



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and
Engineering J-component Report

Attendance using Face Recognition

*Course Title: Computer Vision
(CSE3089)*

“Department of MTech integrated CSE Specialization in Business Analytics”

By

V Kasinath (20MIA1109)

K Mohit (20MIA1108)

A Vasu (20MIA1151)

Under the supervision of

Dr Bharadwaj

Abstract:

In a variety of contexts, including schools, universities, businesses, and events, keeping track of attendance is an essential duty. Manual roll calls and barcode scanning are two time-consuming, error-prone techniques of taking attendance. Face recognition has become a potential tool for tracking attendance thanks to developments in computer vision and deep learning. We outline a project that uses face recognition to track attendance in this abstract.

Our research seeks to create a precise, effective, and practical face recognition-based attendance tracking system. To identify faces in live photos or video streams, the system makes use of a deep learning model that has already been trained. The recorded faces in a real-time database are then matched with the recognized faces to update the attendance status. The system also features an intuitive interface for interacting with the system, managing the database, and producing attendance reports by authorized individuals.

Our facial recognition-based attendance tracking system's real-time capability, accuracy, and ease are its main advantages. By doing away with manual roll calls and barcode scanning, the technology saves time and minimizes mistakes. It can distinguish faces clearly even in situations with changing lighting, angles, and facial expressions. The real-time database update guarantees prompt updates to the attendance status, enabling real-time attendance tracking. Authorized workers can easily interact with the system and generate attendance records thanks to the user-friendly interface.

Our face recognition-based attendance tracking system has a lot of promise in a variety of environments, including companies, events, colleges, and schools. It could increase effectiveness and precision.

Keywords: Attendance, Data Base, Deep Learning, Accuracy, Efficiency,

User-Friendly Interface, Attendance Tracking, Image Processing, and Facial features.

Introduction:

The aim of this project is to create a facial recognition-based attendance system. To record attendance, the system will take a picture of a student's or employee's face and compare it to a database of faces that have been registered. By doing away with manual attendance taking, the system will save time and minimize errors.

A system called facial recognition attendance uses face recognition technology to record and authenticate people's identities for attendance purposes. This technology employs cameras and software to identify a person's face with a pre-registered image in a database to record attendance in place of more conventional techniques like paper or digital sign-in sheets.

The procedure entails taking a picture of someone's face, examining it, and then creating a special template of their facial traits. To identify and confirm the person's identity, this template is then compared to a database of previously registered faces. Their attendance is automatically recorded when their identity has been verified, minimizing the need for manual entry and lowering the possibility of mistakes.

Due to this system's precision, effectiveness, and ease, it has become more and more popular in recent years. Where attendance tracking is crucial, such as in businesses and other organizations, it is extensively utilized. However, there are ongoing discussions over the morality of using facial recognition technology in attendance systems, and privacy and security concerns have been raised.

Related Works:

"Automated Attendance Management System using Face Recognition" by V. P. Patil and V. K. Patil, published in the International Journal of Computer Applications in 2015.

"Face Recognition based Attendance System using Principal Component Analysis" by S. S. Kulkarni and S. B. Patil, published in the International Journal of Computer Applications in 2016.

"Face Recognition based Attendance Management System using Raspberry Pi" by S. M. Khapre and S. V. Kakde, published in the International Journal of Engineering Research and Technology in 2019.

"Facial Recognition System for Attendance Management" by R. R. Kadam and P. R. Garud, published in the International Journal of Advanced Research in Computer Science and Software Engineering in 2017.

"Smart Attendance System using Face Recognition" by A. R. Patil and S. B. Patil, published in the International Journal of Engineering and Advanced Technology in 2018.

These works describe various approaches to using face recognition for attendance management, such as using PCA for face recognition, implementing the system on a Raspberry Pi, and developing a smart attendance system.

About the Dataset:

Real-Time data will be Taken and it will get trained and will be saved in the Firebase database.

Methodology:

In order to mark attendance, we follow a series of steps which includes enrolment, face detection, face recognition, and then marking the attendance in a database. Unlike Eigenfaces and Fisher faces, where in most modern face verification systems, training and enrolment are two different steps. Training is performed on millions of images. On the other hand, enrolment is performed using a small set of images. In the case of Dlib, enrolling a person is simply passing a few images of the person through the network to obtain 128- dimensional feature descriptors corresponding to each image. In other words, we convert each image to a feature in a high-dimensional space. In this high-dimensional space, features belonging to the same person will be close to each other and far away for different persons.

A. Comparison of the Dlib Face Recognition Model with the Conventional Image Classification Pipeline

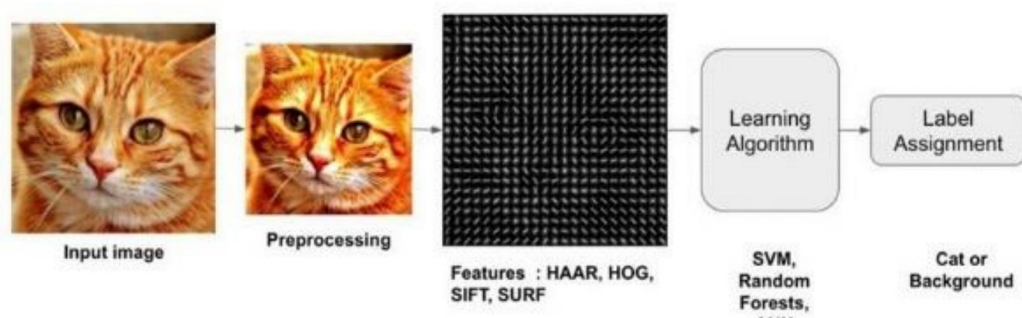
Traditionally, a pipeline for image classification converts the picture into a point (or feature vector, in higher-dimensional space), or vice versa.

This was accomplished by computing the feature descriptor for an image patch (for example, HOG). After the image has been represented as a point in higher dimensional space, we divide the space using hyperplanes to separate the points that correspond to the various classes using a learning technique like SVM.

Deep Learning and the model above have conceptual parallels despite the fact that Deep Learning appears extremely different on the surface.

The following rules define when a function involving two vectors can be called a metric. A mapping $d(x,y)$ is called a metric if

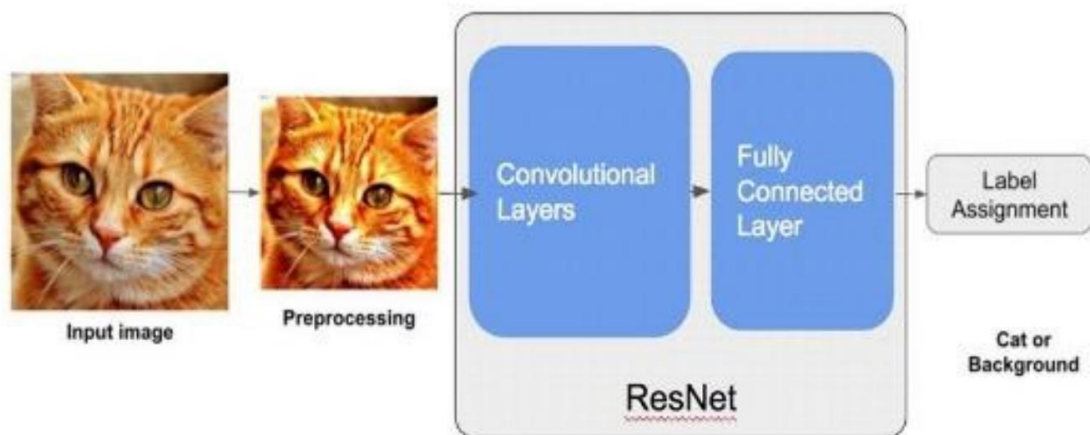
1. The distance between any two points is greater than or equal to zero $d(x,y) \geq 0$
2. A point has zero distance from itself i.e., $d(x,x)=0$
3. The distance from x to y i.e., the same as the distance from y to x i.e., $d(x,y)=d(y,x)$
4. Triangle inequality: For any three points x , y and z the following inequality holds true. i.e.,



$$d(x,y) + d(y,z) \geq d(z,x)$$

This was accomplished by computing the feature descriptor for an image patch (for example, HOG). After the image has been represented as a point in higher dimensional space, we divide the space using hyperplanes to separate the points that correspond to

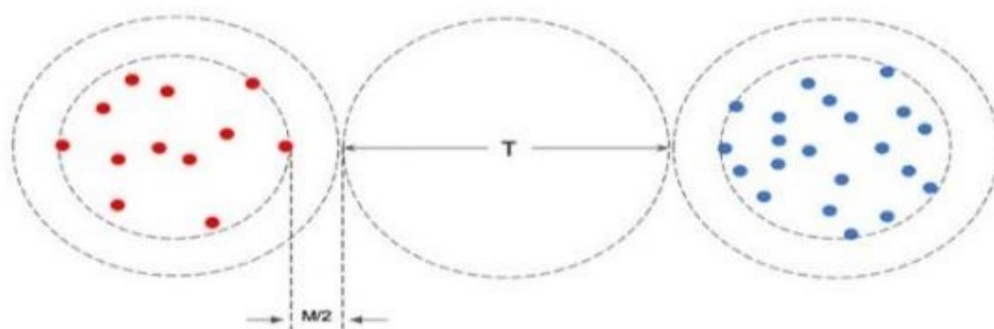
the various classes using a learning technique like SVM. Deep Learning and the model above have conceptual parallels despite the fact that Deep Learning appears extremely different on the surface.



To put it another way, you enter an image, and the result is a point in 128-dimensional space. Simply run both photos through CNN to get the two points in this 128-dimensional space and determine how closely two images are related. Using the straightforward L2 (Euclidean) distance between the two points, you may compare them.

Metric Loss

A production-ready CNN is often trained using millions of images. It is obvious that CNN's dials cannot be updated at the same time as these millions of photographs. A single tiny batch of photos is used for training at a time. A micro-batch is the name for this small quantity.



Metric loss defined by Dlib's Face Recogniser

B. Facial Recognition And Detection

The distance between the enrolled faces and the new face in the 128-dimensional space can be used to determine whether a fresh photograph of a person is the same person. Read the descriptors and name-label mapping from the disc. The query image, which is a picture of a classroom with several pupils, is then read, and it is converted from BGR to RGB format. since RGB is the default format for Dlib. Next, convert the OpenCV RGB image to the Dlib cv image format, and finally the Dlib cv image format to the Dlib matrix format. The neural network module does not recognize the cv image format used by Dlib. In the image in question, find faces.

Recognize facial landmarks for each face. Obtain 150x150-inch warped patches for each face. Make a face descriptor for each face now. The Euclidean distance between the face descriptors in the query photos and the face descriptors in the enrolled images is now calculated. Locate the enrolling face where the distance is the shortest.

According to Dlib, two face descriptor vectors are generally considered to be from the same person if their Euclidean distance is less than 0.6; otherwise, they are considered to be from distinct people. This threshold will change depending on the quantity of photographs enrolled and other differences (camera quality, lighting) between enrolled images and query image. We are employing a 0.5 threshold. Get the person's name from the index if the minimum distance is less than the threshold; otherwise, the individual in the search image is unknown.

C. Marking Attendance

The database records the attendance for the associated Period for each face that is detected and matches an enrolled face. the student's name, the date and hour of the person at live, and **the database will keep track of attendance.**

Algorithms used:

Convolutional Neural Networks (CNN): CNN is a deep learning algorithm that is used to train the system to recognize faces. It is particularly useful in face recognition because it can learn features from images automatically.

Eigenface Algorithm: This algorithm is used to represent faces in a high-dimensional space. It creates a set of "eigenfaces" that represent different facial features and then matches the detected face to the closest eigenface.

Fisherface Algorithm: This algorithm is similar to the eigenface algorithm, but it takes into account the variation in the faces in the database. It creates a set of "fisher faces" that represent the features that are most important for distinguishing between different faces.

Implementation:

1. ADDING DATA TO THE DATABASE

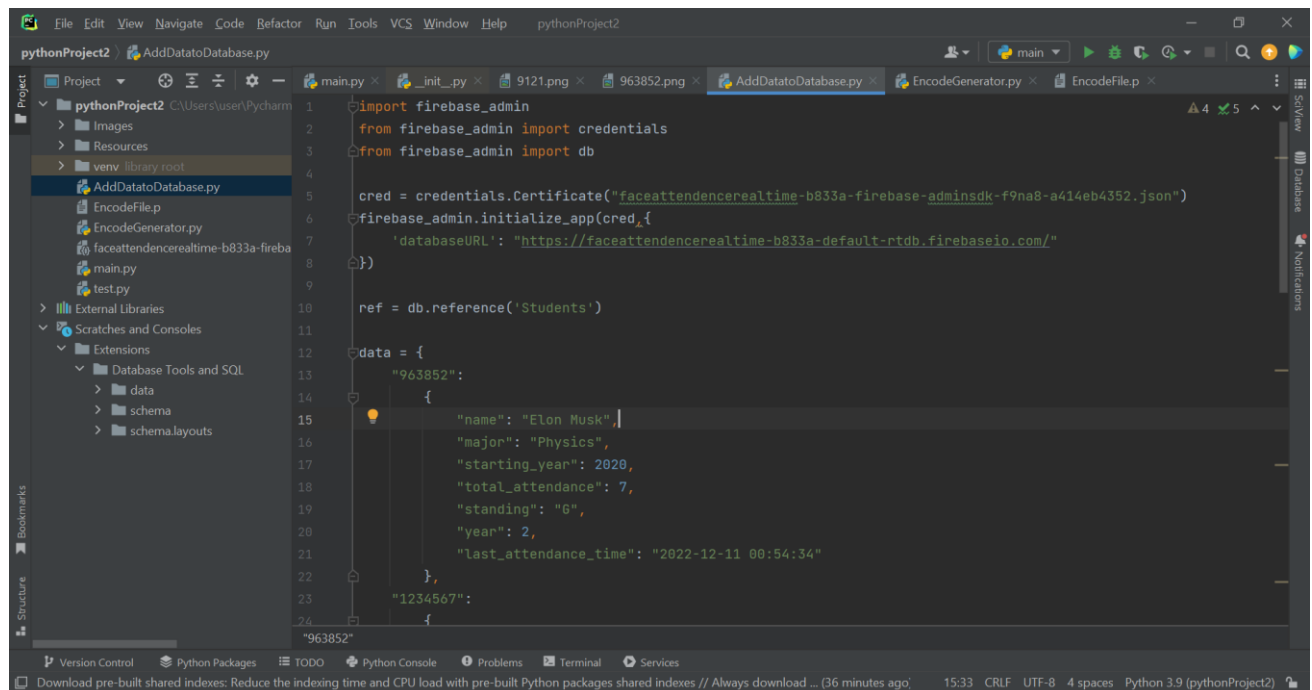
We have used PyCharm for implementing and running the code.

The code is an example of how to use the Firebase Realtime Database API to write data to a database. It first imports the necessary modules from the `firebase_admin` package, and initializes the app with the provided credentials and URL of the database.

Then, we define a Python dictionary variable `data` that represents a set of student records. Each student record is represented as a nested dictionary with keys such as `name`, `major`, `year`, etc.

The for loop iterates through each record in the `data` dictionary and sets the corresponding value as a child node of the "Students" node in the Firebase Realtime Database. The `set()` method is used to write the data to the database.

Overall, this code writes the student records to the Firebase Realtime Database under the "Students" node in a JSON-like format.



```
1 import firebase_admin
2 from firebase_admin import credentials
3 from firebase_admin import db
4
5 cred = credentials.Certificate("faceattendencerealttime-b833a-firebase-adminsdk-f9na8-a414eb4352.json")
6 firebase_admin.initialize_app(cred, {
7     'databaseURL': "https://faceattendencerealttime-b833a-default-rtdb.firebaseio.com/"
8 })
9
10 ref = db.reference('Students')
11
12 data = {
13     "963852":
14         {
15             "name": "Elon Musk",
16             "major": "Physics",
17             "starting_year": 2020,
18             "total_attendance": 7,
19             "standing": "6",
20             "year": 2,
21             "last_attendance_time": "2022-12-11 00:54:34"
22         },
23     "1234567":
24         {
```

2. ENCODEGENERATOR

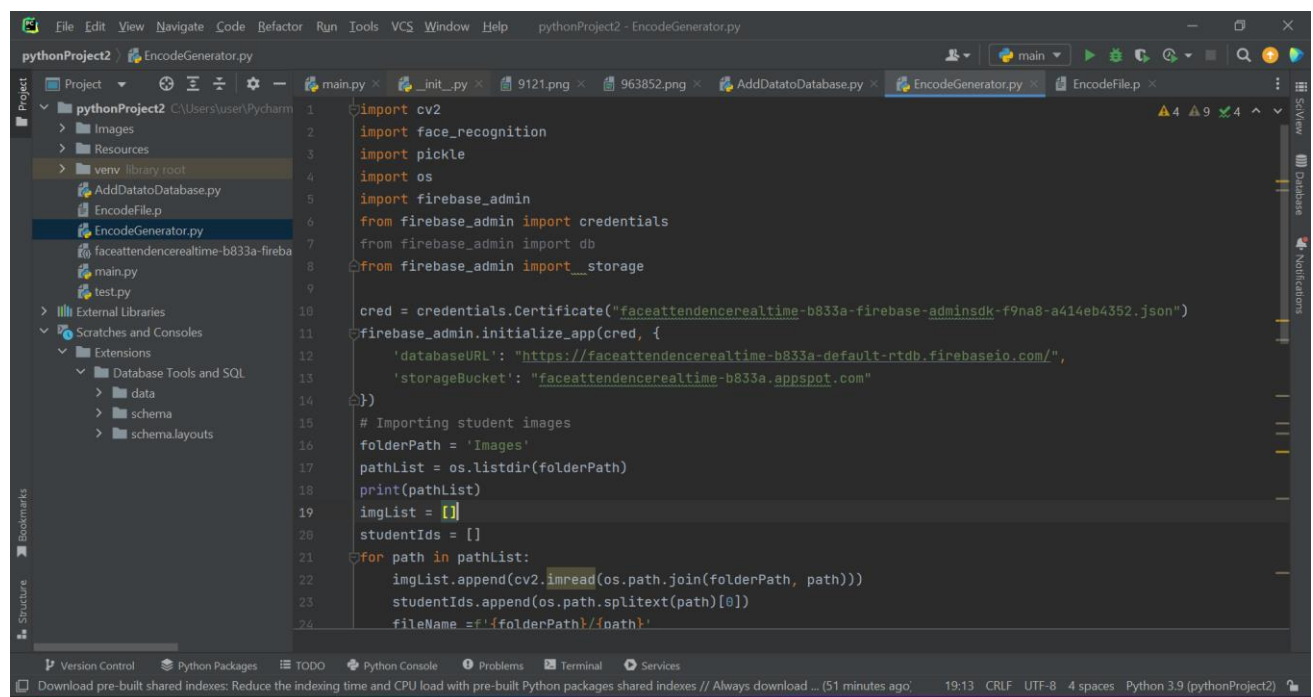
This Python code performs facial recognition by encoding images of students and storing them in a Firebase database. Here's a brief overview of what the code does:

First, the necessary libraries are imported, including `OpenCV` for image processing and `face_recognition` for facial recognition. The Firebase libraries are also imported and initialized using a service account credential file.

The code then reads images of students from a folder and creates a list of images and corresponding student IDs. Each image is uploaded to a Firebase storage bucket using the Google Cloud Storage API.

Next, the function findEncodings is defined to encode each image in the list of images using the face_recognition library. The resulting encodings are stored in a separate list. The code then saves the list of encodings and student IDs to a file using the Python pickle library.

Overall, this code performs facial recognition on a set of student images, stores the resulting encodings and student IDs in a file, and uploads the images to a Firebase storage bucket.



```
1 import cv2
2 import face_recognition
3 import pickle
4 import os
5 import firebase_admin
6 from firebase_admin import credentials
7 from firebase_admin import db
8 from firebase_admin import storage
9
10 cred = credentials.Certificate("faceattendencerealttime-b833a-firebase-adminsdk-f9na8-a414eb4352.json")
11 firebase_admin.initialize_app(cred, {
12     'databaseURL': "https://faceattendencerealttime-b833a-default-rtadb.firebaseio.com/",
13     'storageBucket': "faceattendencerealttime-b833a.appspot.com"
14 })
15
16 # Importing student images
17 folderPath = 'Images'
18 pathList = os.listdir(folderPath)
19 print(pathList)
20 imgList = []
21 studentIds = []
22
23 for path in pathList:
24     imgList.append(cv2.imread(os.path.join(folderPath, path)))
25     studentIds.append(os.path.splitext(path)[0])
26     fileName = f'{folderPath}/{path}'
```

3. MAIN.PY

This code appears to be a real-time face recognition system that uses a webcam to detect faces and recognize them by comparing them to a pre-existing database of facial encodings. The recognized faces are then used to update attendance data for students in a Firebase Realtime Database.

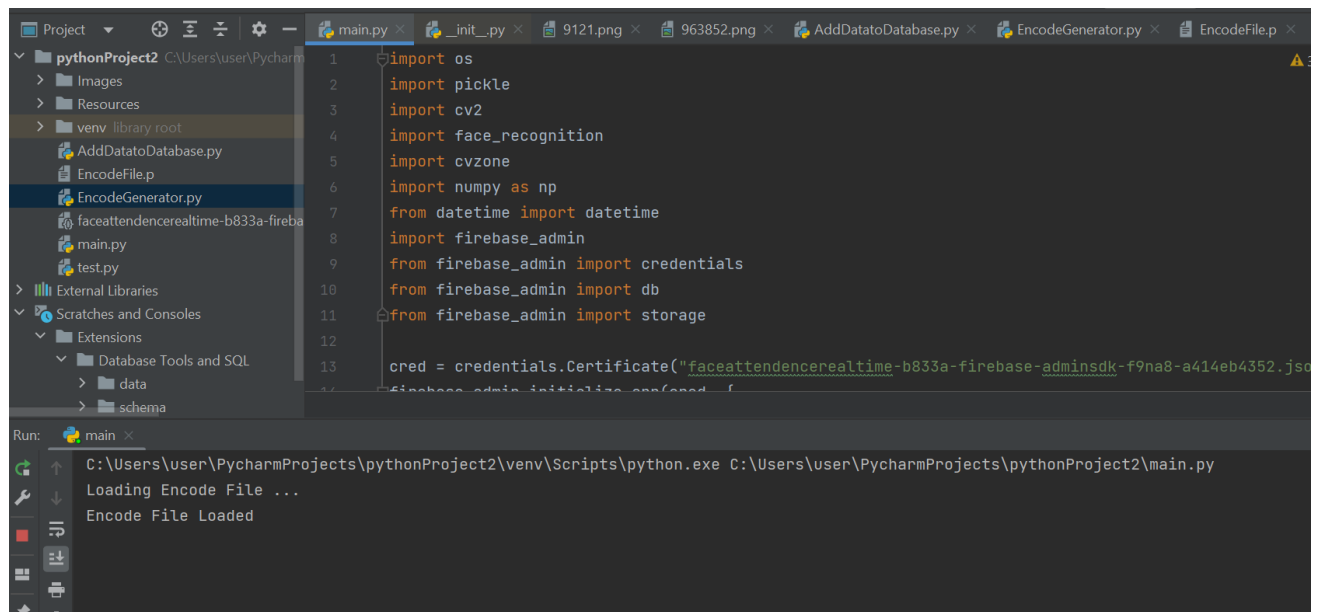
The system works by first initializing Firebase and loading the necessary images and encoding files. It then captures frames from the webcam and detects faces in each frame using the face_recognition library. If a recognized face is detected, the system updates the student's attendance data in the Firebase Realtime Database and displays relevant information on the screen.

The system also uses various modes and images to display different information on the screen depending on the state of the system.

For example, when a face is detected, the system displays a loading screen before showing the relevant student's information. When the attendance data has been updated, the system displays a success screen before resetting to its initial state.

Overall, this code is an example of how face recognition can be used for real-world applications such as attendance tracking. However, it is important to consider privacy concerns and ensure that appropriate measures are taken to protect the data and privacy of the individuals being tracked.

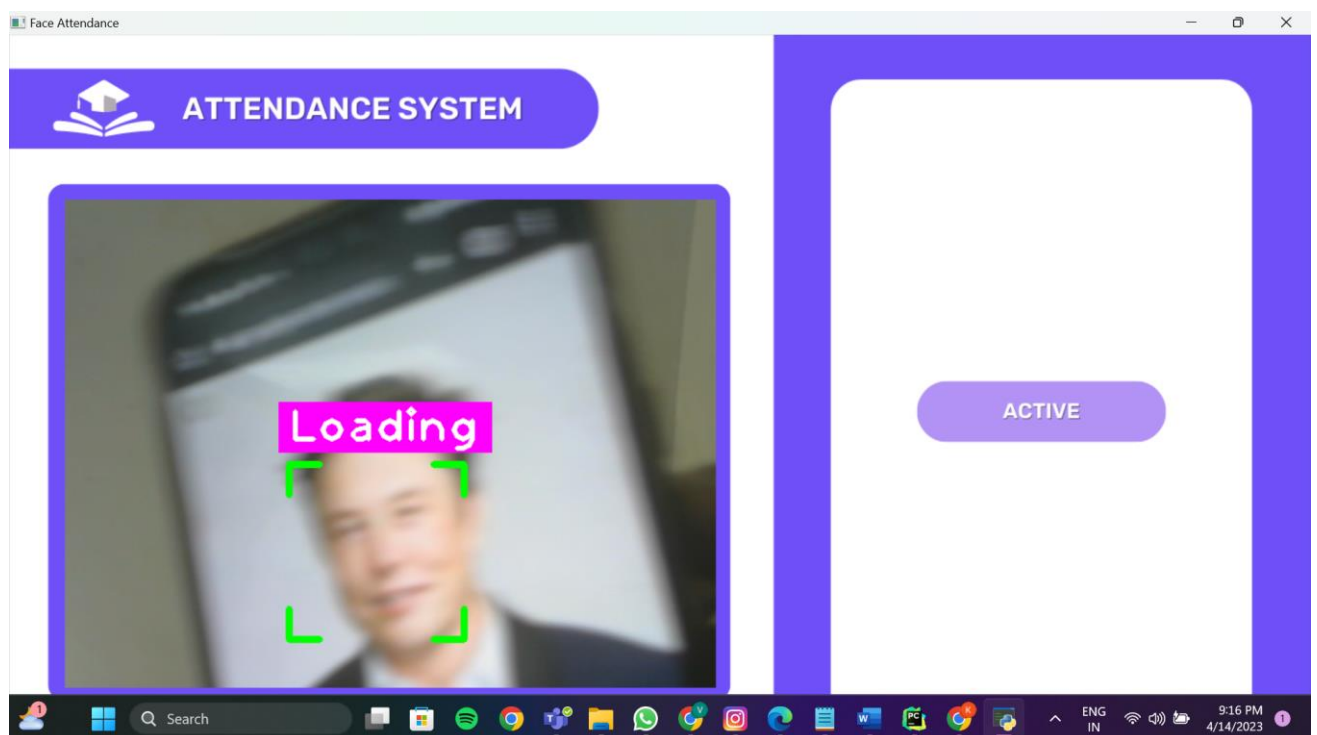
OUTPUTS: -



```
1 import os
2 import pickle
3 import cv2
4 import face_recognition
5 import cvzone
6 import numpy as np
7 from datetime import datetime
8 import firebase_admin
9 from firebase_admin import credentials
10 from firebase_admin import db
11 from firebase_admin import storage
12
13 cred = credentials.Certificate("faceattendencerealttime-b833a-firebase-adminsdk-f9na8-a414eb4352.json")
14 firebase_admin.initialize_app(cred, {
```

Run: main x

C:\Users\user\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:\Users\user\PycharmProjects\pythonProject2\main.py
Loading Encode File ...
Encode File Loaded



FaceAttendanceRealTime - RealTime x +

console.firebase.google.com/project/faceattendencerealttime-b833a/database/faceattendencerealttime-b833a-default-rtdb/data

FaceAttendanceRealTime

Realtime Database

Data Rules Backups Usage Extensions NEW

https://faceattendencerealttime-b833a-default-rtdb.firebaseio.com

```
https://faceattendencerealttime-b833a-default-rtdb.firebaseio.com/  
└── Students  
    ├── 9121  
    ├── 913244  
    ├── 963852  
    └── 1234567
```

Database location: United States (us-central1)

Windows taskbar: Search, File Explorer, Edge, Chrome, Teams, Outlook, WhatsApp, Zoom, Instagram, OneDrive, Word, PowerPoint, VLC, Spotify, System tray: ENG IN, 9:17 PM 4/14/2023

FaceAttendanceRealTime - RealTime x +

console.firebase.google.com/project/faceattendencerealttime-b833a/database/faceattendencerealttime-b833a-default-rtdb/data

FaceAttendanceRealTime

Realtime Database

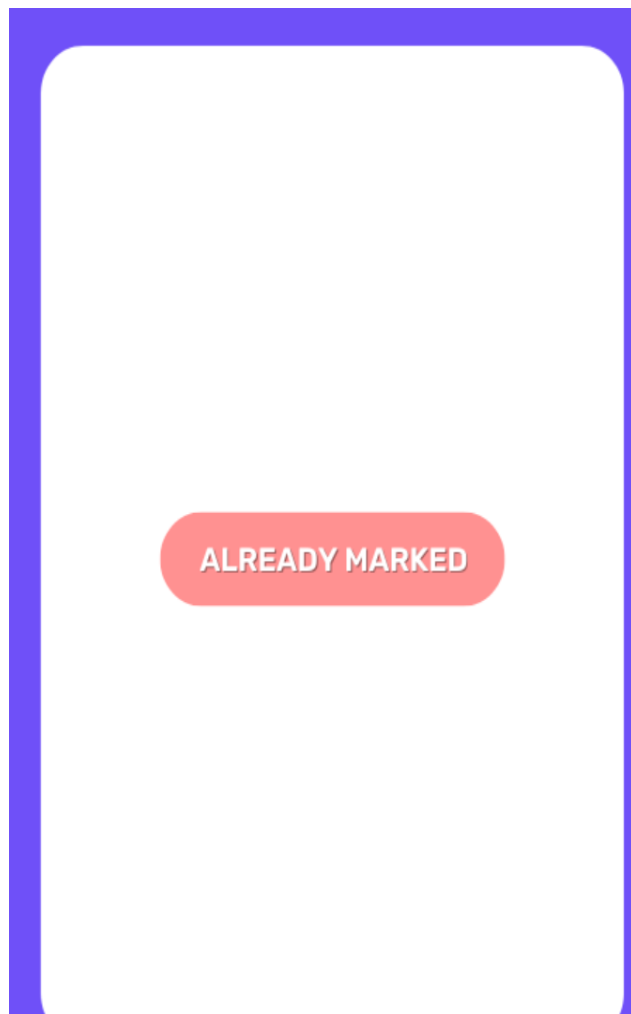
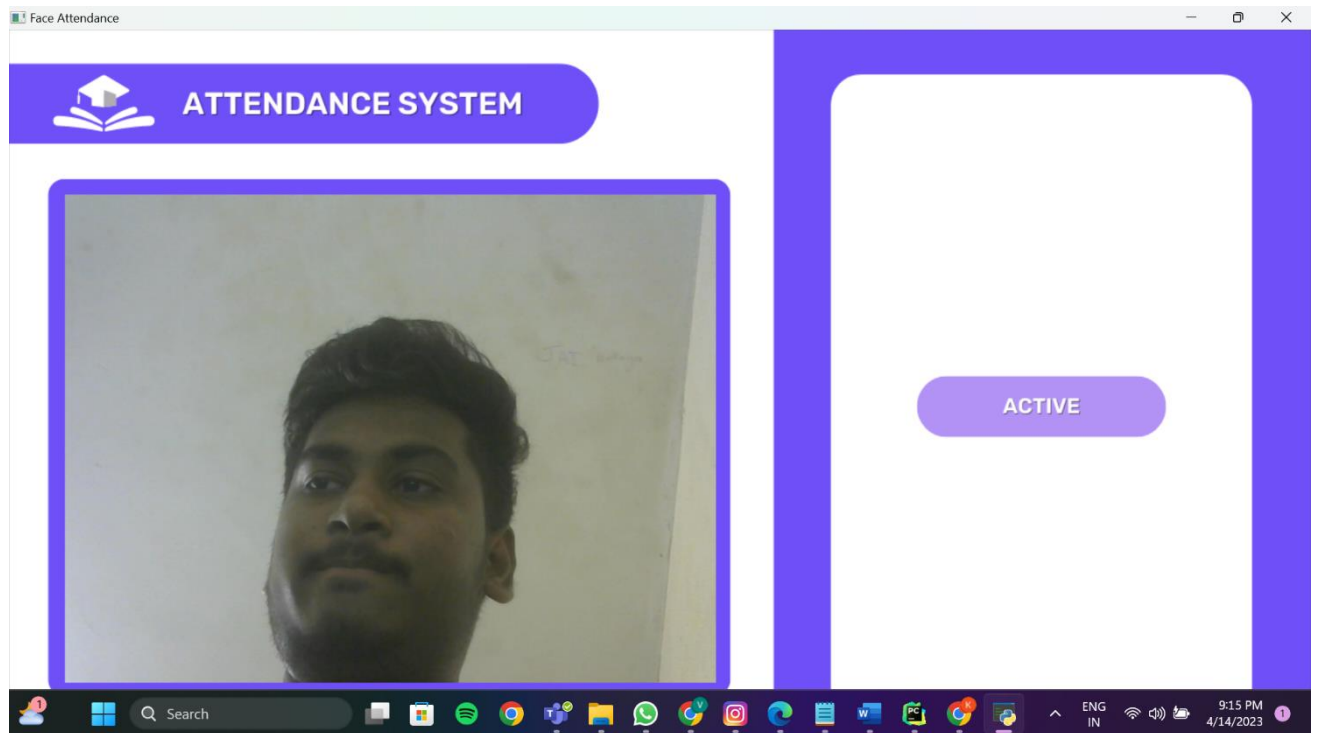
Data Rules Backups Usage Extensions NEW

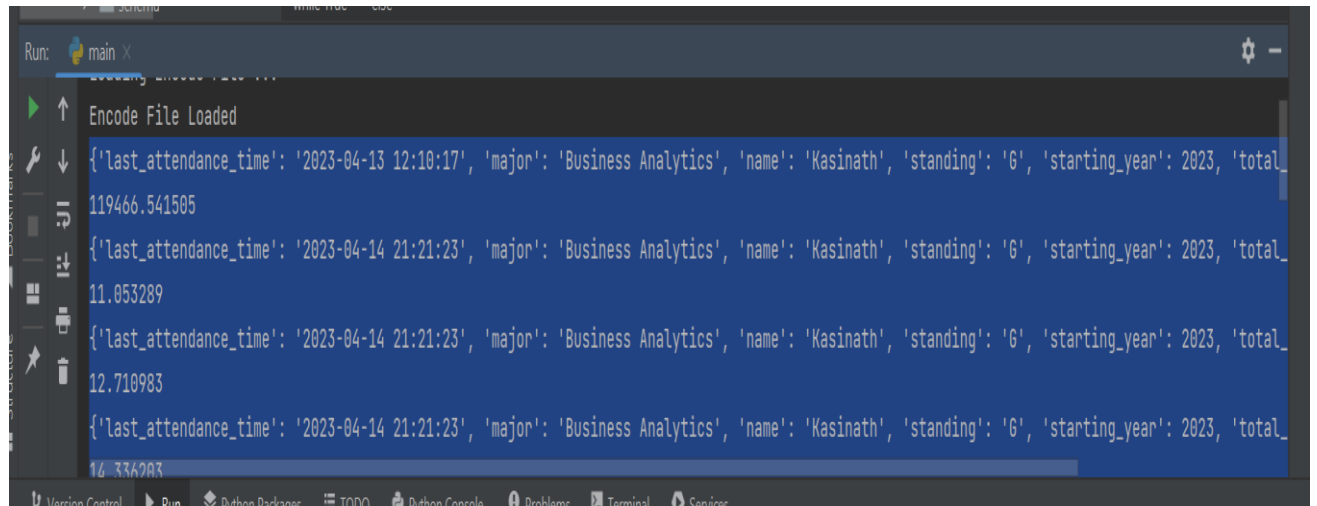
https://faceattendencerealttime-b833a-default-rtdb.firebaseio.com

```
└── 963852  
    ├── last_attendance_time: 2023-14-4 00:54:34 ABC X  
    ├── major: "Physics"  
    ├── name: "Elon Musk"  
    ├── standing: "G"  
    ├── starting_year: 2020  
    ├── total_attendance: 7  
    └── year: 2
```

Database location: United States (us-central1)

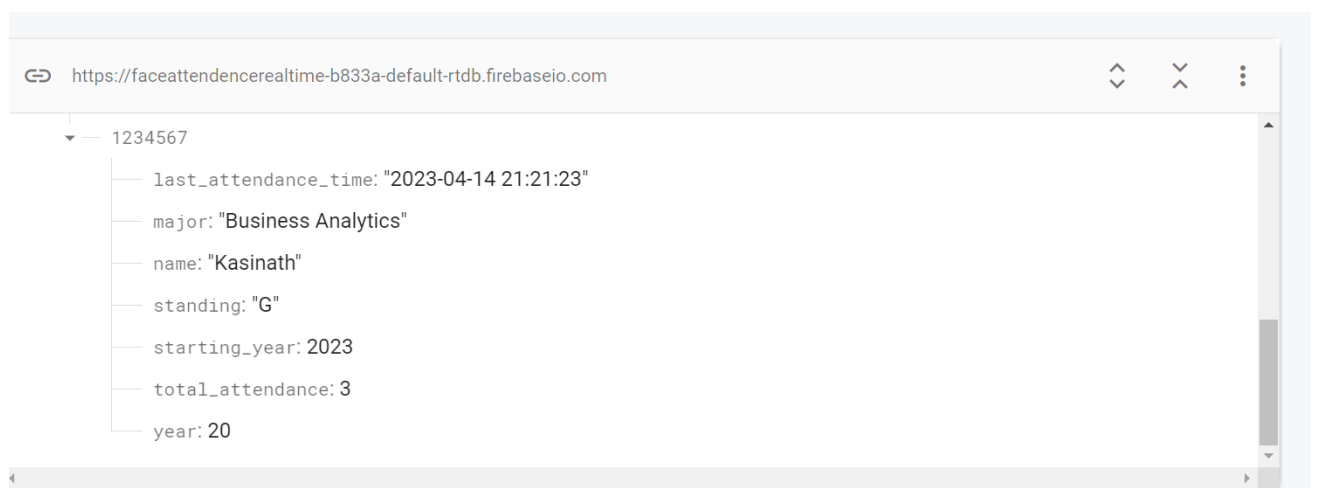
Windows taskbar: Search, File Explorer, Edge, Chrome, Teams, Outlook, WhatsApp, Zoom, Instagram, OneDrive, Word, PowerPoint, VLC, Spotify, System tray: ENG IN, 9:18 PM 4/14/2023





The screenshot shows a terminal window with a dark background. At the top, it says "Run: main x". Below that, there's a message "Encode File Loaded". The main part of the terminal displays several lines of JSON data, each representing an attendance record. The records include fields like 'last_attendance_time', 'major', 'name', 'standing', 'starting_year', and 'total_'. The data is as follows:

```
{ 'last_attendance_time': '2023-04-13 12:10:17', 'major': 'Business Analytics', 'name': 'Kasinath', 'standing': 'G', 'starting_year': 2023, 'total_': 119466.541505 }
{ 'last_attendance_time': '2023-04-14 21:21:23', 'major': 'Business Analytics', 'name': 'Kasinath', 'standing': 'G', 'starting_year': 2023, 'total_': 11.053289 }
{ 'last_attendance_time': '2023-04-14 21:21:23', 'major': 'Business Analytics', 'name': 'Kasinath', 'standing': 'G', 'starting_year': 2023, 'total_': 12.710983 }
{ 'last_attendance_time': '2023-04-14 21:21:23', 'major': 'Business Analytics', 'name': 'Kasinath', 'standing': 'G', 'starting_year': 2023, 'total_': 16.336203 }
```



The screenshot shows a web browser window with the address bar displaying "https://faceattendencerealtime-b833a-default-rtadb.firebaseio.com". The main content area shows a JSON object with the following fields:

```
{
  last_attendance_time: "2023-04-14 21:21:23",
  major: "Business Analytics",
  name: "Kasinath",
  standing: "G",
  starting_year: 2023,
  total_attendance: 3,
  year: 20
}
```

As we see here the attendance will be captured in real-time and it will be saved in the cloud database, we can monitor the data anytime anywhere

CONCLUSION:

The procedure mentioned above guarantees the best results. This is accomplished by utilizing dlib for facial recognition and OpenCV for frame extraction. This technique will require less time to recognize several faces from a single frame and will be more accurate.

REFERENCES:

<https://www.computervision.zone/courses/face-recognition-with-real-time-database/>

https://www.researchgate.net/publication/348733119_Facial_Recognition_Based_Attendance_System_using_OpenCV_and_Python

<https://www.sciencedirect.com/science/article/pii/S2666827021000135>