by adding a timestamp and recognised name to a CSV file.

Using cv2.imshow, the processed frame is presented with bounding boxes, labels, and the status of blink detection.

User Interaction:

```
k = cv2.waitKey(1)
    if k == ord('o') and TOTAL > 0:
        speak("Attendance Taken..")
        time.sleep(5)
        if exist:
            with open("Attendance/Attendance_" + date + ".csv", "+a") as csvfile:
                writer = csv.writer(csvfile)
                writer.writerow(attendance)
            csvfile.close()
        else:
            with open("Attendance/Attendance_" + date + ".csv", "+a") as csvfile:
                writer = csv.writer(csvfile)
                writer.writerow(COL_NAMES)
                writer.writerow(attendance)
            csvfile.close()
        attendance_taken = False
        COUNTER = 0
        TOTAL = 0
    elif k == ord('o') and TOTAL == 0:
        speak("No blink detected. Attendance not taken.")
    if k == ord('q'):
        break
```

The programme keeps track of keypress occurrences.

The 'o' key starts recording attendance, speaks a confirmation message, and sends the attendance information to the CSV file if blinks are noticed (TOTAL > 0) and the key is depressed.

The 'o' key speaks a message stating no attendance was taken if no blinks are noticed (TOTAL == 0) but the key is pushed.

When the 'q' key is pushed, the programme breaks out of the loop and stops recording video.

Cleanup:

```
video.release()
```

```
cv2.destroyAllWindows()
```

's termination, the code uses video.release() and cv2.destroyAllWindows() to release the video capture resources and close all open windows.

Data Dictionary:

Class: Student

Attributes:

student_id (int): The unique identifier of the student.

student_name (str): The name of the student.

Class: FaceData

Attributes:

face_id (int): The unique identifier of the face data.

student_id (int): The unique identifier of the student associated with the face data.

face_image (image): The image data representing the student's face.

Class: Attendance

Attributes:

attendance_id (int): The unique identifier of the attendance entry.

class_id (int): The unique identifier of the class for which the attendance is recorded.

student_id (int): The unique identifier of the student whose attendance is recorded.

date (date): The date on which the attendance is recorded.

time (time): The time at which the attendance is recorded.

Class: BlinkDetector

Functions:

shape_to_landmarks(shape): Converts the shape object to a list of landmarks.

eye_aspect_ratio(eye): Calculates the eye aspect ratio based on the given eye landmarks.

detect_blinks(frame): Detects blinks in the given frame using the eye aspect ratio method.

Overall architecture:

Different functionalities are organised into discrete modules or functions in the code provided, which uses a modular architecture. An overview of the architecture is provided here:

Data Loading and Preprocessing:

Importing the necessary libraries and modules, such as cv2, pickle, numpy, os, csv, time, datetime, win32com.client, dlib, and scipy.spatial, is the first step in the code.

It defines useful functions that are used throughout the code for various tasks, such as shape_to_landmarks, talk, and eye_aspect_ratio.

Face Data Capture and Storage:

Using cv2.VideoCapture, the code initialises the video capture from the default camera.

To find faces in the collected frames, it employs the Haar cascade classifier (cv2.CascadeClassifier).

It crops and resizes the face image to a set size for each face that is detected.

The faces_data list is supplemented with the resized face photos.

Until either the maximum limit (100 faces) or a termination condition (keypress "q") is satisfied, the algorithm keeps collecting face data.

The recorded facial data is moulded into a uniform manner and transferred to a numpy array.

Storage and Update of Face Data Files:

The programme determines whether the data files "names.pkl" and "faces_data.pkl" are present in the requested directory.

If the files don't already exist, it makes new ones and stores the face data that was taken along with any associated names.

If the files already exist, it updates the files by loading the old data and appending the fresh data.

Face Recognition and Blink Detection:

The pre-trained face recognition model (KNeighborsClassifier) is loaded with the previously stored face data when the code resets the video clip to its initial state.

It establishes constants and settings for blink detection, including eye aspect ratio thresholds and recognised blink duration in consecutive frames.

For face and facial landmark detection, the code makes use of the Dlib library (dlib.get_frontal_face_detector() and dlib.shape_predictor()).

By applying the KNN classifier to estimate the identity from the acquired face image for each detected face, face recognition is carried out.

In order to identify blinks, it simultaneously calculates the eye aspect ratio and recognises facial landmarks.

By counting consecutive frames while keeping its eyes closed, it keeps track of the blink events.

Attendance Recording and Output:

To guarantee that attendance is only recorded once every blink event, the code keeps track of the attendance_taken state.

It checks to see if there is already a CSV file for the attendance for the current date.

It adds the attendance information (recognised name and timestamp) to the attendance CSV file if it already exists.

It creates a new file and writes the attendance information along with the column names if the file doesn't already exist.

By using cv2.imshow to display processed frames with bounding boxes, labels, and blink detection status, the code shows visual feedback.

User Interaction and Termination:

The code watches for keypress events to start particular processes.

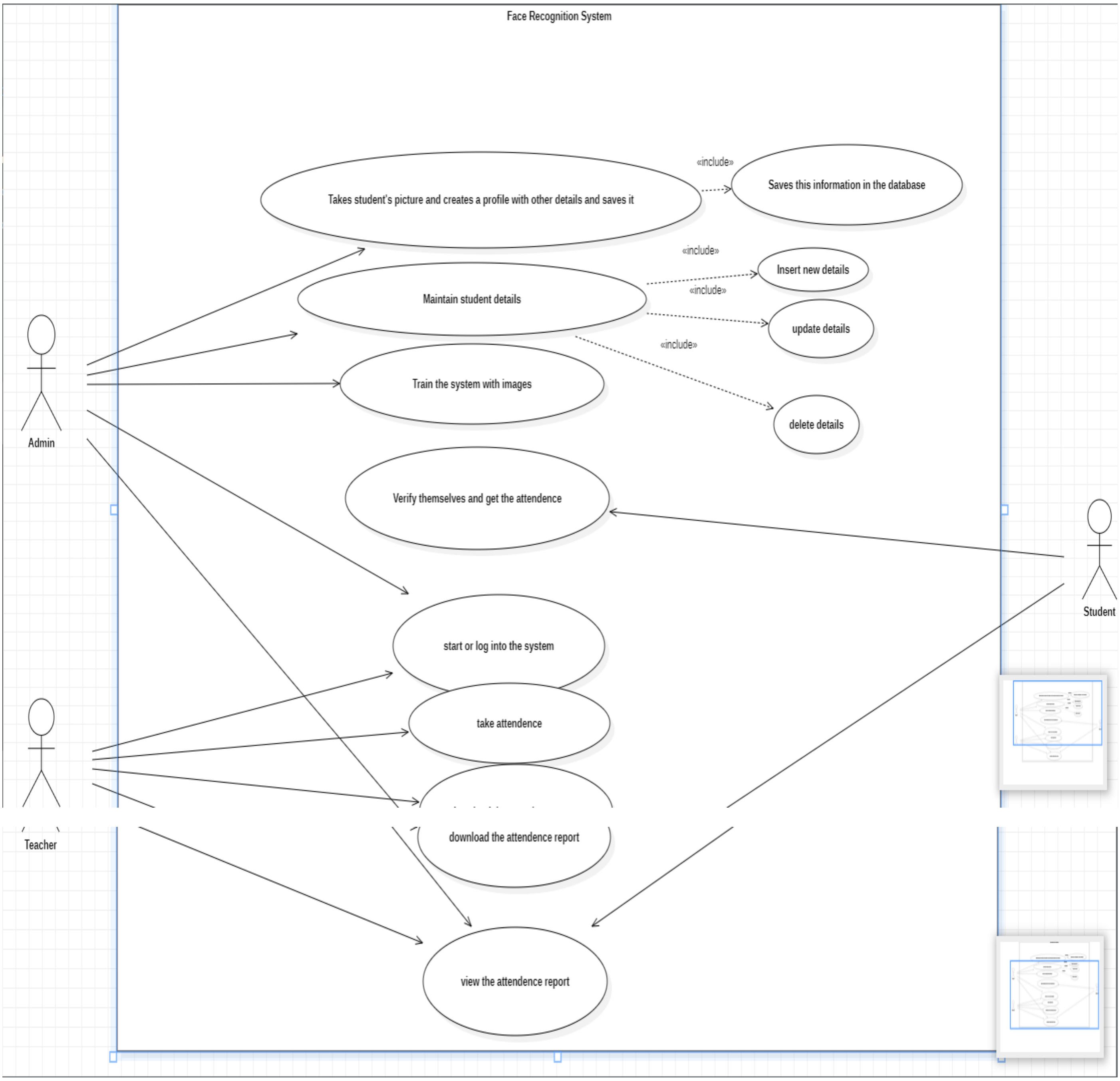
The 'o' key validates the attendance recording using speech synthesis when blinks are detected, and the attendance data is then written to the CSV file.

The 'o' key notifies the user that no attendance was taken when it is pushed but no blinks are noticed.

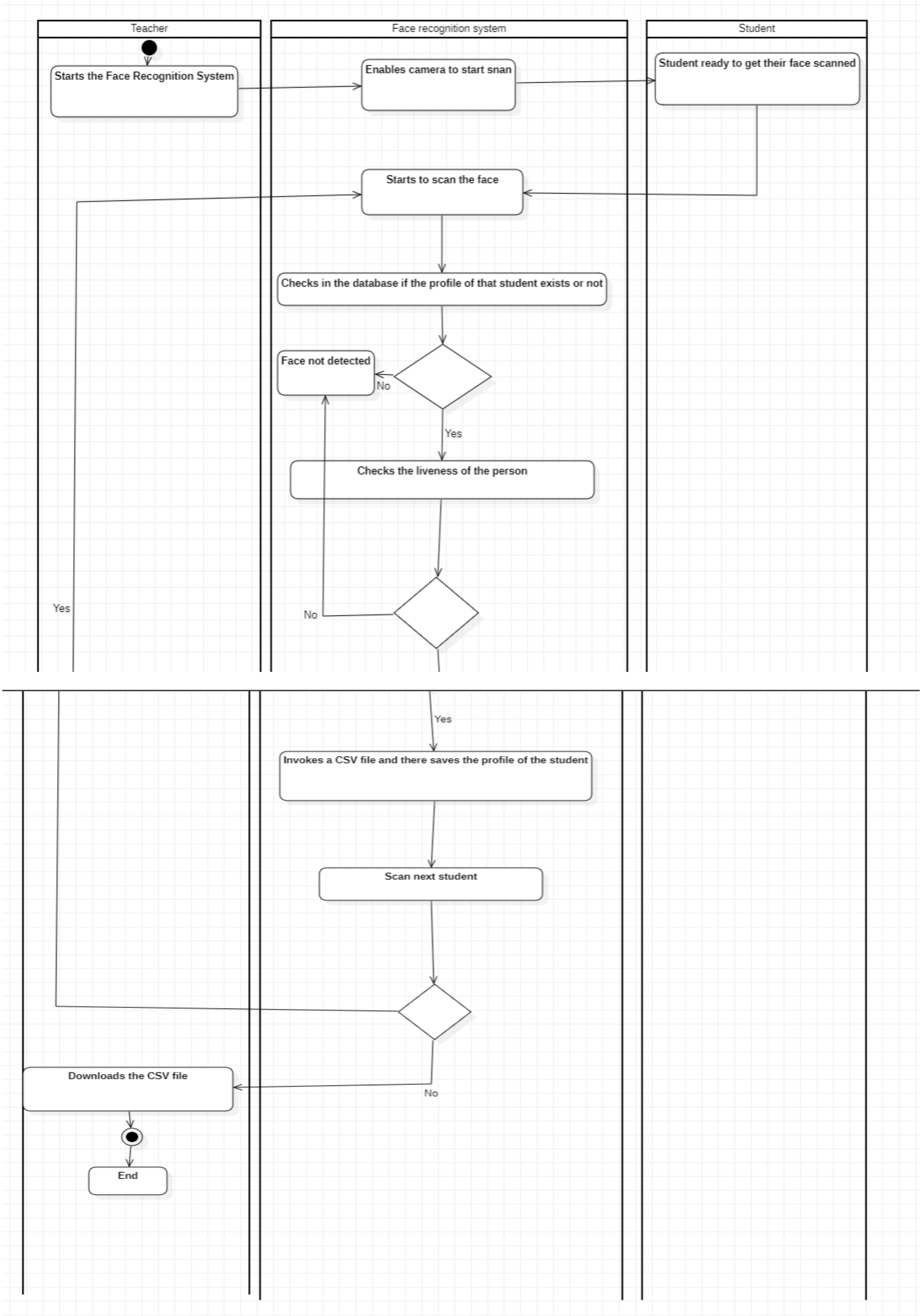
The code loops endlessly until the 'q' key is pressed, which ends the programme and releases the resources used for video capture.

UML DIAGRAMS:

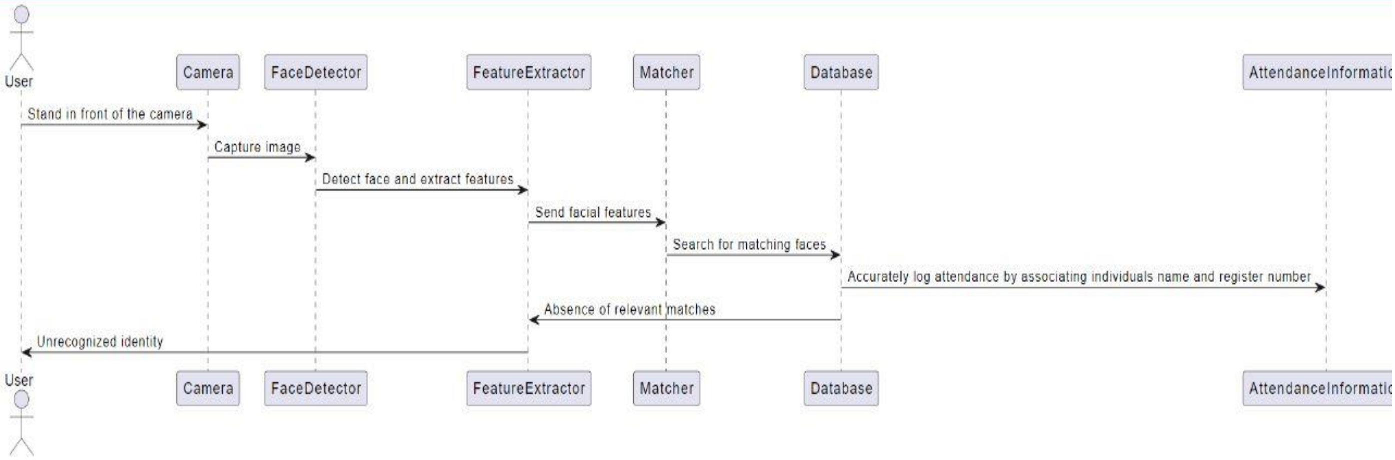
Usecase Diagram for Face recognition attendance system:


Face Recognition System
Takes student's picture and creates a profile with other details and saves it
«include»
Saves this information in the database
«include»
Insert new details
«include»
update details
«include»
delete details
Maintain student details
Train the system with images
Verify themselves and get the attendence
start or log into the system
take attendence
download the attendence report
view the attendence report
Admin
Teacher
Student

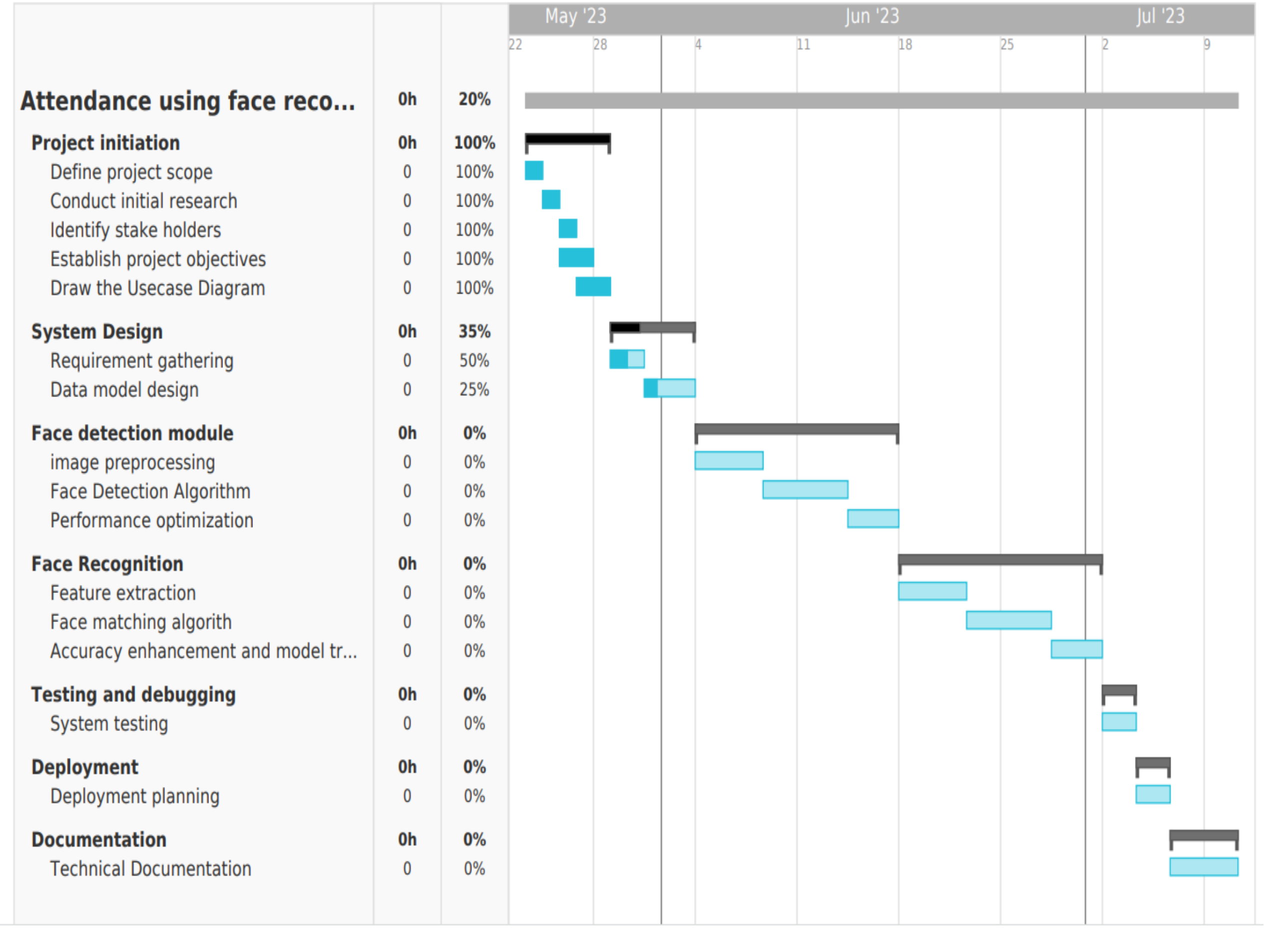
Activity Diagram for Face recognition attendance system:


Teacher
Face recognition system
Student
Starts the Face Recognition System
Enables camera to start snan
Student ready to get their face scanned
Starts to scan the face
Checks in the database if the profile of that student exists or not
Face not detected
No
Yes
Checks the liveness of the person
Yes
No
Yes
Invokes a CSV file and there saves the profile of the student
Scan next student
Downloads the CSV file
No
End

Sequence Diagram for Face recognition attendance system:

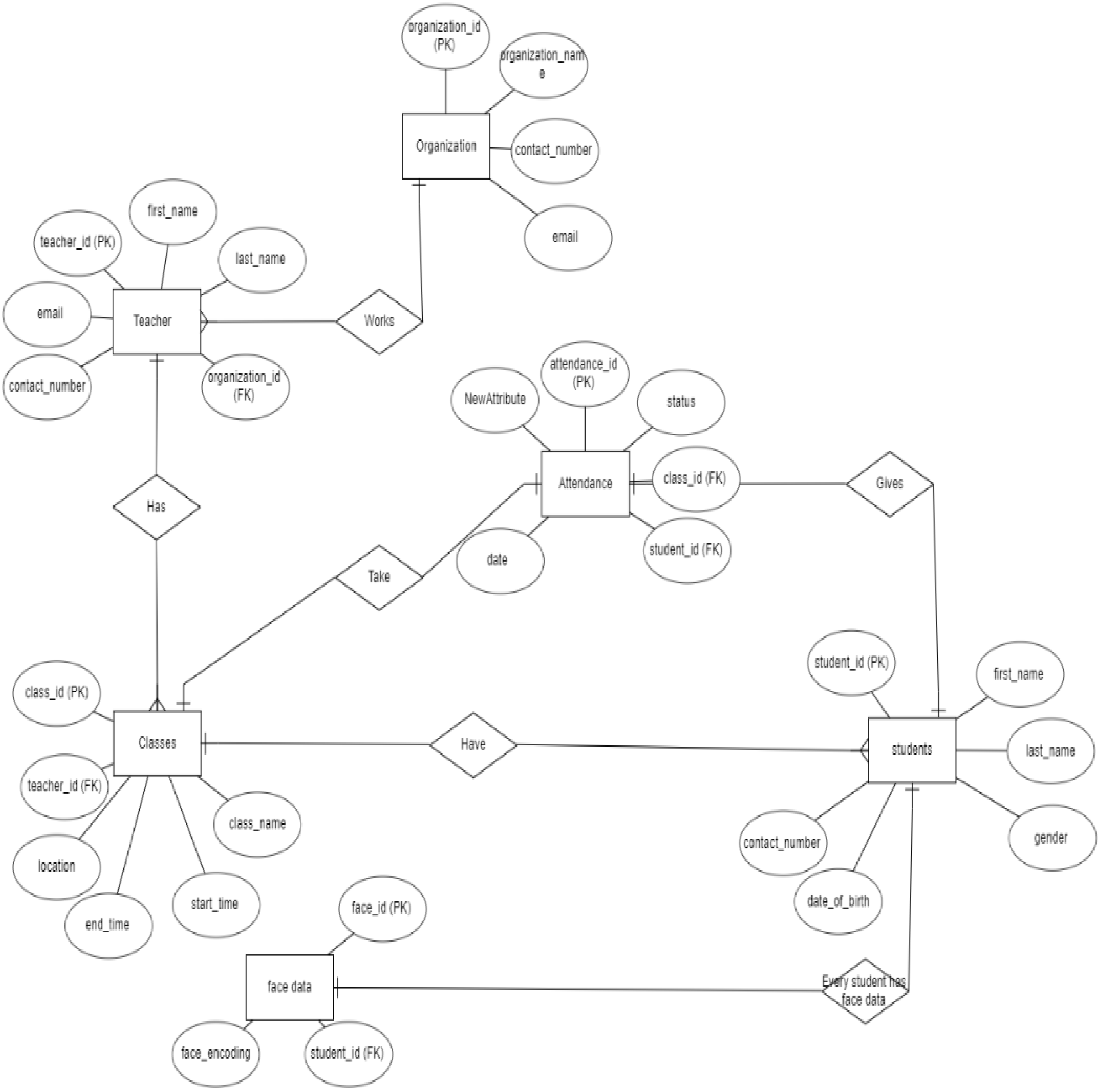


Gantt Chart:



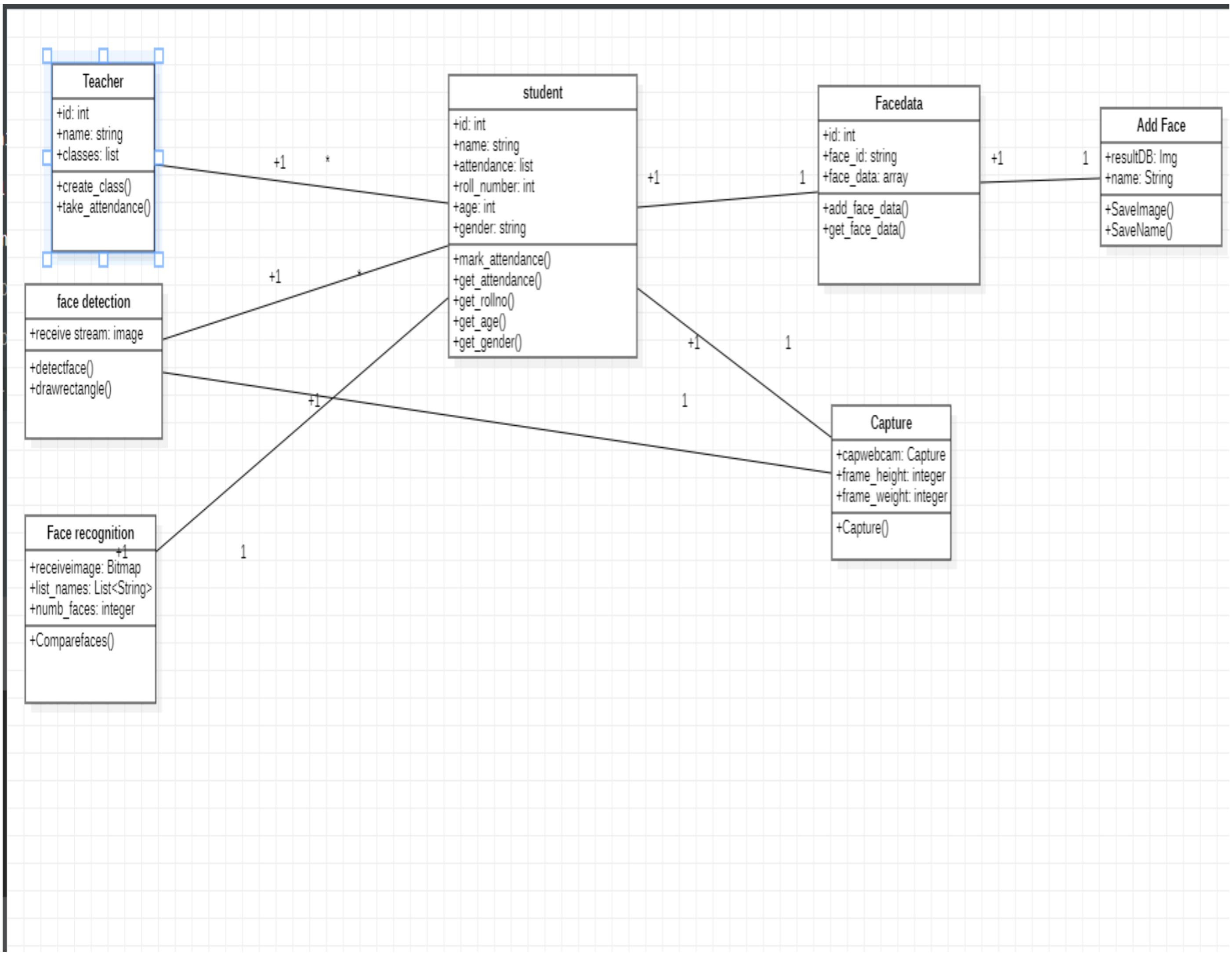
Attendance using face reco...	**0h**	**20%**
Project initiation	**0h**	**100%**
Define project scope	0	100%
Conduct initial research	0	100%
Identify stake holders	0	100%
Establish project objectives	0	100%
Draw the Usecase Diagram	0	100%
System Design	**0h**	**35%**
Requirement gathering	0	50%
Data model design	0	25%
Face detection module	**0h**	**0%**
image preprocessing	0	0%
Face Detection Algorithm	0	0%
Performance optimization	0	0%
Face Recognition	**0h**	**0%**
Feature extraction	0	0%
Face matching algorith	0	0%
Accuracy enhancement and model tr...	0	0%
Testing and debugging	**0h**	**0%**
System testing	0	0%
Deployment	**0h**	**0%**
Deployment planning	0	0%
Documentation	**0h**	**0%**
Technical Documentation	0	0%

ER Diagram:



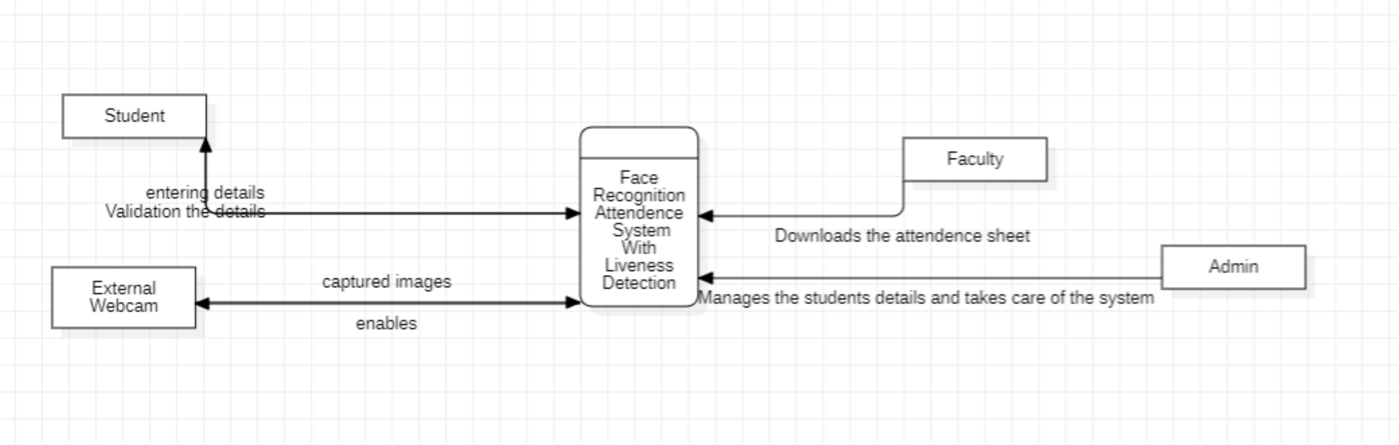
The er diagram is made in https://erdplus.com/documents

Class Diagram:

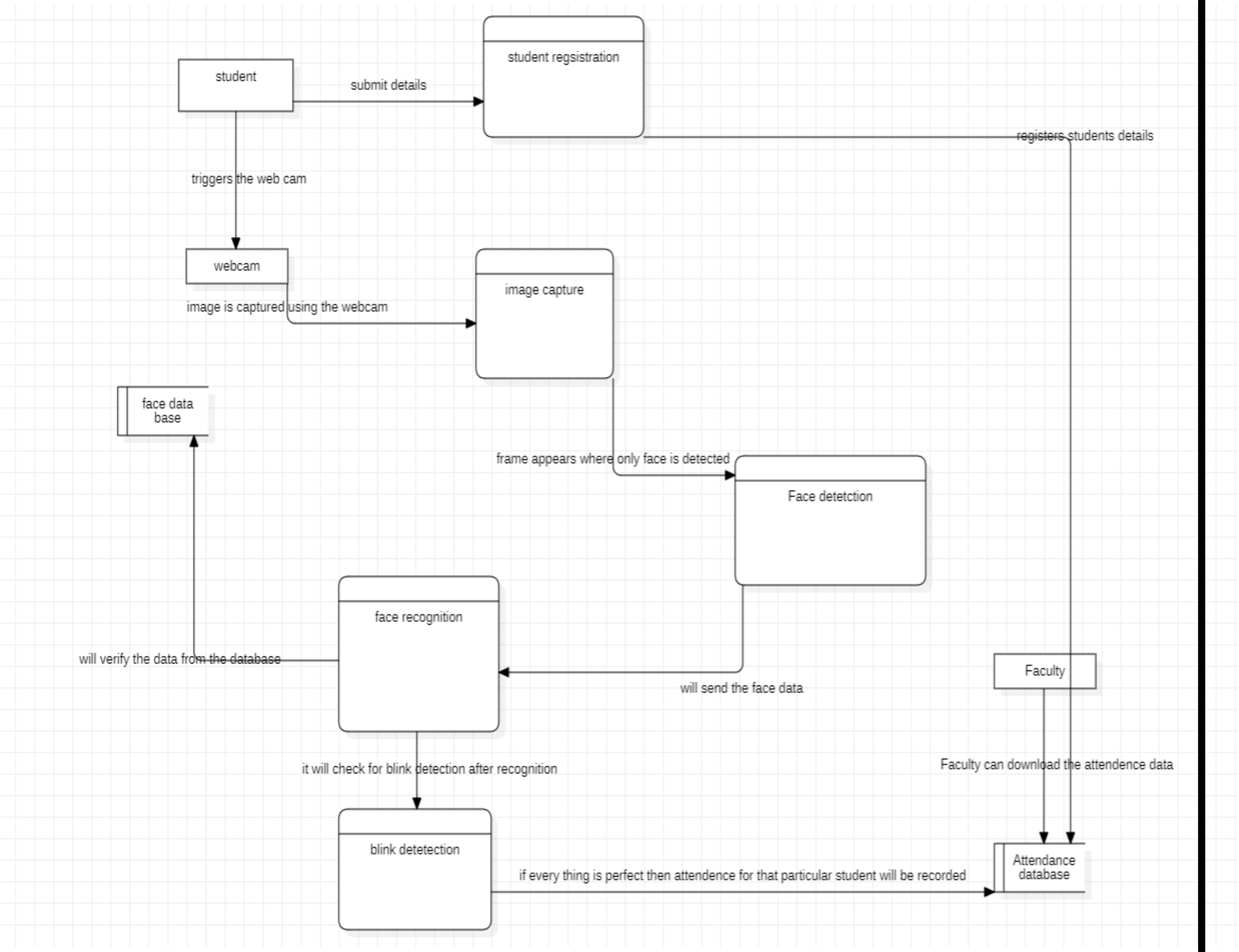


DFD diagram:

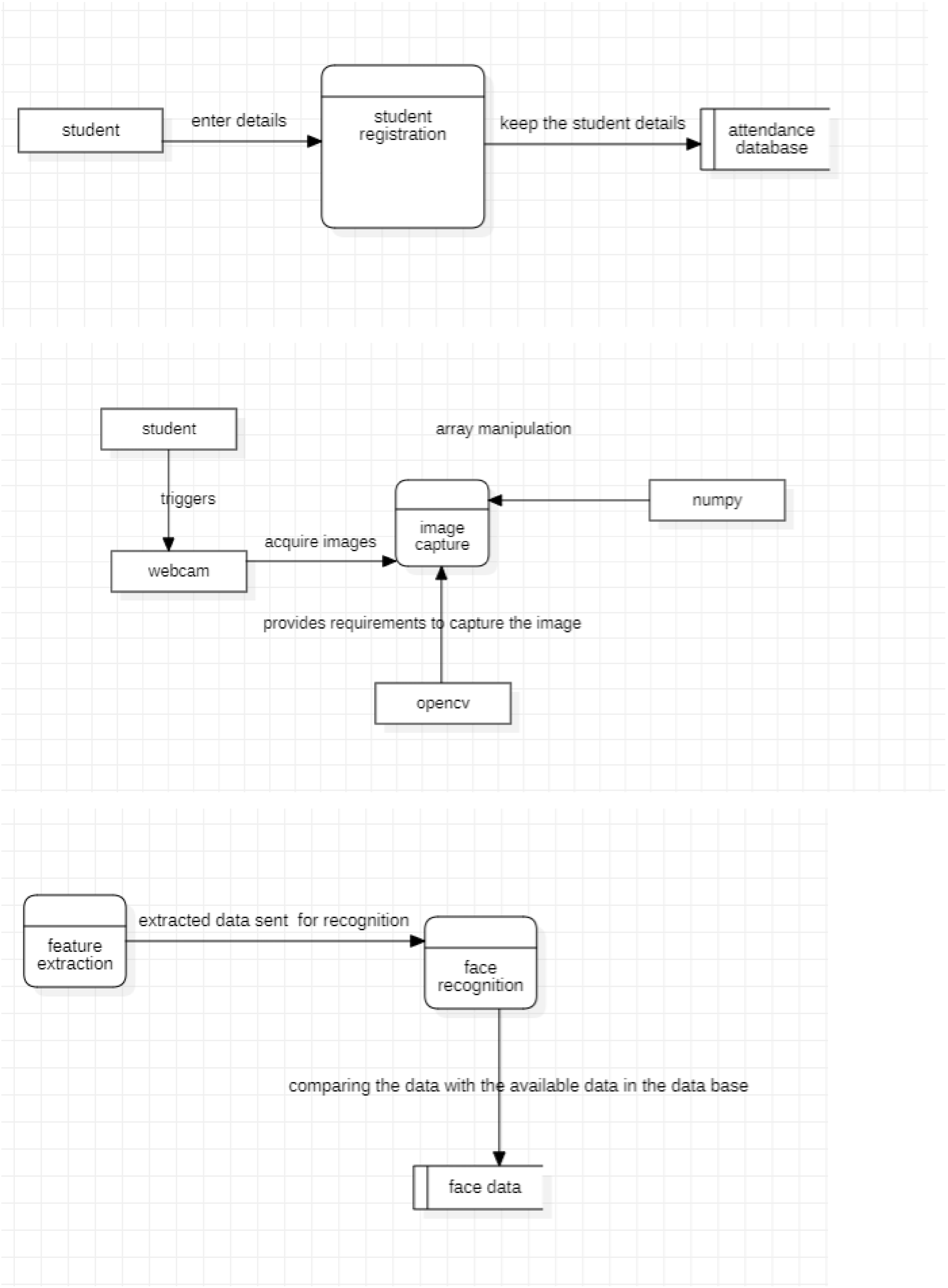
Level 0 DFD diagram:

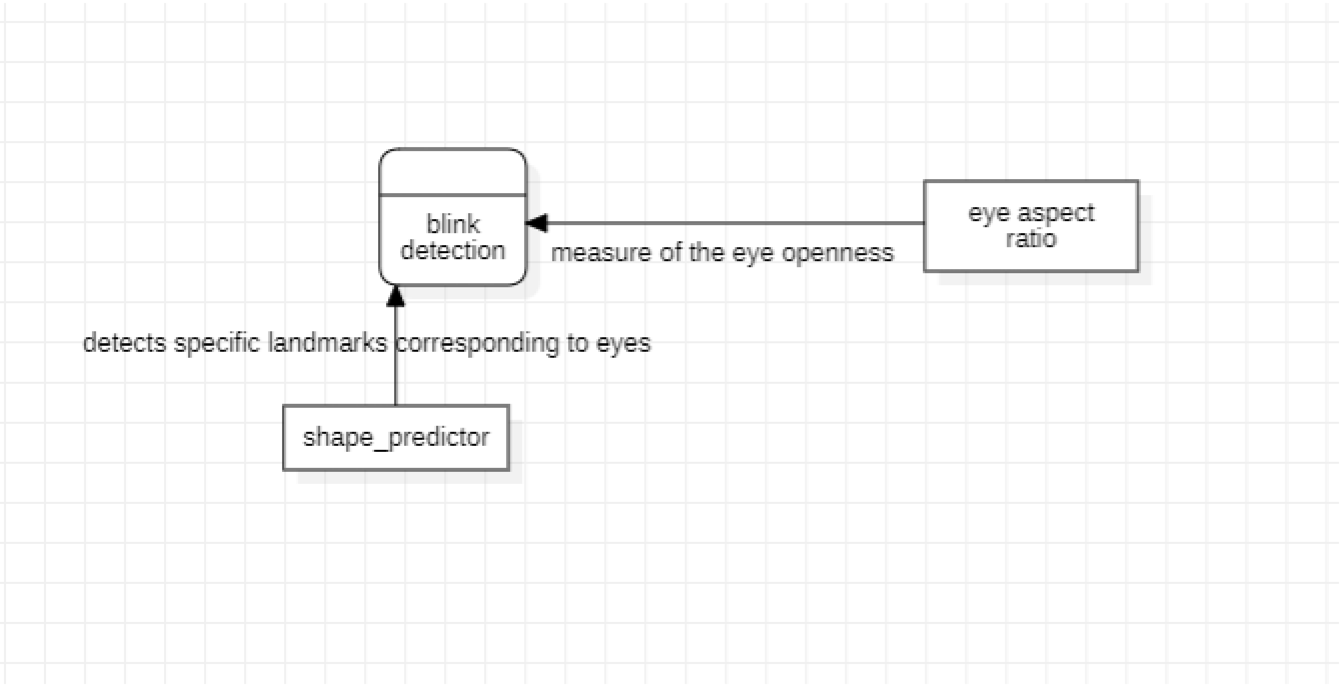

Student
entering details
Validation the details
External Webcam
captured images
enables
Face Recognition Attendence System With Liveness Detection
Faculty
Downloads the attendence sheet
Admin
Manages the students details and takes care of the system

Level 1 DFD diagram:


student
submit details
student regsistration
registers students details
triggers the web cam
webcam
image is captured using the webcam
image capture
face data base
frame appears where only face is detected
Face detetction
face recognition
will verify the data from the database
will send the face data
Faculty
Faculty can download the attendence data
it will check for blink detection after recognition
blink detetection
if every thing is perfect then attendence for that particular student will be recorded
Attendance database

Level 2 DFD diagram:




blink detection
eye aspect ratio
measure of the eye openness
detects specific landmarks corresponding to eyes
shape_predictor

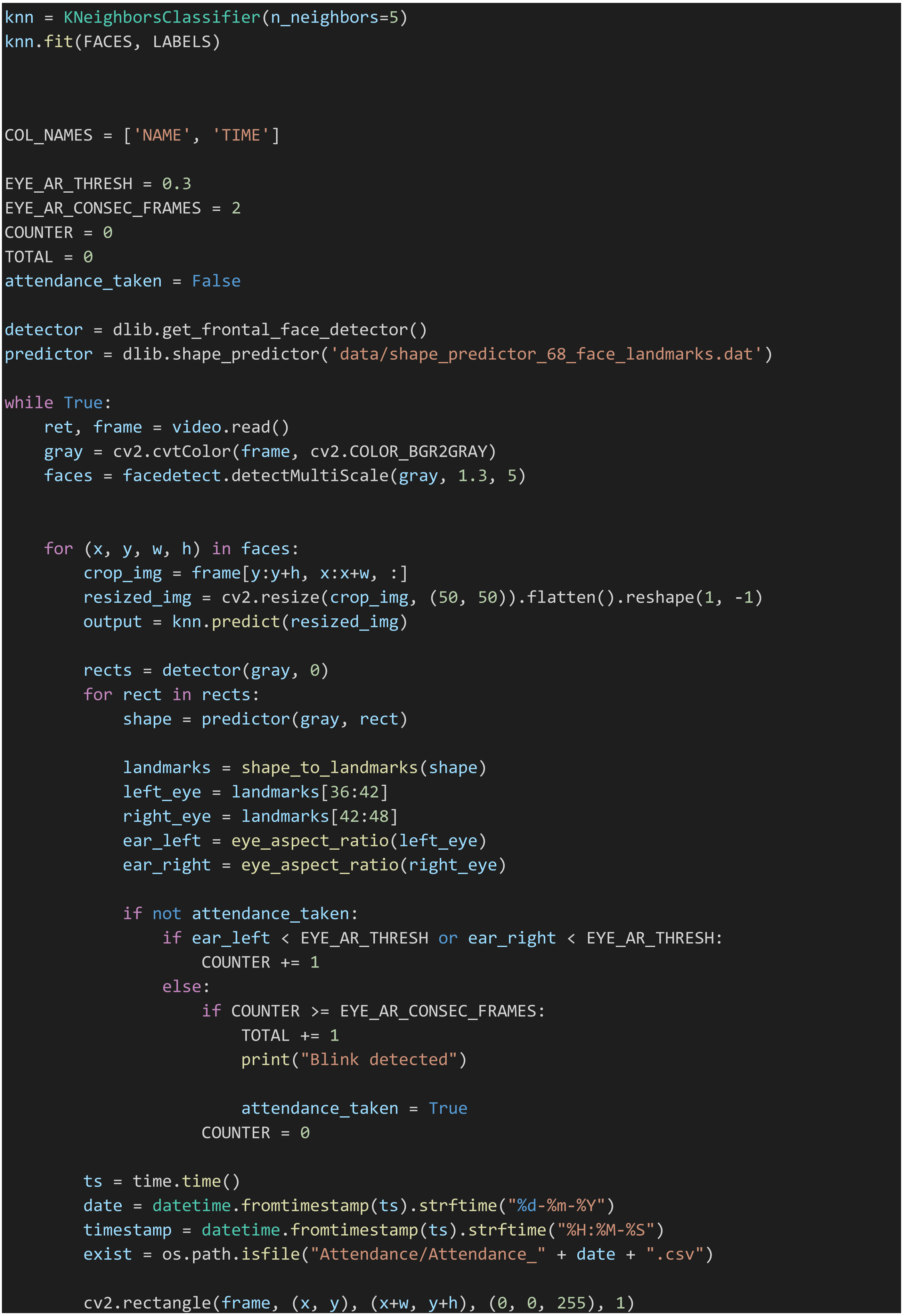
SAMPLE SOURCE CODE:

Captures_facedata.py:

```
import cv2
import pickle
import numpy as np
import os
video=cv2.VideoCapture(0)
facedetect=cv2.CascadeClassifier('data/haarcascade_frontalface_default.xml')

faces_data=[]

i=0

name=input("Enter Your Name: ")

while True:
    ret,frame=video.read()
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces=facedetect.detectMultiScale(gray, 1.3 ,5)
    for (x,y,w,h) in faces:
        crop_img=frame[y:y+h, x:x+w, :]
        resized_img=cv2.resize(crop_img, (50,50))
        if len(faces_data)<=100 and i%10==0:
            faces_data.append(resized_img)
        i=i+1
        cv2.putText(frame, str(len(faces_data)), (50,50), cv2.FONT_HERSHEY_COMPLEX, 1,
(50,50,255), 1)
        cv2.rectangle(frame, (x,y), (x+w, y+h), (50,50,255), 1)
    cv2.imshow("Frame",frame)
    k=cv2.waitKey(1)
    if k==ord('q') or len(faces_data)==100:
        break
video.release()
cv2.destroyAllWindows()

faces_data=np.asarray(faces_data)
faces_data=faces_data.reshape(100, -1)


if 'names.pkl' not in os.listdir('data/'):
    names=[name]*100
    with open('data/names.pkl', 'wb') as f:
        pickle.dump(names, f)
else:
    with open('data/names.pkl', 'rb') as f:
        names=pickle.load(f)
    names=names+[name]*100
    with open('data/names.pkl', 'wb') as f:
        pickle.dump(names, f)

if 'faces_data.pkl' not in os.listdir('data/'):
    with open('data/faces_data.pkl', 'wb') as f:
        pickle.dump(faces_data, f)
else:
    with open('data/faces_data.pkl', 'rb') as f:
        faces=pickle.load(f)
    faces=np.append(faces, faces_data, axis=0)
    with open('data/faces_data.pkl', 'wb') as f:
        pickle.dump(faces, f)
```

rld.py

```
from sklearn.neighbors import KNeighborsClassifier
import cv2
import pickle
import numpy as np
import os
import csv
import time
from datetime import datetime

from win32com.client import Dispatch
import dlib
from scipy.spatial import distance as dist

def shape_to_landmarks(shape):
    landmarks = []
    for i in range(0, 68):
        x = shape.part(i).x
        y = shape.part(i).y
        landmarks.append((x, y))
    return landmarks

def speak(str1):
    speak = Dispatch(("SAPI.SpVoice"))
    speak.Speak(str1)

def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

video = cv2.VideoCapture(0)
facedetect = cv2.CascadeClassifier('data/haarcascade_frontalface_default.xml')

with open('data/names.pkl', 'rb') as w:
    LABELS = pickle.load(w)
with open('data/faces_data.pkl', 'rb') as f:
    FACES = pickle.load(f)

print('Shape of Faces matrix --> ', FACES.shape)
```



```python
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(FACES, LABELS)



COL_NAMES = ['NAME', 'TIME']

EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 2
COUNTER = 0
TOTAL = 0
attendance_taken = False

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('data/shape_predictor_68_face_landmarks.dat')

while True:
    ret, frame = video.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = facedetect.detectMultiScale(gray, 1.3, 5)


    for (x, y, w, h) in faces:
        crop_img = frame[y:y+h, x:x+w, :]
        resized_img = cv2.resize(crop_img, (50, 50)).flatten().reshape(1, -1)
        output = knn.predict(resized_img)

        rects = detector(gray, 0)
        for rect in rects:
            shape = predictor(gray, rect)

            landmarks = shape_to_landmarks(shape)
            left_eye = landmarks[36:42]
            right_eye = landmarks[42:48]
            ear_left = eye_aspect_ratio(left_eye)
            ear_right = eye_aspect_ratio(right_eye)

            if not attendance_taken:
                if ear_left < EYE_AR_THRESH or ear_right < EYE_AR_THRESH:
                    COUNTER += 1
                else:
                    if COUNTER >= EYE_AR_CONSEC_FRAMES:
                        TOTAL += 1
                        print("Blink detected")

                        attendance_taken = True
                    COUNTER = 0

        ts = time.time()
        date = datetime.fromtimestamp(ts).strftime("%d-%m-%Y")
        timestamp = datetime.fromtimestamp(ts).strftime("%H:%M-%S")
        exist = os.path.isfile("Attendance/Attendance_" + date + ".csv")

        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 1)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (50, 50, 255), 2)
        cv2.rectangle(frame, (x, y-40), (x+w, y), (50, 50, 255), -1)
        cv2.putText(frame, str(output[0]), (x, y-15), cv2.FONT_HERSHEY_COMPLEX, 1, (255,
255, 255), 1)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (50, 50, 255), 1)
        attendance = [str(output[0]), str(timestamp)]


    cv2.imshow("Frame", frame)

    k = cv2.waitKey(1)
    if k == ord('o') and TOTAL > 0:
        speak("Attendance Taken..")
        time.sleep(5)
        if exist:
            with open("Attendance/Attendance_" + date + ".csv", "+a") as csvfile:
                writer = csv.writer(csvfile)
                writer.writerow(attendance)
            csvfile.close()
        else:
            with open("Attendance/Attendance_" + date + ".csv", "+a") as csvfile:
                writer = csv.writer(csvfile)
                writer.writerow(COL_NAMES)
                writer.writerow(attendance)
            csvfile.close()
        attendance_taken = False
        COUNTER = 0
        TOTAL = 0
    elif k == ord('o') and TOTAL == 0:
        speak("No blink detected. Attendance not taken.")
    if k == ord('q'):
        break

video.release()
cv2.destroyAllWindows()
```

Test plan/Methods and Output Screenshots:

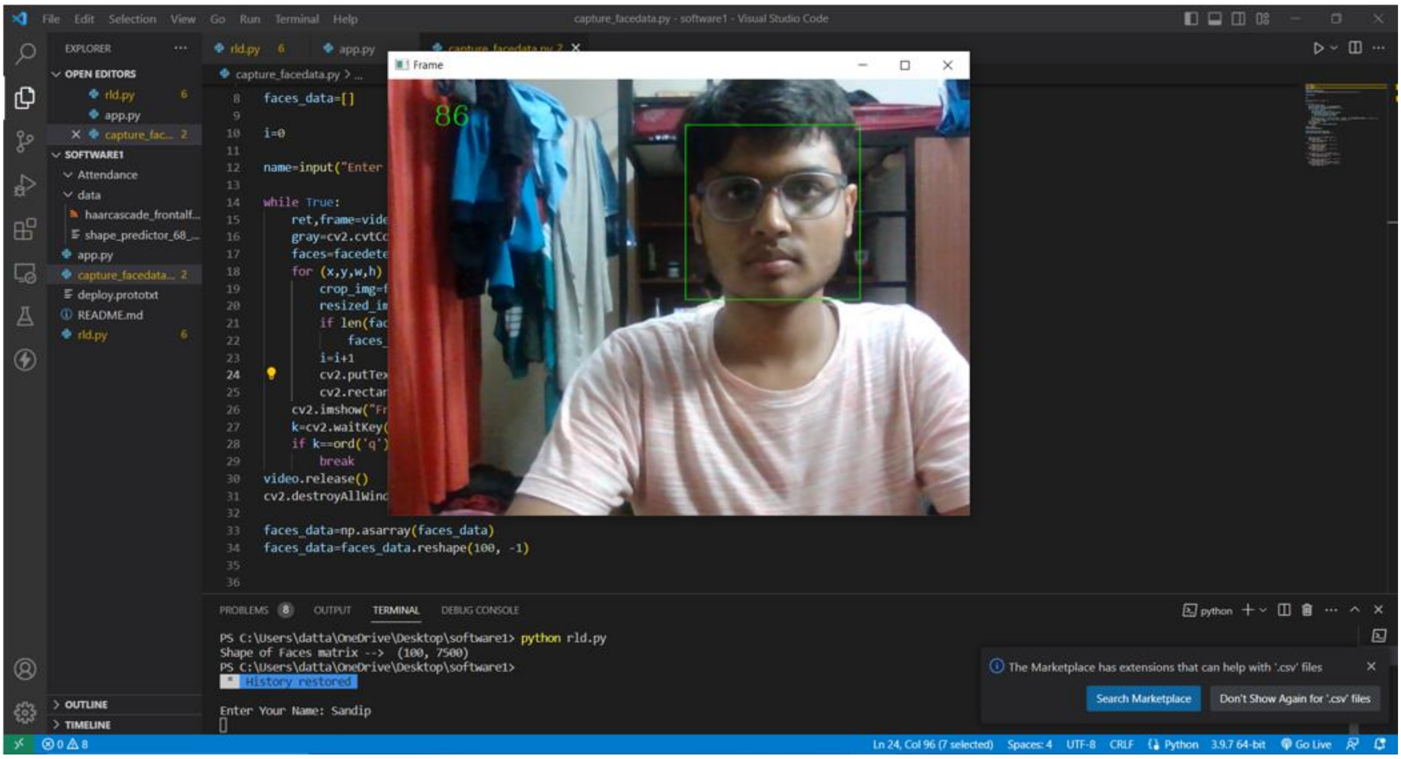
Verify Face Data Capture and Storage:

Check the camera's ability to capture face information.

Make that the face photos that were recorded are appropriately scaled and added to the faces_data list.

'names.pkl' and 'faces_data.pkl' test the storing and updating of face data files.

Check to see if the data files have been updated or produced with the face data that was recorded.

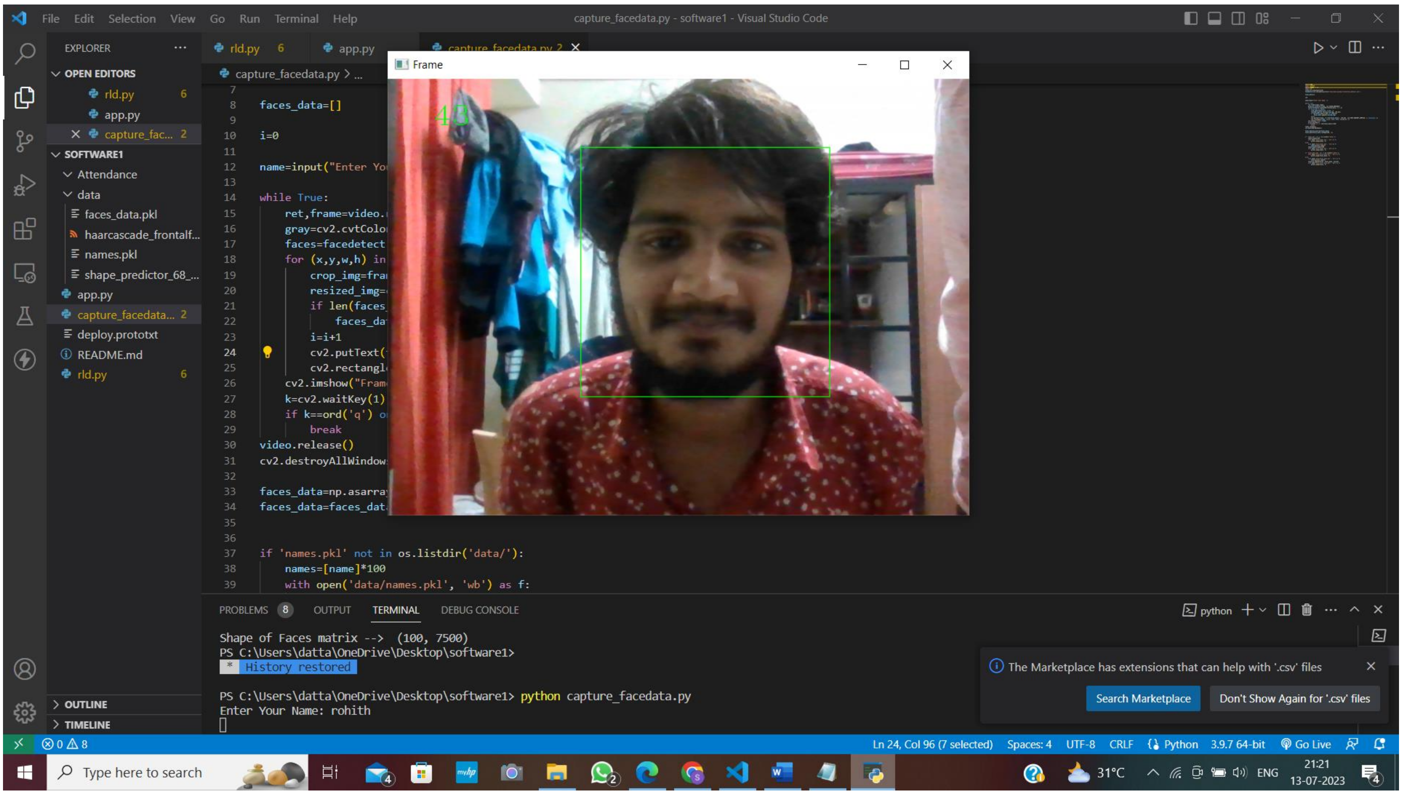


```
faces_data=[]

i=0

name=input("Enter

while True:
    ret,frame=vide
    gray=cv2.cvtCo
    faces=facedete
    for (x,y,w,h)
        crop_img=f
        resized_im
        if len(fac
            faces_
        i=i+1
        cv2.putTex
        cv2.rectan
    cv2.imshow("Fr
    k=cv2.waitKey(
    if k==ord('q')
        break
video.release()
cv2.destroyAllWinc

faces_data=np.asarray(faces_data)
faces_data=faces_data.reshape(100, -1)
```

```
PS C:\Users\datta\OneDrive\Desktop\software1> python rld.py
Shape of Faces matrix -->  (100, 7500)
PS C:\Users\datta\OneDrive\Desktop\software1>
 *  History restored

Enter Your Name: Sandip
```

The Marketplace has extensions that can help with '.csv' files

Search Marketplace | Don't Show Again for '.csv' files



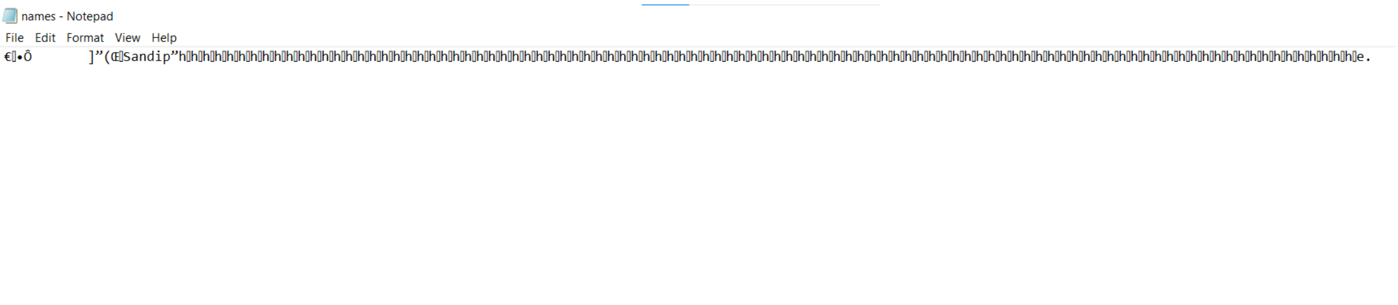
```
faces_data=[]

i=0

name=input("Enter Yo

while True:
    ret,frame=video.
    gray=cv2.cvtColo
    faces=facedetect
    for (x,y,w,h) in
        crop_img=fra
        resized_img=
        if len(faces
            faces_da
        i=i+1
        cv2.putText(
        cv2.rectangl
    cv2.imshow("Fram
    k=cv2.waitKey(1)
    if k==ord('q') o
        break
video.release()
cv2.destroyAllWindow

faces_data=np.asarra
faces_data=faces_dat


if 'names.pkl' not in os.listdir('data/'):
    names=[name]*100
    with open('data/names.pkl', 'wb') as f:
```

```
Shape of Faces matrix -->  (100, 7500)
PS C:\Users\datta\OneDrive\Desktop\software1>
 *  History restored

PS C:\Users\datta\OneDrive\Desktop\software1> python capture_facedata.py
Enter Your Name: rohith
```

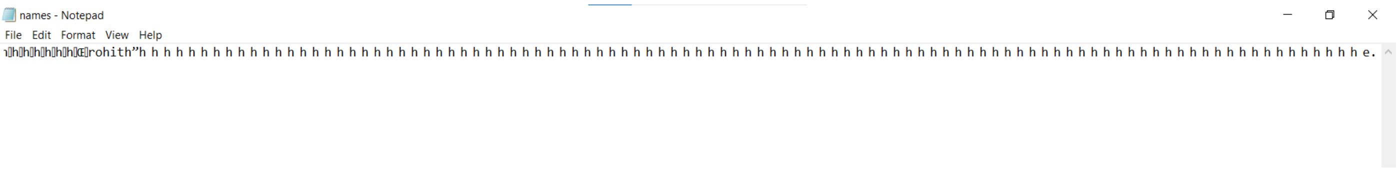
The Marketplace has extensions that can help with '.csv' files

Search Marketplace | Don't Show Again for '.csv' files

Names.pickle file:



```
€•Õ       ]”(Sandip”hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhe.
```



```
hhhhhhrohith”h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h h e.
```

Face_data.pickle file:

```
€•„        numpy.core.multiarray”Œ_reconstruct”“”Œnumpy”Œndarray”“”K …”Cb”‡”R”(KKdML†”hŒdtype”“”Œ u1”‰ˆ‡”R”(K|”NNNJÿÿÿÿJÿÿÿÿK t”b‰B°q (%&#"" "%'(*-,::<;>9>
YVUHIGGIFJLHLLJC=;>53@53 (.0"2/- 0-  /#2"4"&:')>.4H1=Q7LdPcwZjfq{Ž“•¥¤¥³«¯·´¹ÂÆÈÐÚÝåäâñàäîÏÏÜª®ºzzMRKJKFSSQXYVWUPD?:A88=43 +!2"1 0"3!2#2$3$3!&:!%<#
ˆ€ƒˆ…†‡“’•ž ¥¬³À³¼Ï¨ºÑž®Æ¤?BEAAA89:57879?88B "*/ $4"+;&0A ,>!-A$1D#/A'2G!/G0G2I6L%?X)C^0Hd1Jf9Rn?VpJ[qO\sKXkBKZ<BS@GU7BMCJSSO\e_irqv„‹’•¤¤¨ª©ºÉ¨²Ì"®RP_@>E75;155143202!  "+$/-'7
¥Œ-«ds=K\6:L43@0/811:8:D/398>A_`gsottpyljxpqlp…yz-’’¡”- —‹˜»t¶Sq±Nc•"/#,+-247*/54;D7AM19G*4D&-A$);&)8!$2(  "%'*:5>E4?E<DYMa,z”±Ž¢fm~\_v?BW45C/2;75<66;34;GGRacktpxƒ,Œ~Œqm{xv…š—¥©¨´š «-
+3@.9H+9I/;I-5E>@]IU{cv˜ž¬Æ,ÅÜ«µÑº¥®ÄÆé®²ÄŽ’¯,†›„™‰ª-³‹™µ“Ÿ%¥«Ä¤ˆÏ¦¯Ãfi-¿´%Ö±%Ø«µÑž®½Š™‰…‡ =5D*+2*(3I'/L$+K1;X9FZlrzmrxEKW:EW5?V,8O(7K!2C(7G-9H*7F*:H.;G0:I/8H>Hg@R|Wn“Ÿ%¿É±%Ê±à>ª»-³¾³ˆº››-†ƒ
 %4)3K+A!0G->\2Fc0Gc,D^)@Y2C\,<U+;U&:R&4P!-K$A%G!4T,<]-S+T(8a1@e9FdDPk3<Z+2TGPvgou šs€šfy—h{š|Š-…•¶ƒ-·~“˜~•µ~—·™¸ƒ›¼ˆ À‰ »†-º…“¬Žš%9:L#&/7M'0I%,J2>[3B`1Cc.B_1C]1C\.>Z):V':S%7S-I,I*H!-M)
(-*>!5#-D-;W,>Y,@Z+A\+?]->[.=Z)9W"3O%8T,:V,7P#-E*?#,B',E37PAHb=@`BKkv,“}ˆ>z•rynx‰‡˜Ž—ª˜¤¹-¡¸Šµ‡ˆ²€“³|’²y‘²{”²,˜µ…›‡-ˆ{†“MGR,!)#"+%&'#(/%<'5R/=[.?[*=\,Aa*?a);](9W%9R$6N,<R#2J+B+B%/
Œ9CW)&2&!$($#&$%$(",!%(%*'#0!.<(7K+=S*;U-A[-C`.Bb)=Z#4M-F%2K%1K!,J$/P"-T)X,ˆ2f(9p9EtMV|O\}J^{Sh…ZnŒZo‹Ti†]sŽj}˜{Š©€’°}ˆu&¤nƒ ew”T`r*/:%%1%#+$!#%!$$##'"$#'-#, +5'4F+7R-;T0B[-D`.Ca)>[$6T
O[rMUhLQajq‡——ª¥‘¡yeqTARO@MH@ED;@C=;J@APGGRFH[JQ$!&&!%(#(*%,-*3/(45,3?4DGA_43I<BV/8M'/K!+E +H-K /L#4O+9T(7Q%7Q%9U$4Q$2M"1M#3Q+8U3DZ>QbK_qObyJ\uEViEPdCJaCJ^dn…™µŒ¦ƒ,’úüûÿÿÿüüûžŽcMSfORcOQ˜}|ì

 %,.0/13446>>@OPMJG?QPKYWVhgi•™^^bstxUX[SPPGC@@?9?@;FFD<<:A?@=8<h\|



 $#$*/13/35CFBYZVWXXG@@WSSjglx|~ˆ‹‘zvz¤›§vw€][aJHFFDBJHDIHC=;9:79;7>?7G

  ## '#%)*+1,.2248GKMKNNQOOJDB\YXwy,{{ƒ•–„€…º¸À-¯·,’jdl\RQZQKHE<<88=8<A;AA;E





(#"""""!& '(!!*()3,0<8;FFIQMNTPOVOJUx}‡…Œª-‚µÃ¡£ªÂÂÂÀÀË¬®¼ƒ~ˆmb`pi_GB7?=6FCBKFJ4(2







'.,'&('('-"#1$-#'2+/;/7G@FYNRdSVcVZc„Š”’“žŽ˜²º¿ÑÕÜÅÇÎÛÝáÑÓØÇÇÐ¢¡§rok]XRKG=IIAOKGUPLf`_









# '&'*/-*.!1"#3&(6*-;39K7EVQ^xXghs`€š$˜ -®±¼,µÁÂ¿ÄÒÒÓâáâêïðèêôÕÕÛ¶µºurtSNGKF?SNH\XR_YVg``









 )!*"-!$2"4!0"/"/"&5)*:+.>+4C1>S?KlMd‰f{œv˜Ÿ µº%Í¼ÅÑÊÎÙÜÖÜßáçèæêìêêìçåìÑÕÝ¶,%Ž‰ŒKKBHH=QRFOOFXZUXX\



,/ $4 #3#%6"$;$(> )= (9$*<(/A/7H:BW<IeIVxbo‘w,Ÿš˜³-Á²%ÒÅÍÝÒxàÔÛÝäàçç êëäåèßßéØØáÃÆÏ›˜XT[GF@VVNKL?IRHOWV



```

Validate Face Recognition and Blink Detection and attendance recording output:

Checking how the pre-trained face recognition model (KNeighborsClassifier) performs when loaded with the previously stored face data.

By contrasting the anticipated identities with the actual identities, you can confirm the accuracy of facial recognition.

Check whether the eye aspect ratio for blink detection is calculated correctly.

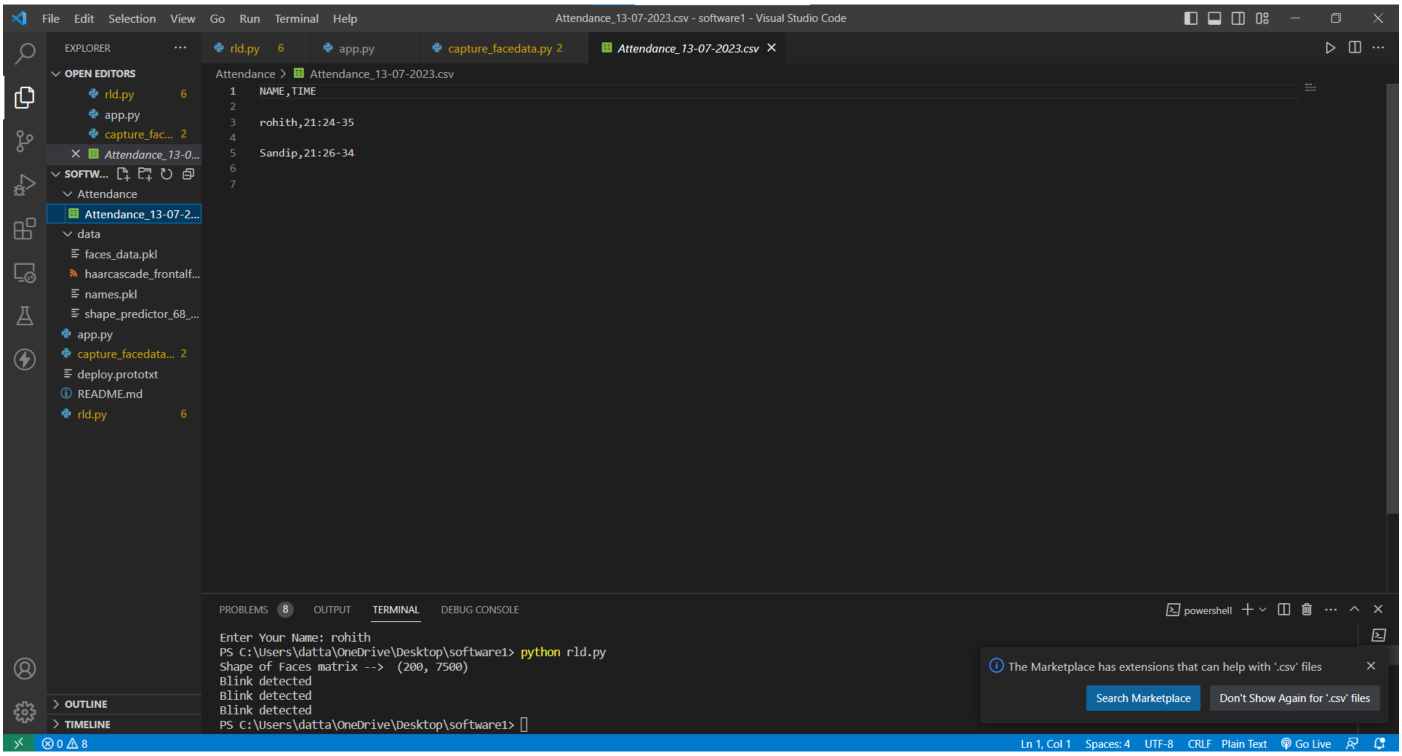
By contrasting the detected blink events with the actual blink events, you may confirm the blink detection's accuracy.

By imitating blink and keypress events, we tested the attendance recording feature.

Checked to see if the recognised name and timestamp from the attendance CSV file are accurately written.

Checked that a fresh CSV file is generated for each date and that the column names and attendance information are correct.

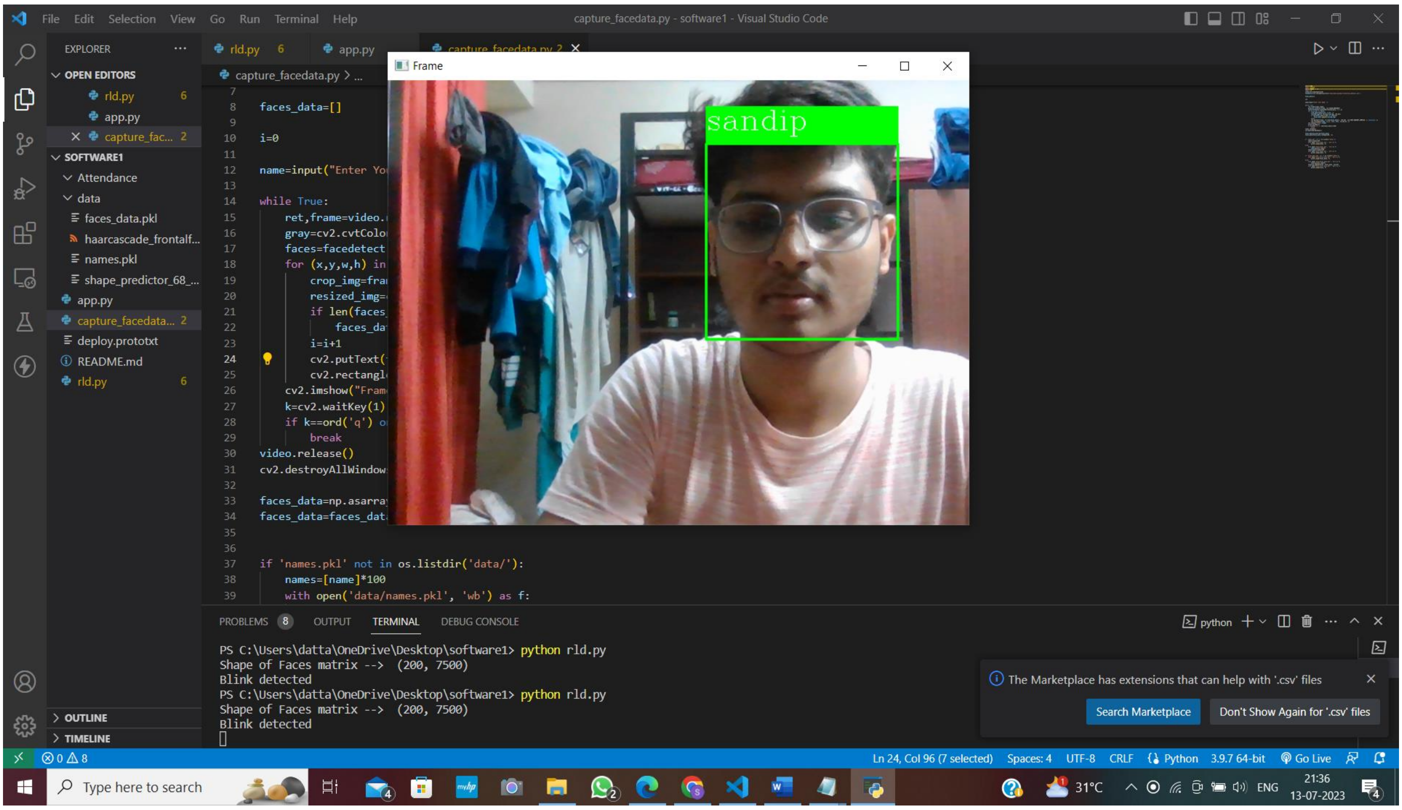
Verifying that the attendance confirmation message is appropriately uttered to validate the voice synthesis functioning.



```
NAME,TIME

rohith,21:24-35

Sandip,21:26-34

```

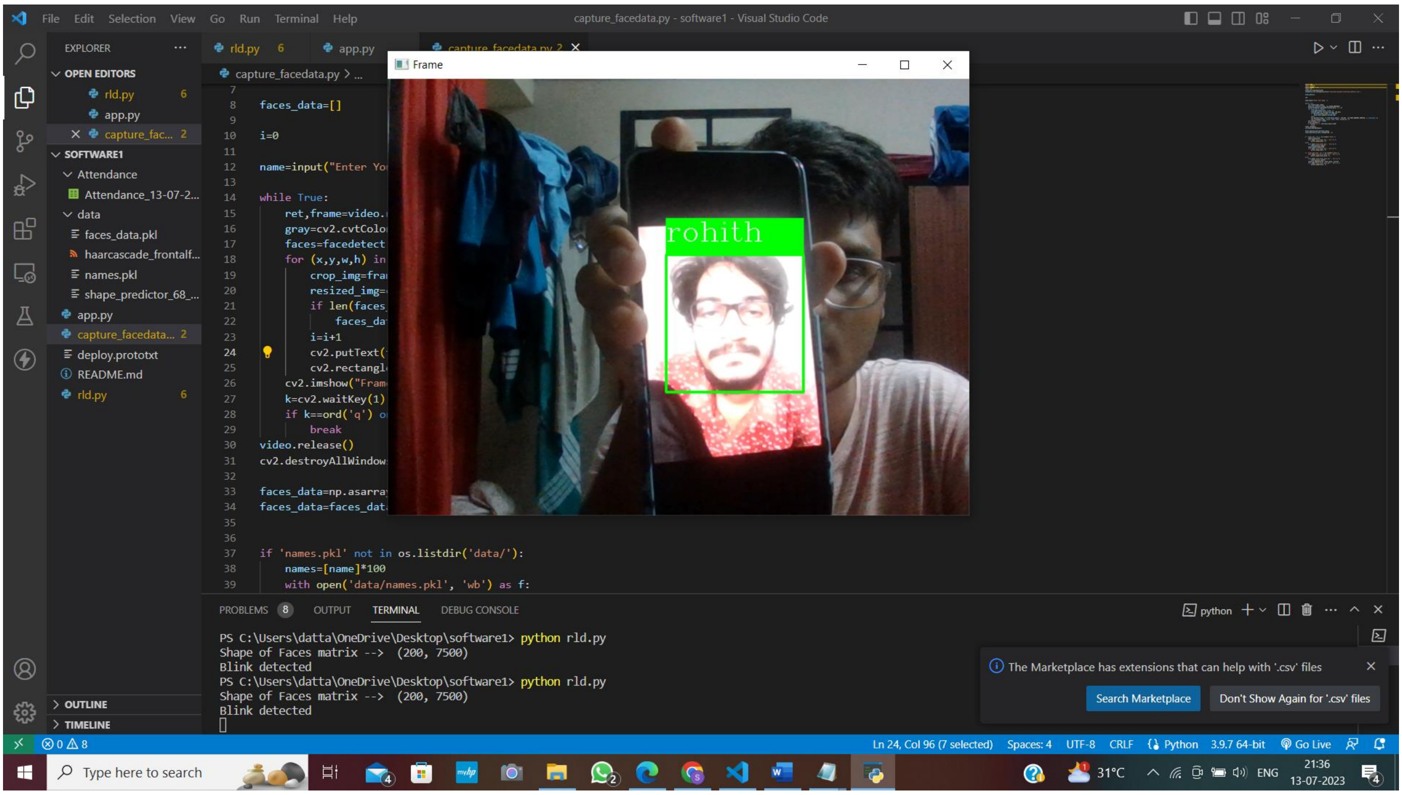
```
Enter Your Name: rohith
PS C:\Users\datta\OneDrive\Desktop\software1> python rld.py
Shape of Faces matrix -->  (200, 7500)
Blink detected
Blink detected
Blink detected
PS C:\Users\datta\OneDrive\Desktop\software1> 
```

If spoofing is there:



```
faces_data=[]

i=0

name=input("Enter Yo

while True:
    ret,frame=video.
    gray=cv2.cvtColo
    faces=facedetect
    for (x,y,w,h) in
        crop_img=fra
        resized_img=
        if len(faces
            faces_da
        i=i+1
        cv2.putText(
        cv2.rectangl
    cv2.imshow("Fram
    k=cv2.waitKey(1)
    if k==ord('q') o
        break
video.release()
cv2.destroyAllWindow


faces_data=np.asarra
faces_data=faces_dat


if 'names.pkl' not in os.listdir('data/'):
    names=[name]*100
    with open('data/names.pkl', 'wb') as f:
```

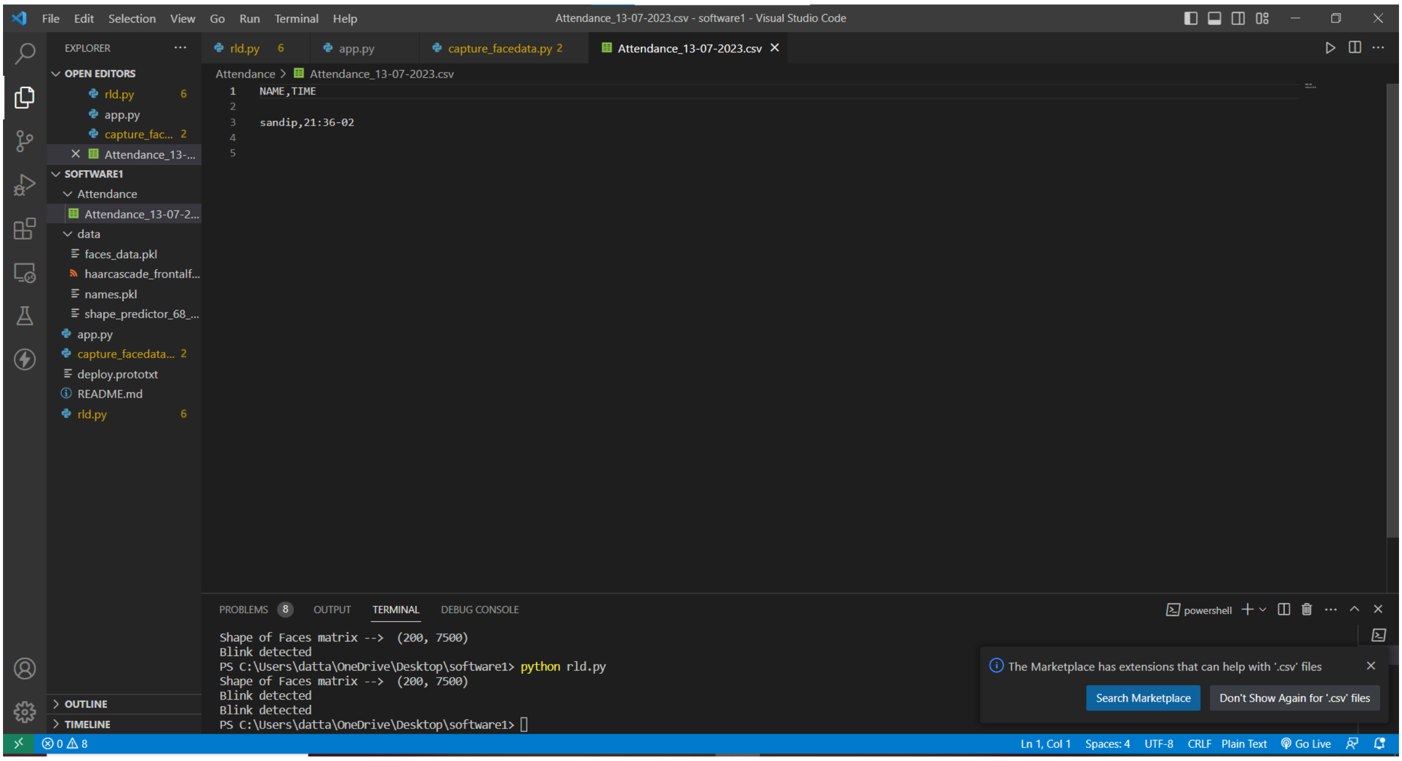
```
PS C:\Users\datta\OneDrive\Desktop\software1> python rld.py
Shape of Faces matrix -->  (200, 7500)
Blink detected
PS C:\Users\datta\OneDrive\Desktop\software1> python rld.py
Shape of Faces matrix -->  (200, 7500)
Blink detected
```



capture_facedata.py > ...

```
faces_data=[]

i=0

name=input("Enter Yo

while True:
    ret,frame=video.
    gray=cv2.cvtColo
    faces=facedetect
    for (x,y,w,h) in
        crop_img=fra
        resized_img=
        if len(faces
            faces_da
        i=i+1
        cv2.putText(
        cv2.rectangl
    cv2.imshow("Fram
    k=cv2.waitKey(1)
    if k==ord('q') o
        break
video.release()
cv2.destroyAllWindow

faces_data=np.asarra
faces_data=faces_dat


if 'names.pkl' not in os.listdir('data/'):
    names=[name]*100
    with open('data/names.pkl', 'wb') as f:
```

```
PS C:\Users\datta\OneDrive\Desktop\software1> python rld.py
Shape of Faces matrix -->  (200, 7500)
Blink detected
PS C:\Users\datta\OneDrive\Desktop\software1> python rld.py
Shape of Faces matrix -->  (200, 7500)
Blink detected
```



Attendance > Attendance_13-07-2023.csv

```
NAME,TIME

sandip,21:36-02
```

```
Shape of Faces matrix -->  (200, 7500)
Blink detected
PS C:\Users\datta\OneDrive\Desktop\software1> python rld.py
Shape of Faces matrix -->  (200, 7500)
Blink detected
Blink detected
PS C:\Users\datta\OneDrive\Desktop\software1>
```

Performance:

Taking attendance for one student after running the code takes approximately 8 seconds

After taking attendance of one student the system takes approximately 5 seconds of time to be ready again to take the next attendance.

Takes very less time after taking the first attendance because it takes some time in the beginning to run the code once it starts running it takes less time for taking the next attendances.

Approximately if the face is 65cm away under good lighting condition from the camera then every face it is recognizing correctly.

Conclusion and Future Work:

Conclusion:

In this project, we used computer vision techniques to create a system for face identification and blink detection. The system records live video from a webcam, recognises faces with the help of the Haar cascade classifier, and then extracts facial attributes. In order to train a K-Nearest Neighbours classifier for face recognition, the collected face photos are kept and utilised. The system uses the dlib library's blink detection feature to track the patterns of eye blinking.

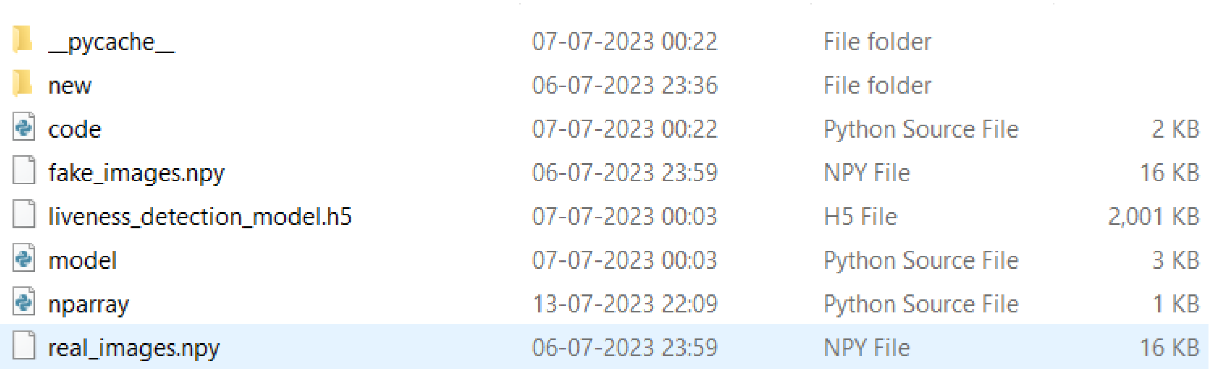
The system constantly takes frames while in operation, identifies faces, and recognises and detects blinks on each identified face. The system keeps track of attendance by saving each person's name and timestamp in a CSV file when their face is recognised and a blink is noticed.

The system uses a number of libraries for image processing, face feature extraction, classification, and data storage, including OpenCV, dlib, numpy, and pickle. Additionally, text-to-speech capabilities is included using the win32com package to provide auditory feedback while collecting attendance.

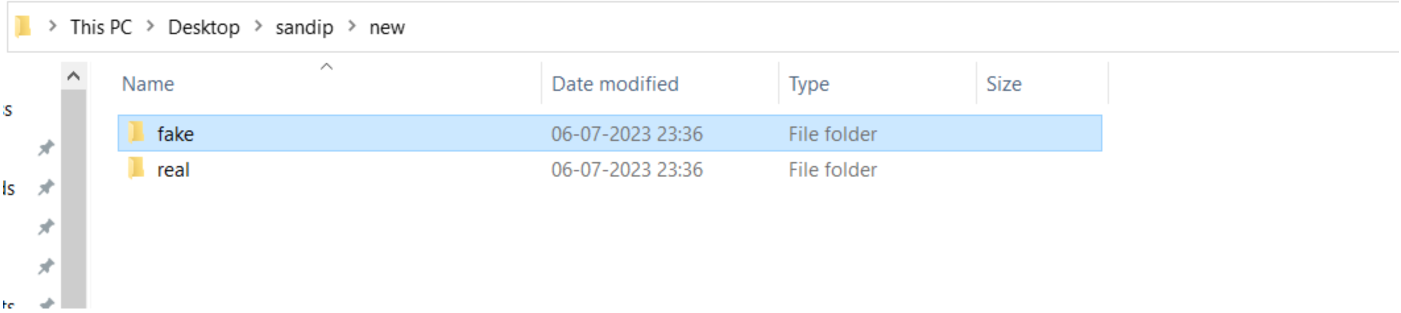
This project offers a potential remedy for automated attendance management systems in diverse situations, such as schools or workplaces, by demonstrating the application of computer vision algorithms for facial identification and blink detection.

Future Plans:

Our priority is removing this blink detection and adding one trained model with the help of tensorflow and keras which will identify real and spoof images, that model needs to be trained on a very large datasets of real and spoofed images then it will achieve higher accuracy, here are some model work

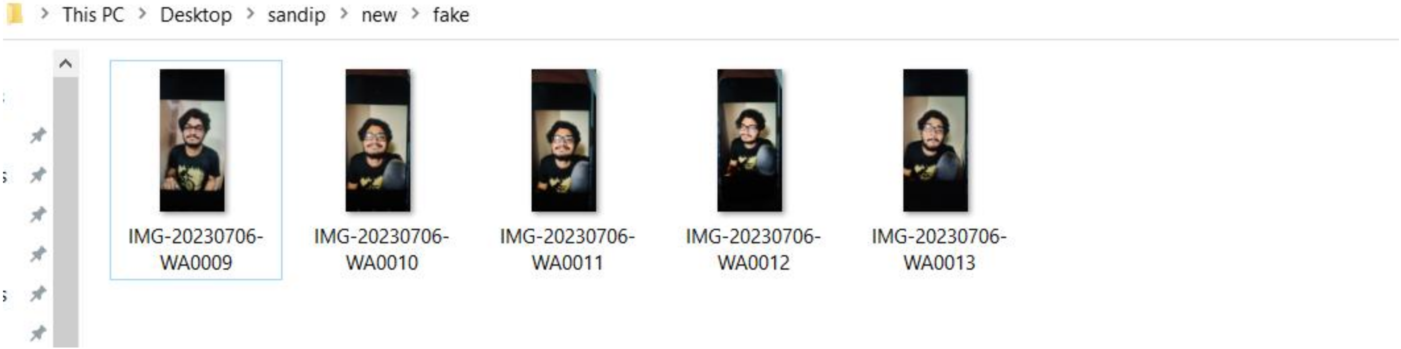


__pycache__	07-07-2023 00:22	File folder	
new	06-07-2023 23:36	File folder	
code	07-07-2023 00:22	Python Source File	2 KB
fake_images.npy	06-07-2023 23:59	NPY File	16 KB
liveness_detection_model.h5	07-07-2023 00:03	H5 File	2,001 KB
model	07-07-2023 00:03	Python Source File	3 KB
nparray	13-07-2023 22:09	Python Source File	1 KB
real_images.npy	06-07-2023 23:59	NPY File	16 KB



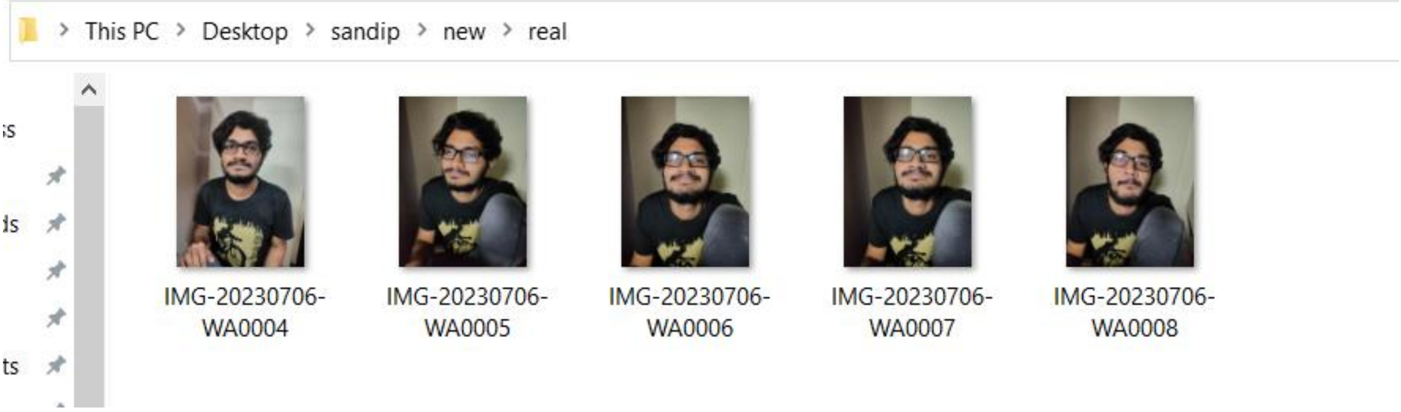
> This PC > Desktop > sandip > new

Name	Date modified	Type	Size
fake	06-07-2023 23:36	File folder	
real	06-07-2023 23:36	File folder	



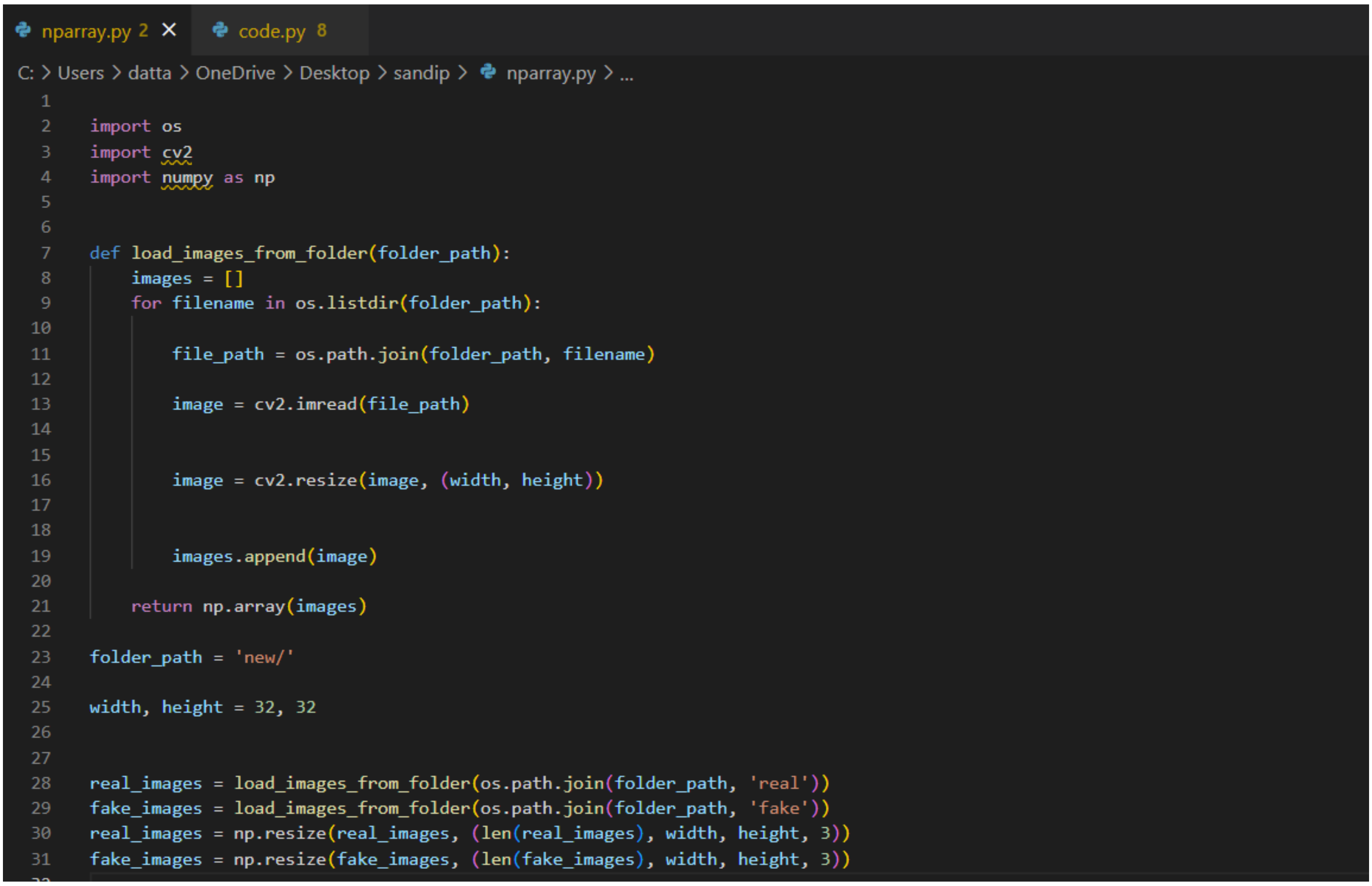
> This PC > Desktop > sandip > new > fake

IMG-20230706-WA0009 IMG-20230706-WA0010 IMG-20230706-WA0011 IMG-20230706-WA0012 IMG-20230706-WA0013

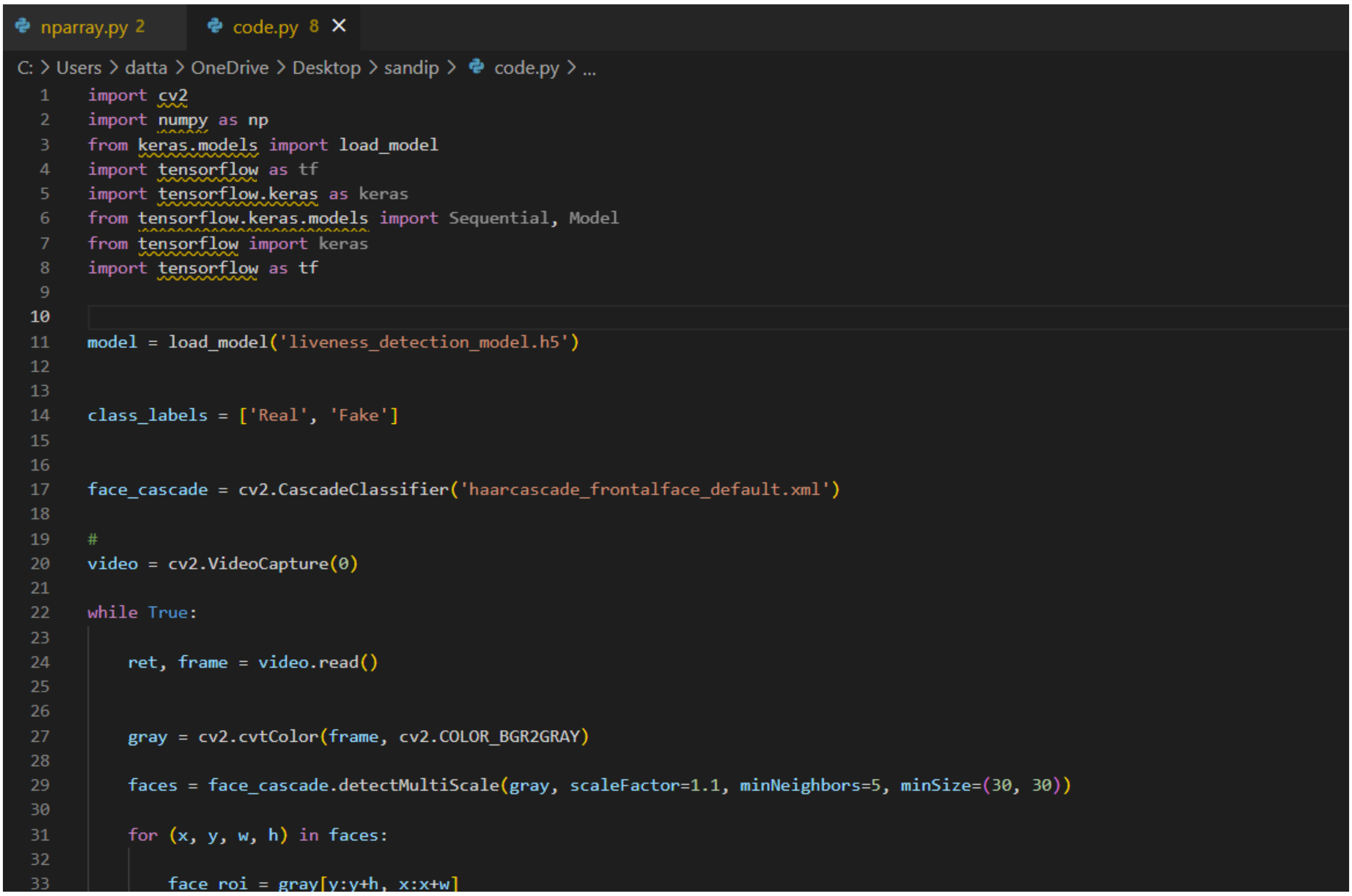


> This PC > Desktop > sandip > new > real

IMG-20230706-WA0004 IMG-20230706-WA0005 IMG-20230706-WA0006 IMG-20230706-WA0007 IMG-20230706-WA0008


nparray.py 2 ×
code.py 8
C: > Users > datta > OneDrive > Desktop > sandip > nparray.py > ...

```python

import os
import cv2
import numpy as np


def load_images_from_folder(folder_path):
    images = []
    for filename in os.listdir(folder_path):

        file_path = os.path.join(folder_path, filename)

        image = cv2.imread(file_path)


        image = cv2.resize(image, (width, height))


        images.append(image)

    return np.array(images)

folder_path = 'new/'

width, height = 32, 32


real_images = load_images_from_folder(os.path.join(folder_path, 'real'))
fake_images = load_images_from_folder(os.path.join(folder_path, 'fake'))
real_images = np.resize(real_images, (len(real_images), width, height, 3))
fake_images = np.resize(fake_images, (len(fake_images), width, height, 3))
```


nparray.py 2
code.py 8 ×
C: > Users > datta > OneDrive > Desktop > sandip > code.py > ...

```python
import cv2
import numpy as np
from keras.models import load_model
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras.models import Sequential, Model
from tensorflow import keras
import tensorflow as tf


model = load_model('liveness_detection_model.h5')


class_labels = ['Real', 'Fake']


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

#
video = cv2.VideoCapture(0)

while True:

    ret, frame = video.read()


    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    for (x, y, w, h) in faces:

        face_roi = gray[y:y+h, x:x+w]
```

After finishing that thing we are planning to develop a desktop application for attendance management within an organization. The application will run in the background while providing a user interface for managing classes and taking attendance. The organization will be able to create multiple classes, save the face data of employees/students for each class, and track attendance using the application.

Technically, the desktop application will be developed using a suitable programming language and framework, such as Python with a GUI library like Tkinter or PyQt. The application will have an intuitive interface where administrators or authorized personnel can create new classes, add employees/students, and save their corresponding face data. The application will utilize the provided code for face detection, recognition, and blink detection to facilitate attendance tracking.

In the background, the application will continuously capture frames from the webcam, detect faces using the Haar cascade classifier, and match them with the stored face data of individuals in each class. When a match is found and a blink is detected, the application will mark the individual as present and record the attendance in a database or file system.

The user interface will provide features for class management, employee/student enrollment, attendance reports, and options for taking attendance. Authorized personnel will have the ability to select a class, view the enrolled individuals, and initiate the attendance-taking process. The application will provide visual feedback on the status of attendance and may also include additional features such as notifications or alerts for missing individuals or anomalies in the attendance pattern.

Overall, the proposed desktop application will automate the attendance management process within an organization, leveraging the existing face recognition and blink detection code. The application will offer an intuitive interface for managing classes and tracking attendance effectively, contributing to streamlined and efficient attendance management practices.