International Institute of Information Technology Bhubaneswar

Face Recognition Attendance System

(Report)

Prepared By

- B420004 - Akash Parida
- B420037 - Priyanshu
- B420043 - Sailesh Agarwal

Guided by: Prof. Sarthak Padhi

April 16th, 2023

TABLE OF CONTENTS

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

1. Introduction

The "Face Recognition Attendance System" is a Python-based application that enables effective and precise attendance management using face recognition technology. The system can be used to automate the attendance management process in a variety of settings, including schools, universities, and workplaces. This reduces the workload on staff and gets rid of the mistakes associated with conventional methods.

The system uses deep learning models and advanced image and video processing techniques to recognize faces accurately. It uses the "Face Recognition library," which is based on dlib's face recognition "deep learning" models and offers developers a straightforward interface to use the models for different face recognition tasks.

The Face Recognition Attendance System also includes features such as real-time face recognition, automatic attendance management, and face detection and tracking. It can process video streams from a camera in real-time to detect and recognize faces, and it can automatically update the attendance records based on the recognized faces.

2. Utility of The Project

The Face Recognition Attendance System is an innovative solution that utilizes image and video processing techniques and deep learning models for efficient and accurate attendance management using face recognition technology. The system provides a number of benefits, including:

- **Automation:** The system automates the attendance management process, reducing the workload on staff and eliminating the errors associated with traditional methods.
- **Efficiency:** The system can quickly and accurately recognize faces, making attendance management faster and more efficient.
- **Accuracy:** The system uses advanced face recognition algorithms and deep learning models, resulting in highly accurate attendance records.
- **Security:** The system can be used to enhance the security of educational institutions and workplaces by accurately tracking attendance and detecting unauthorized entry.
- **Flexibility**: The system is highly customizable and can be configured to meet the specific needs of different settings and institutions.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

3. Proposed Workflow in the Project

The proposed workflow in the project is as shown in the flowchart below:

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

4. Framework(s) and Tool(s) Used

The Face Recognition Attendance System uses a combination of pre-trained algorithms and deep neural networks to detect and recognize faces in real-time, and a realtime database management system to manage attendance records and provide a simple user interface.

The modules and algorithms used in the Face Recognition Attendance System repository are:

• face_recognition module:

The face_recognition module is used in the Face Recognition Attendance System to perform the facial recognition task. This module is built on top of the dlib, numpy, and scipy libraries. It provides a simple and easy-to-use interface for face detection, face recognition, and face manipulation.

In this project, the face_recognition module is used to recognize the faces of the students and staff members during the attendance process. The module uses deep learning algorithms to create a facial encoding for each face. This encoding is a 128-dimension vector that represents the unique features of each face. The module then compares these encodings to recognize the faces.

The face_recognition module provides the following functions:

1. face_recognition.face_locations(image, model='hog'): This function detects the location of faces in an image using either the hog or cnn model.

2. face_recognition.face_encodings(face_image, known_face_locations=None, num_jitters=1): This function generates a 128-dimension encoding for a face image.

3. face_recognition.compare_faces(known_face_encodings, face_encoding_to_check, tolerance=0.6): This function compares a face encoding with a list of known face encodings to determine if they match.

4. face_recognition.face_distance(face_encodings, face_to_compare): This function calculates the Euclidean distance between a face encoding and a list of face encodings to determine the similarity score.

5. face_recognition.load_image_file(file, mode='RGB'): This function loads an image file into a numpy array in RGB format.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

• **OpenCV:**

The cv2 module is used in the Face Recognition Attendance System to capture frames from the video stream and perform various image processing tasks, including face detection, resizing, and drawing. cv2 is a popular computer vision library that provides a wide range of functions and algorithms for image and video processing.

In this project, the cv2 module is used to capture frames from the video stream and perform face detection. The module provides various functions for face detection, including Haar Cascade, Local Binary Patterns (LBP), and Deep Learning-based face detection using Single Shot Detector (SSD) and Region-based Convolutional Neural Network (R-CNN).

• **cvzone:**

The cvzone module is a computer vision library built on top of OpenCV and provides a collection of useful functions and classes for image and video processing. It is used in the Face Recognition Attendance System to draw GUI elements on the output video stream, including text, buttons, and rectangles.

In this project, the cvzone module is used to create a GUI interface that displays the attendance status of the recognized faces in real-time.

• **Firebase SDK:**

The Firebase Admin SDK is a set of tools for building server-side applications that interact with Firebase services. The Face Recognition Attendance System repository uses the firebase_admin module to connect to and interact with Firebase Cloud Firestore, which is a NoSQL document database that stores data in documents organized into collections.

Here are some of the functions used in the Face Recognition Attendance System:

1. initialize_app(cred, options=None, name=None): This function initializes the Firebase Admin SDK with the given credentials and options. The credentials are obtained from the Firebase Console, and the options can be used to specify the project ID and other settings.

2. set(value): This function is used to update attendance records for students. When a student is recognized, the system checks if the corresponding record exists in the database, and if so, updates the attendance status for that student using the set() function. If the record does not exist, a new record is created with the student's name and attendance status.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

5. Algorithms of face_recognition module

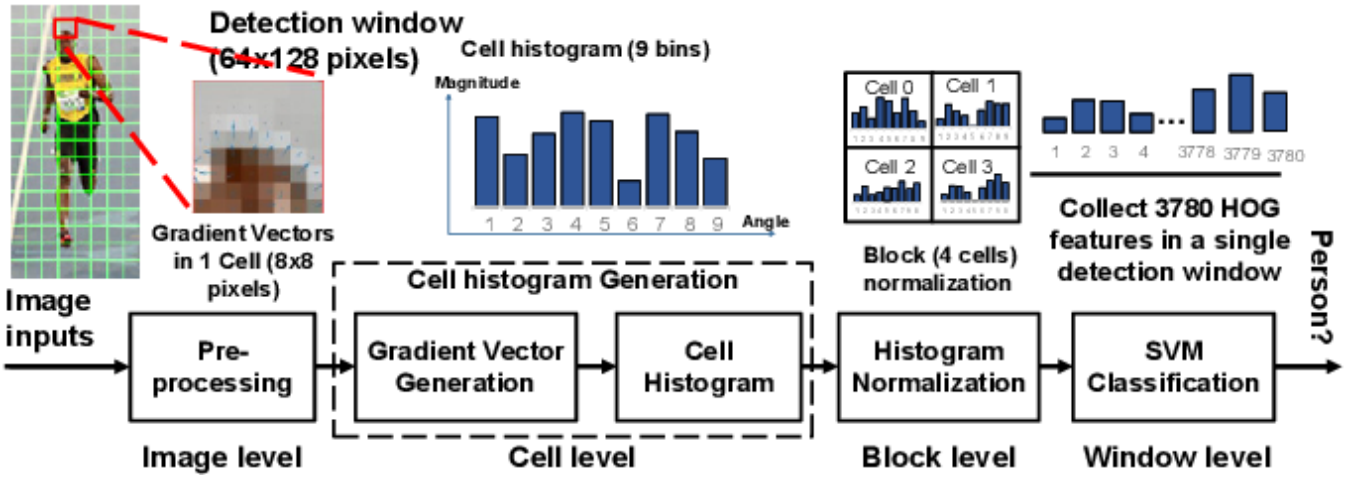
Some of the advanced algorithms used in the module are listed as follows:

1. **Histogram of Oriented Gradients (HOG) algorithm**
   This algorithm is used for detecting the presence of a face in an image by identifying the location of facial features such as eyes, nose, and mouth.

   The HOG algorithm extracts features from an image by dividing the image into small, overlapping cells and computing the histogram of gradient orientations within each cell. The gradients are computed using simple derivative filters (e.g., Sobel filters) to estimate the direction and magnitude of local intensity changes in the image. The histogram of gradient orientations captures the distribution of gradient directions within the cell, providing a robust and efficient representation of local image structure.

   After computing the histograms of gradient orientations for each cell, adjacent cells are combined into larger blocks and normalized using a technique known as block normalization. Block normalization helps to improve the robustness of the feature extraction process by accounting for variations in illumination and contrast across different regions of the image.

   Finally, the HOG features are typically fed into a machine learning algorithm (e.g., SVM, neural network) to train a classifier for object detection or recognition. The HOG features provide a compact and discriminative representation of the image, allowing for efficient and accurate classification.



Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

2. **Convolutional Neural Network (CNN) algorithm**
   CNN is a deep learning algorithm that is capable of learning complex representations of images. It is used in face_recognition for extracting high-level features from facial images, such as facial expressions, gender, and age. This algorithm is trained on a large dataset of facial images and is capable of accurately recognizing faces even under different lighting conditions and angles.

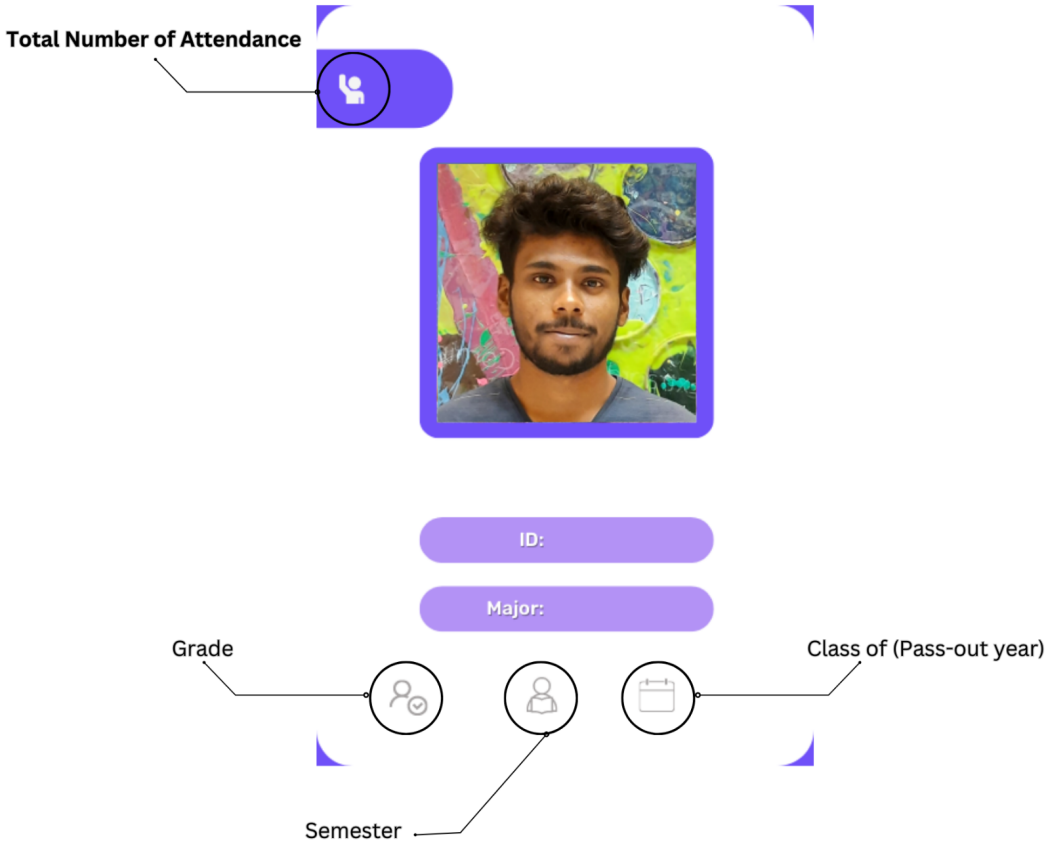
3. **Face Alignment Technique:** The face_recognition module uses this technique to improve the accuracy of face recognition. This technique involves identifying facial landmarks such as eyes, nose, and mouth, and then aligning the facial image based on these landmarks. This technique helps to reduce the variability in facial images caused by differences in head pose and facial expression.

4. In addition to these algorithms, face_recognition also employs other techniques such as clustering, **principal component analysis** (PCA), and **linear discriminant analysis** (LDA) to improve the accuracy and speed of face recognition tasks.

6. Results

All the programs worked completely fine as expected. The attendance was marked and updated in the realtime database along with the last time of attendance. On running the program, the web cam activates and scans the person standing in front. It determines the face locations, encodes them, and finally, compares them with the face-encodings of the faces already in the database. If the face matches, then the program fetches the student information from the database and displays it in the format as shown below:

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System



From this point on, we are going to show some of the significant code-snippets that did the most significant functions in the project starting from setting up the web cam to fetching data from Firebase.

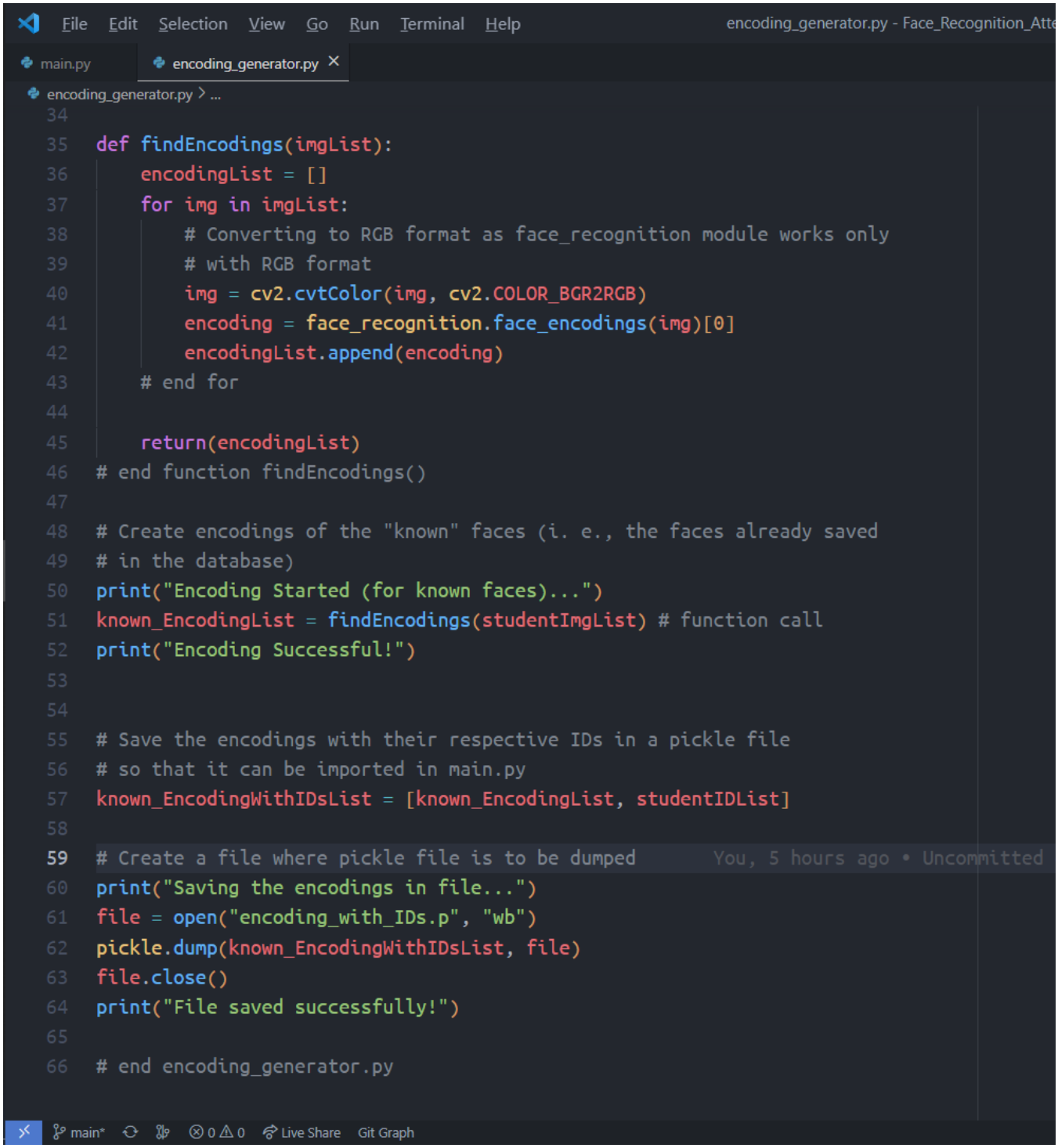
Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

```
# Set-up the web-cam
capture = cv2.VideoCapture(0) # zero for default camera
capture.set(3, 640) # 3 -> CAP_PROP_FRAME_WIDTH
capture.set(4, 480) # 4 -> CAP_PROP_FRAME_HEIGHT
# Intialise variables
modeType = 0 # current state of the attendance cam
frameCount = 0 # track number of frames captured and time to be waited out
studentID = -1 # student id whose face matched
studentImgList = []

while True:
    success, img = capture.read()

    # Overlay the webcam-captured image on the background
    imgBackground[162: 162 + 480, 55: 55 + 640] = img
    # Overlay the current mode image
    imgBackground[44: 44 + 633, 808: 808 + 414] = modesImgList[modeType]
```

Here, the web cam is set-up using OpenCV's VideoCapture() method and the width and height of the *capture* object are set. A while loop is used along with VideoCapture() method to keep capturing frames which actually means recording a video. In this snippet, the webcam is also overlaid on a background along with the mode image beside it.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

The encodings of the knwon faces were generated by this code snippet:



face_recognition module's findEncodings() method generates an encoding of the student image (face only) as a 128 dimensional array.

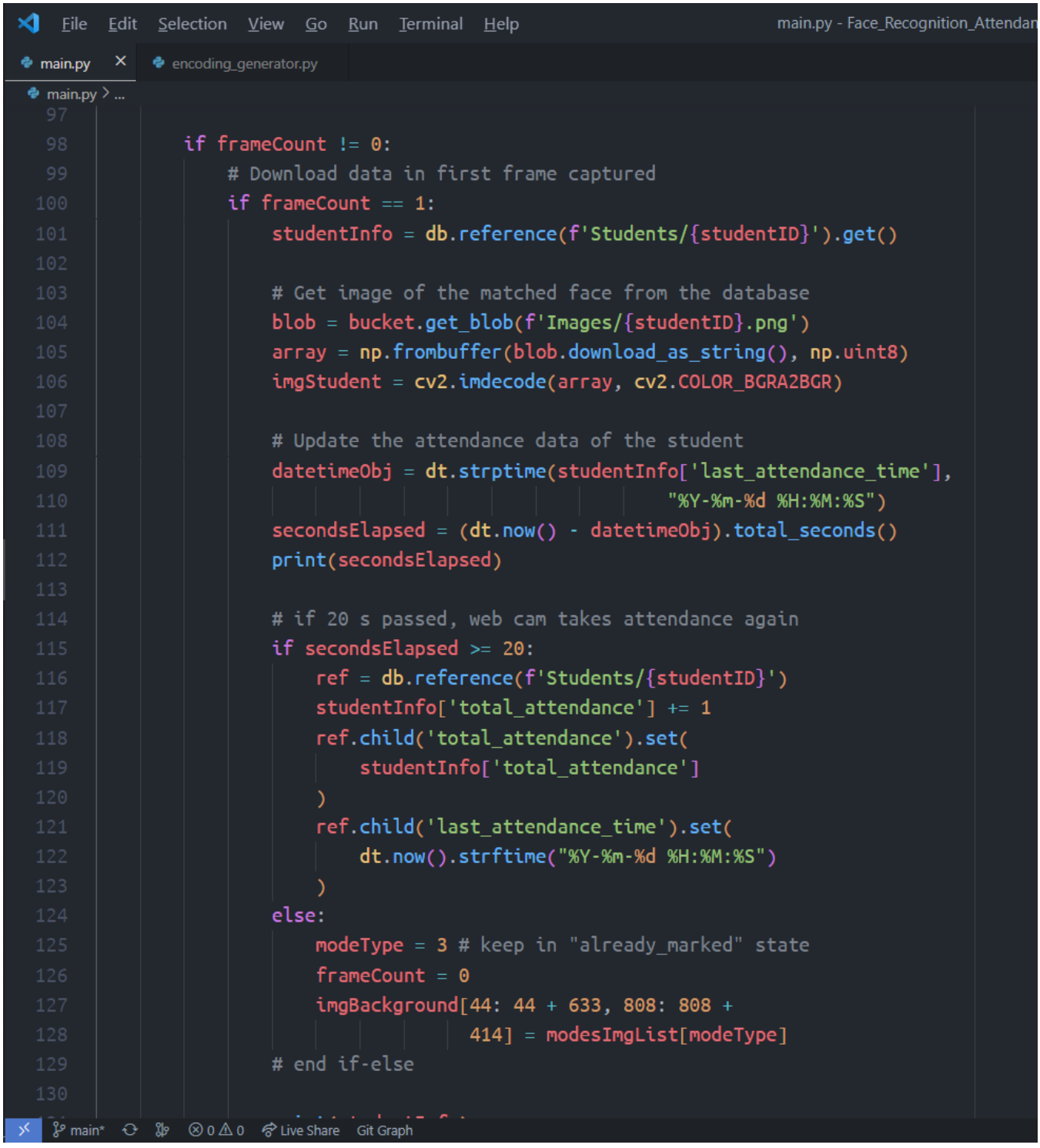
Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

The face recognition and comparisons functions were done by this code snippet as shown below:

```
59      # Prepare the image from webcam for commparisons
60      imgSmall = cv2.resize(img, (0, 0), None, 0.25, 0.25) # down-scaling image
61      imgSmall = cv2.cvtColor(imgSmall, cv2.COLOR_BGR2RGB)
62
63      # Use face_recognition module to locate the faces        You, 5 hours ago • Unc
64      webCamFaceLoc = face_recognition.face_locations(imgSmall)
65      webCamFaceEncoding = face_recognition.face_encodings(imgSmall,
66                                                           webCamFaceLoc)
67
68      if webCamFaceLoc:
69          # Compare the web cam face-encodings with the saved encodings
70          for faceEncode, faceLoc in zip(webCamFaceEncoding, webCamFaceLoc):
71              faceMatches = face_recognition.compare_faces(known_encodingList,
72                                                           faceEncode)
73              faceDists = face_recognition.face_distance(known_encodingList,
74                                                         faceEncode)
75
```

The face_recognition module's face_location() method detected the location of face and the compare_faces() method compared the encodings of the already known faces (in the database) against the encoding of the current face captured by the web cam.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

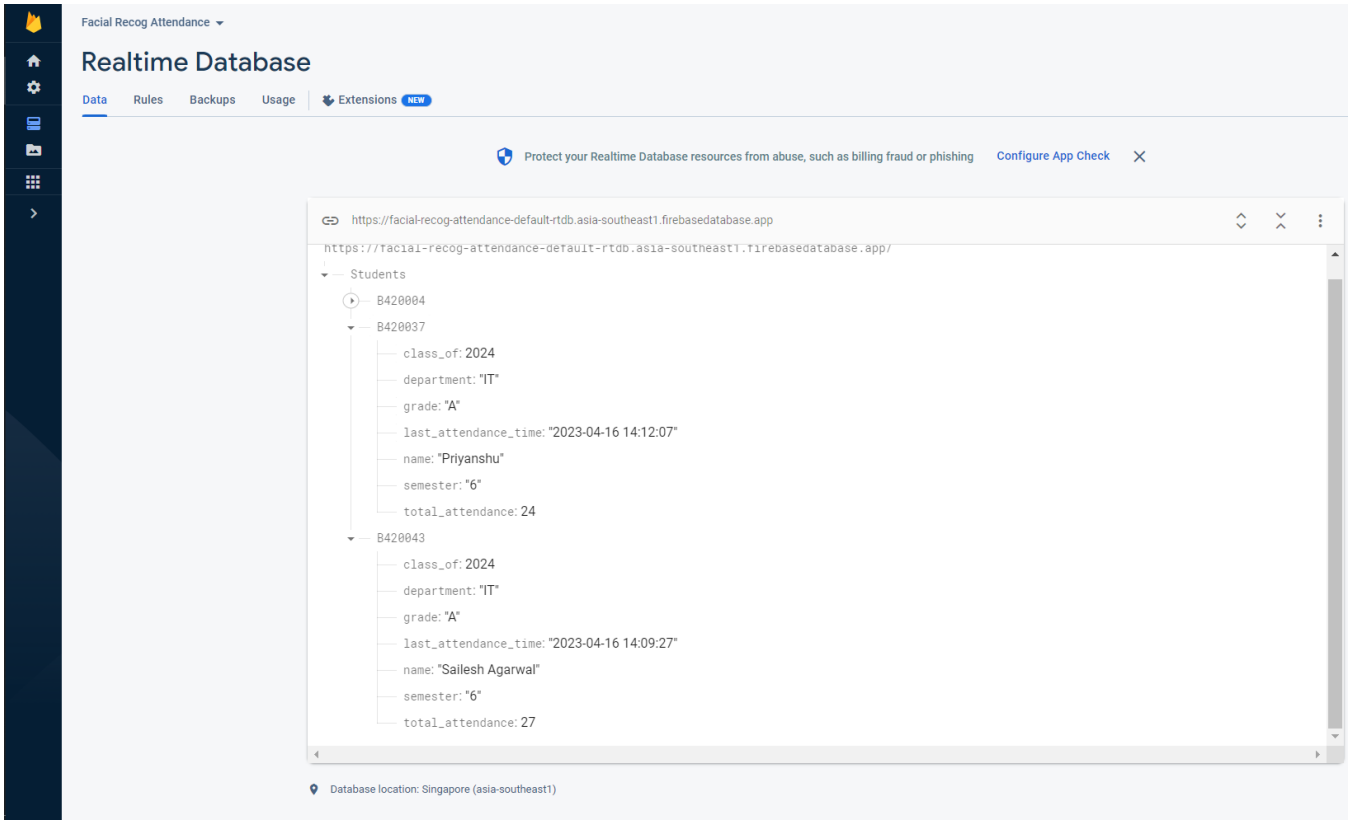


Here , the *datetimeObj* stores the date and time of the most recent attendance-marking (i.e., when someone just showed their face for scanning and the attendance was marked). The *secondsElapsed* variable tracks the time interval between current time and the last attendance time. This is used to limit the student's attendance if s/he kept standing in front of the of the camera. For real-world

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

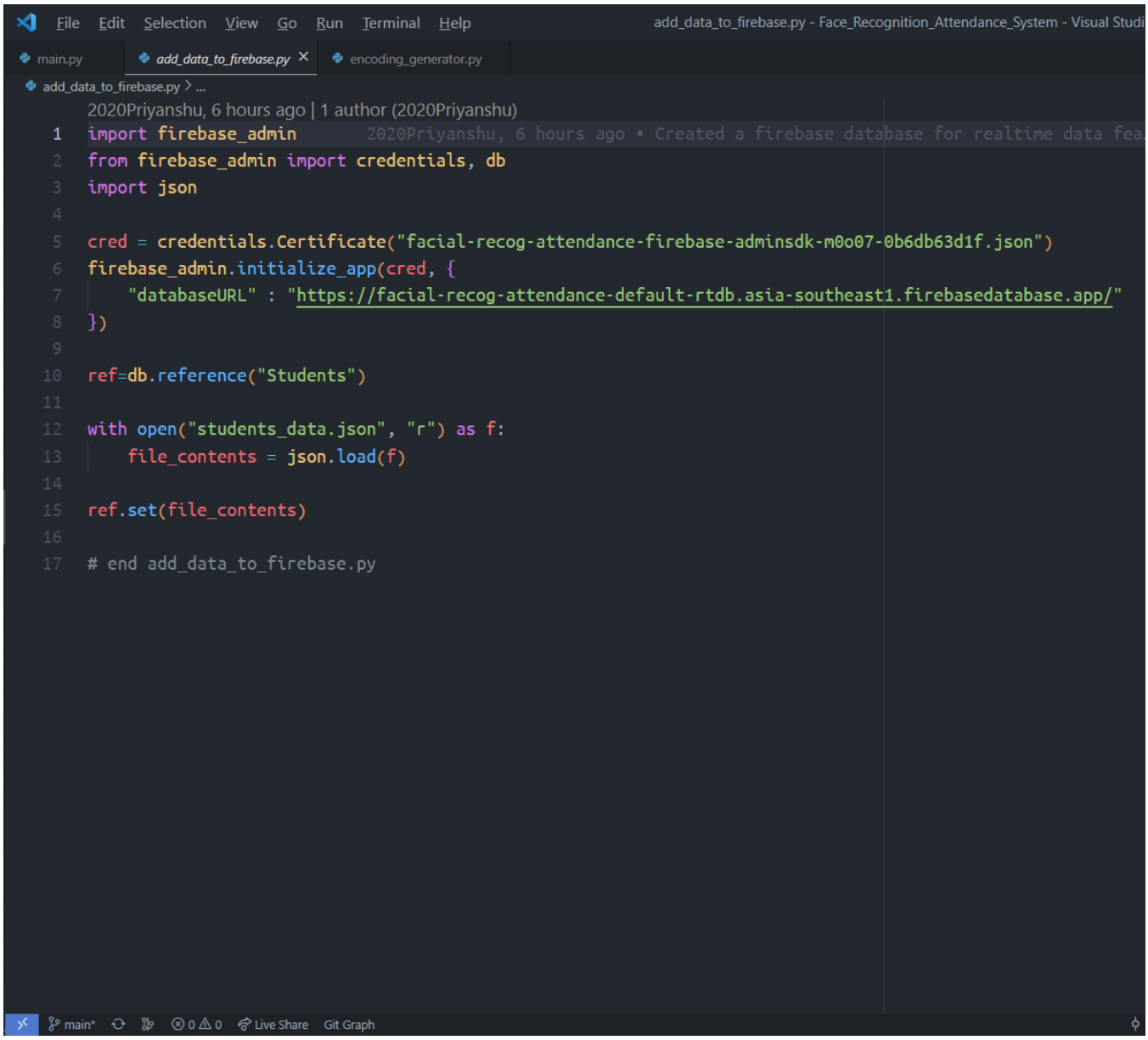
scenarios, we can use (24 hours x 3600) seconds as the limit above which the system will consider another attendance for the same student.

The real-time database was made at Google's Firebase:



We chose Google Firebase because Firebase is an easy-to-use platform for with real-time database, authentication, analytics, cloud functions, and seamless integration with other Google services.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System



In the above snippet, the database itself was created and set-up using the “Private Key” of the database. This ensures that we can update the database realtime from our virtual environment. Then the data of all students which is stored in the *students_data.json* file is read and uploaded in the database using the set() function. To execute the above process we use the reference data structure of db imported from firebase_admin.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

File Edit Selection View Go Run Terminal Help

students_data.json

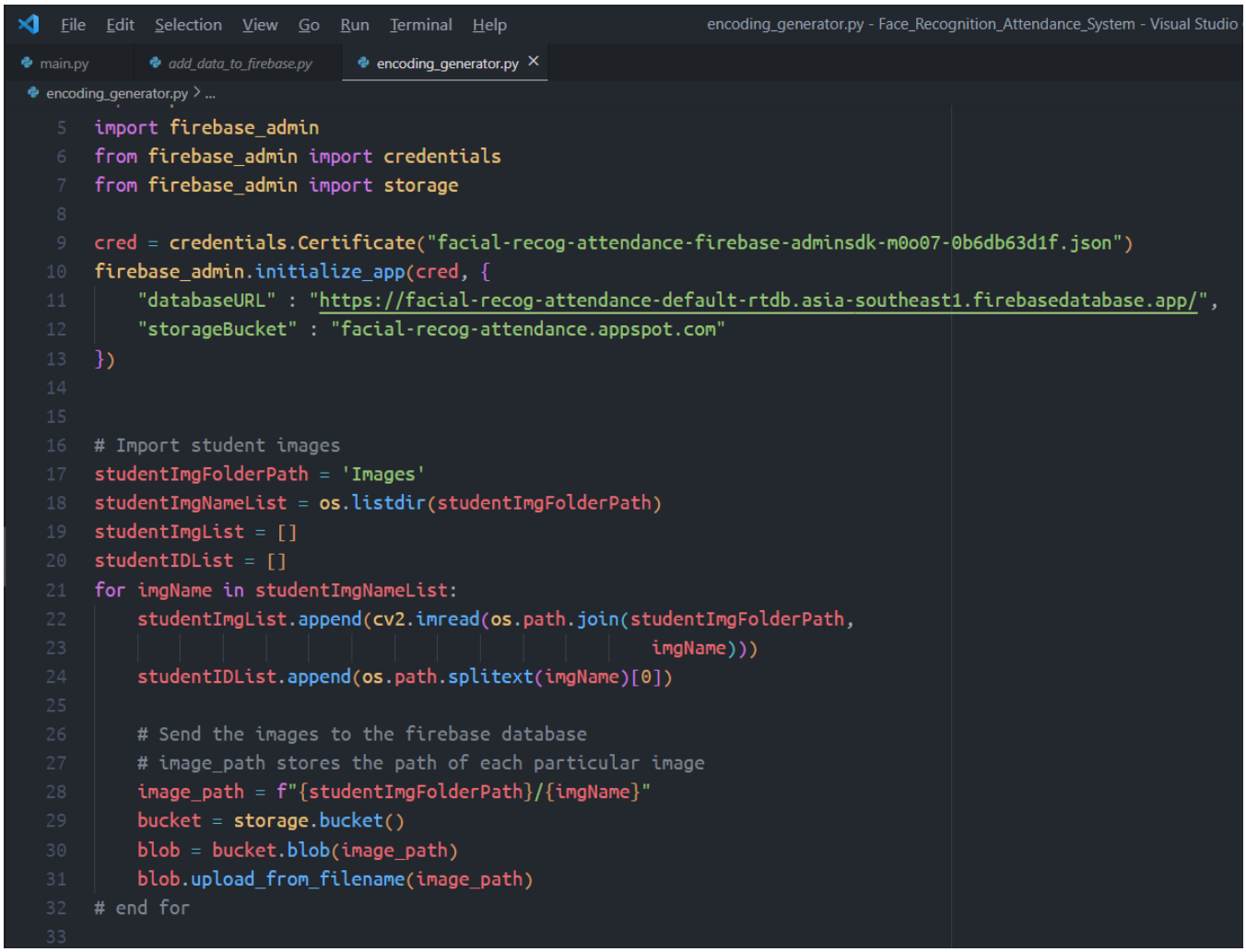
C: > Users > vkfak > OneDrive > Documents > IVP Project > Face_Recognition_Attendance_System > students_da

2020Priyanshu, 7 hours ago | 1 author (2020Priyanshu)

```
{       2020Priyanshu, 7 hours ago • Created a firebase datal
    "B420004":
    {
        "name": "Akash Parida",
        "department": "IT",
        "class_of": 2024,
        "total_attendance": 17,
        "grade": "A",
        "semester": "6",
        "last_attendance_time": "2023-04-11 13:54:34"
    },
    "B420037":
    {
        "name": "Priyanshu",
        "department": "IT",
        "total_attendance": 19,
        "grade": "A",
        "class_of": 2024,
        "semester": "6",
        "last_attendance_time": "2023-04-11 13:54:34"
    },
    "B420043":
    {
        "name": "Sailesh Agarwal",
        "department": "IT",
        "total_attendance": 16,
        "grade": "A",
        "class_of": 2024,
        "semester": "6",
        "last_attendance_time": "2023-04-11 13:54:34"
    }
}
```

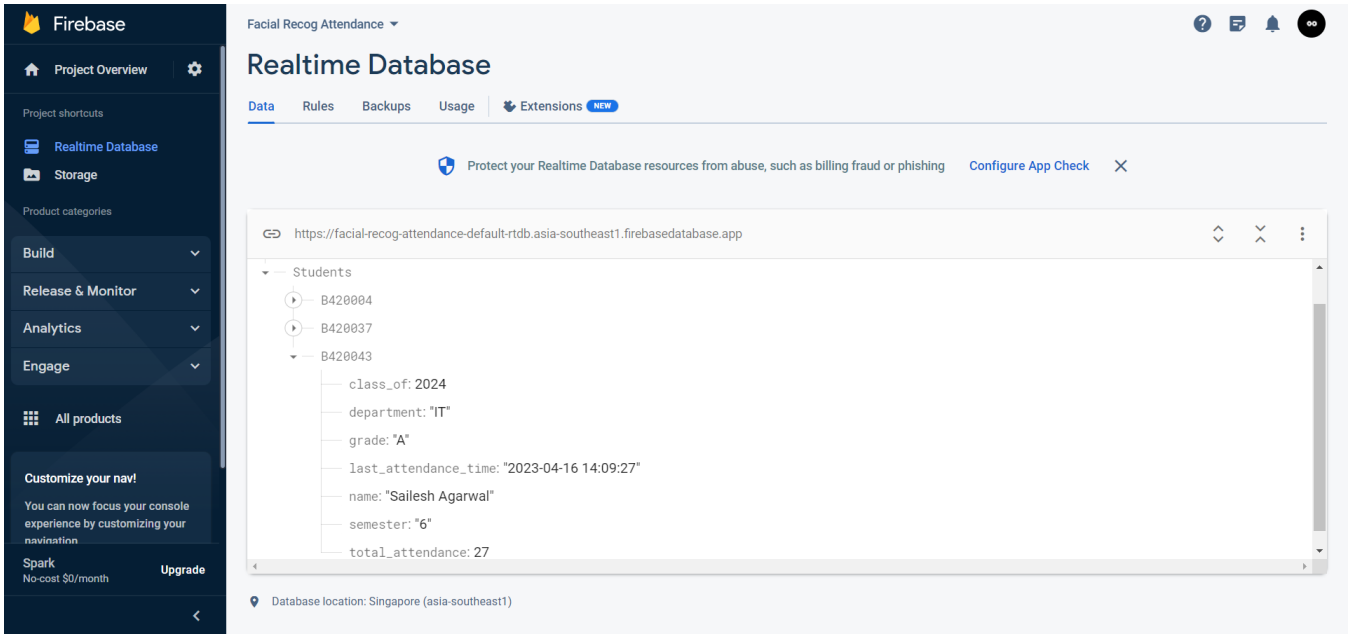
The *students_data.json* file stores the data in the shown structure whose reference is set iin the *add_data_to_firebase.py* file

The following snippet adds images to the Firebase database:

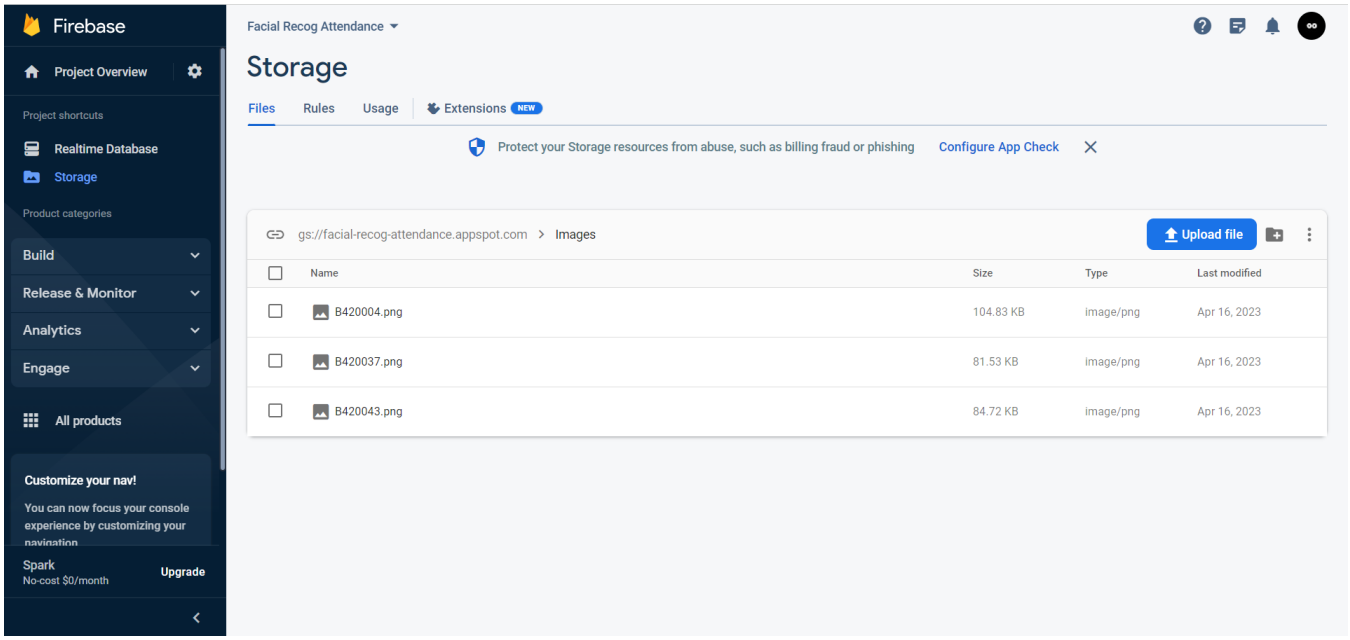


The bucket data structure used above is a unit for storage of data. It is created to store the images into the “storage” of our firebase database by using the blob structures. Blobs are simply an immutable class representing an array of bytes in Cloud Firestore. They are the basic unit of data storage.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System



The above snippet shows the Realtime Database of FRAS on Firebase used to store the student data



The above snippet shows the Storage of FRAS on Firebase used to store the images

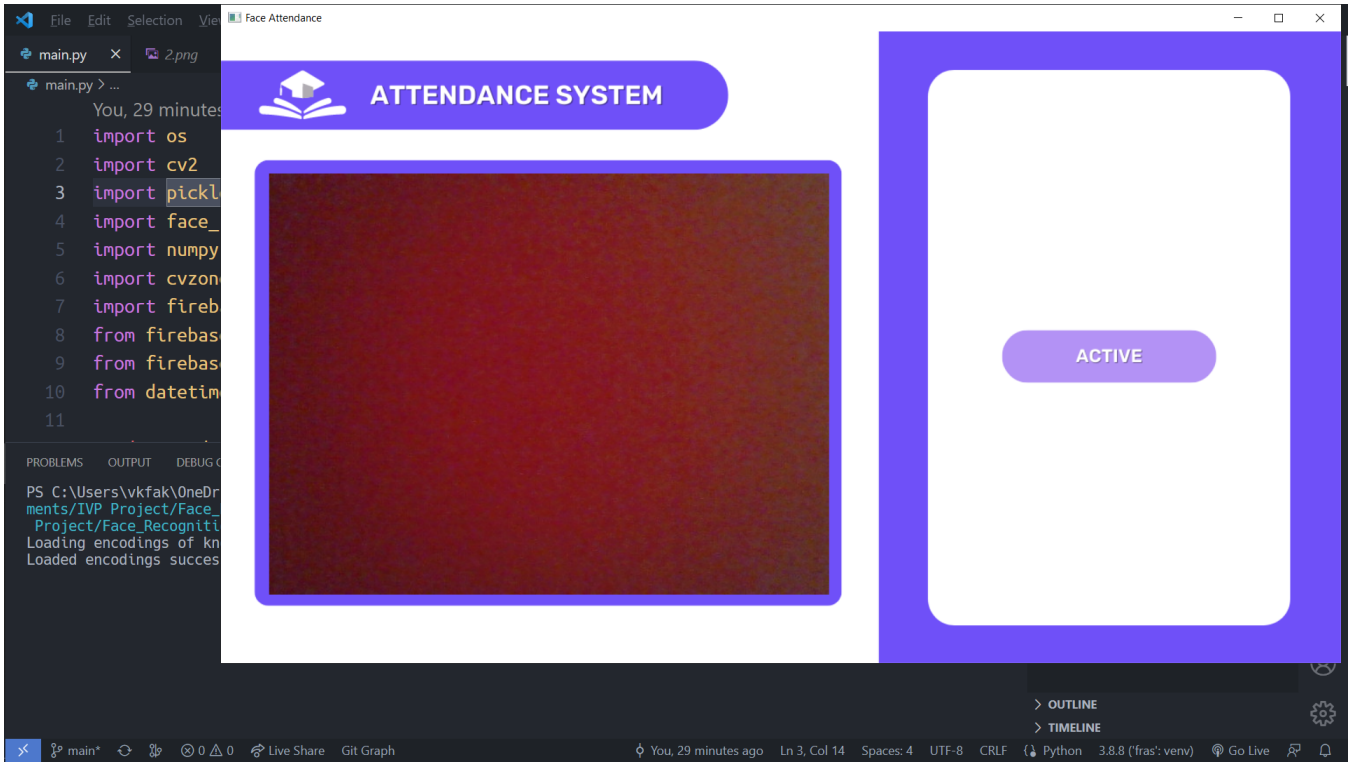
Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

The code to display the name is separated as we needed to centralise the name on the mode image "3". First, we found out the width of the name by using the getTextSize() method of OpenCV in pixels. Next, we find the difference between the total width of the space available for the name label and the text width. Dividing it by 2, we get the offset. So we simply put the name text after leaving the "*offset*"

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

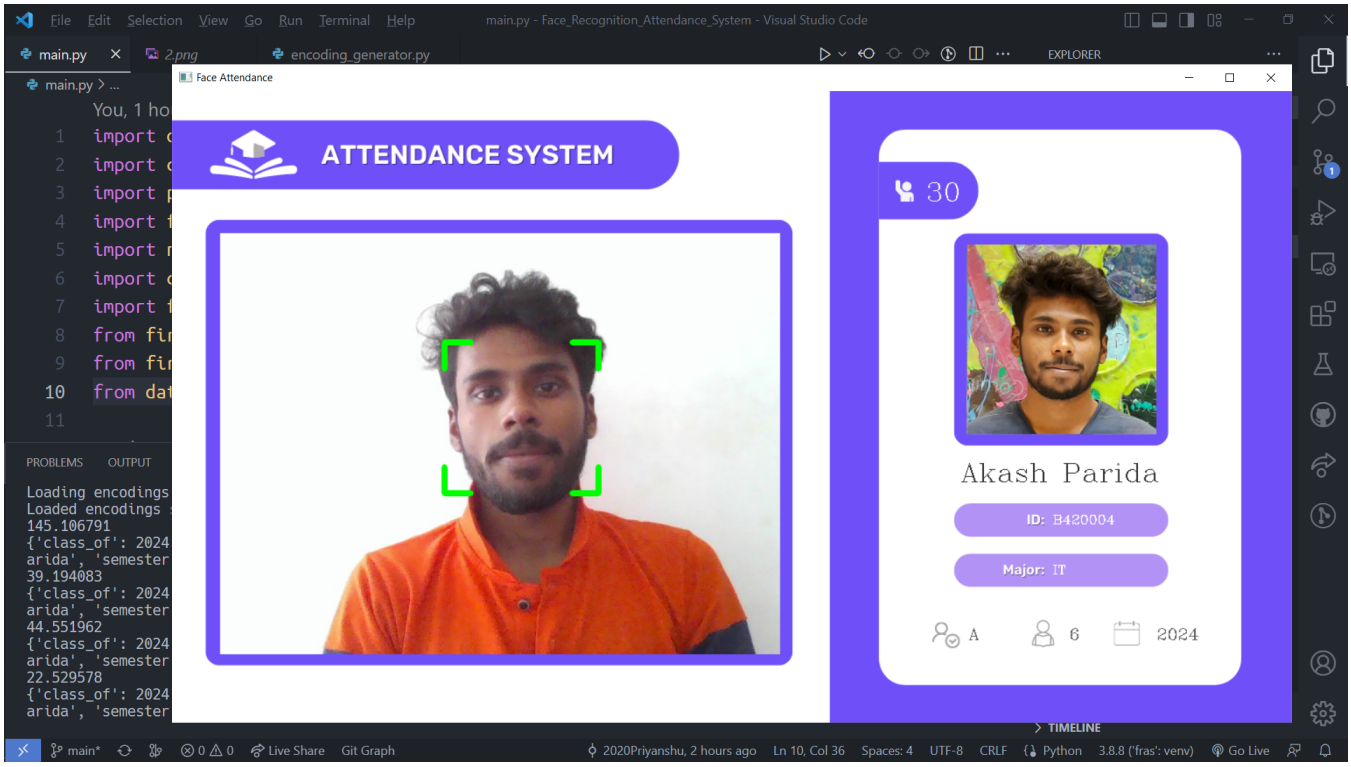
amount of space from the starting. OpenCV's putText() method adds a text onto the image provided as an argument.

Now, let's see the outcomes after running the main.py:

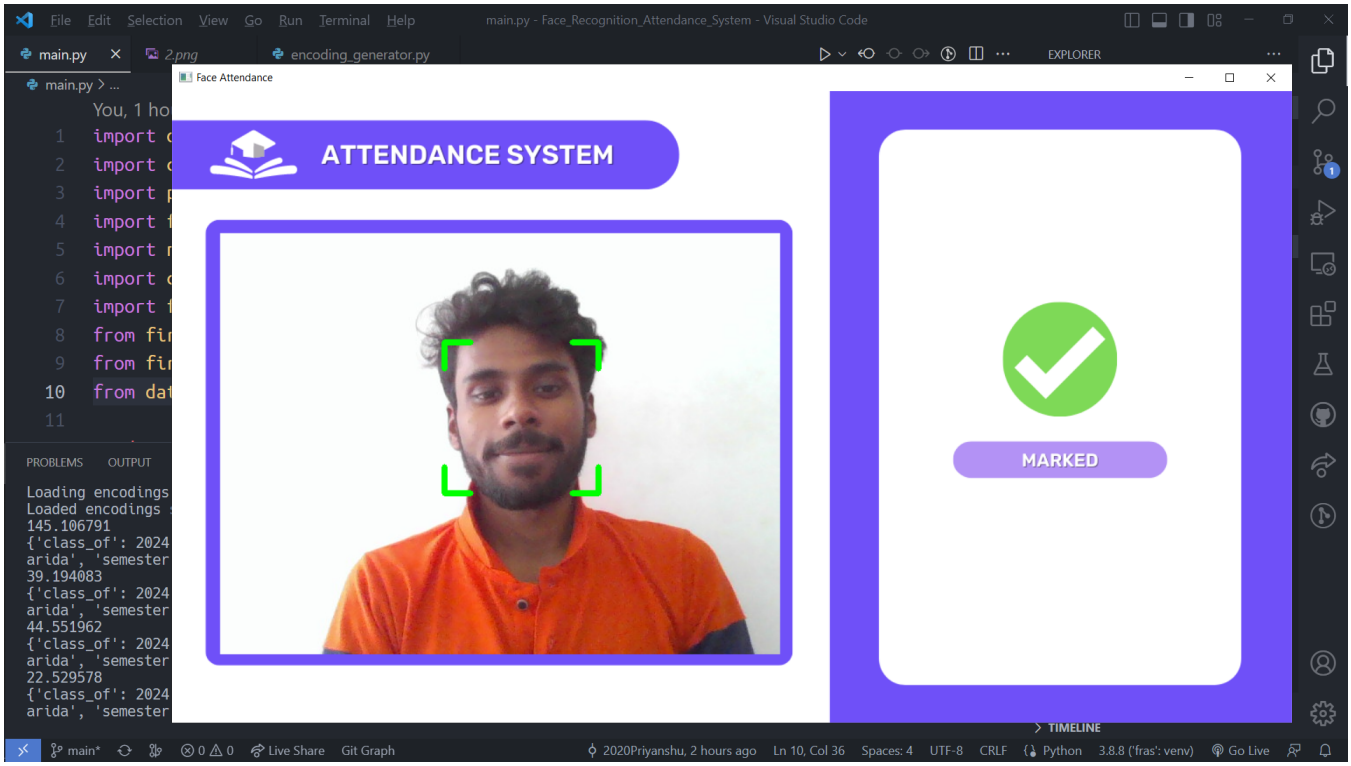


This shows that the web cam is active and scanning when no face is being shown to it.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

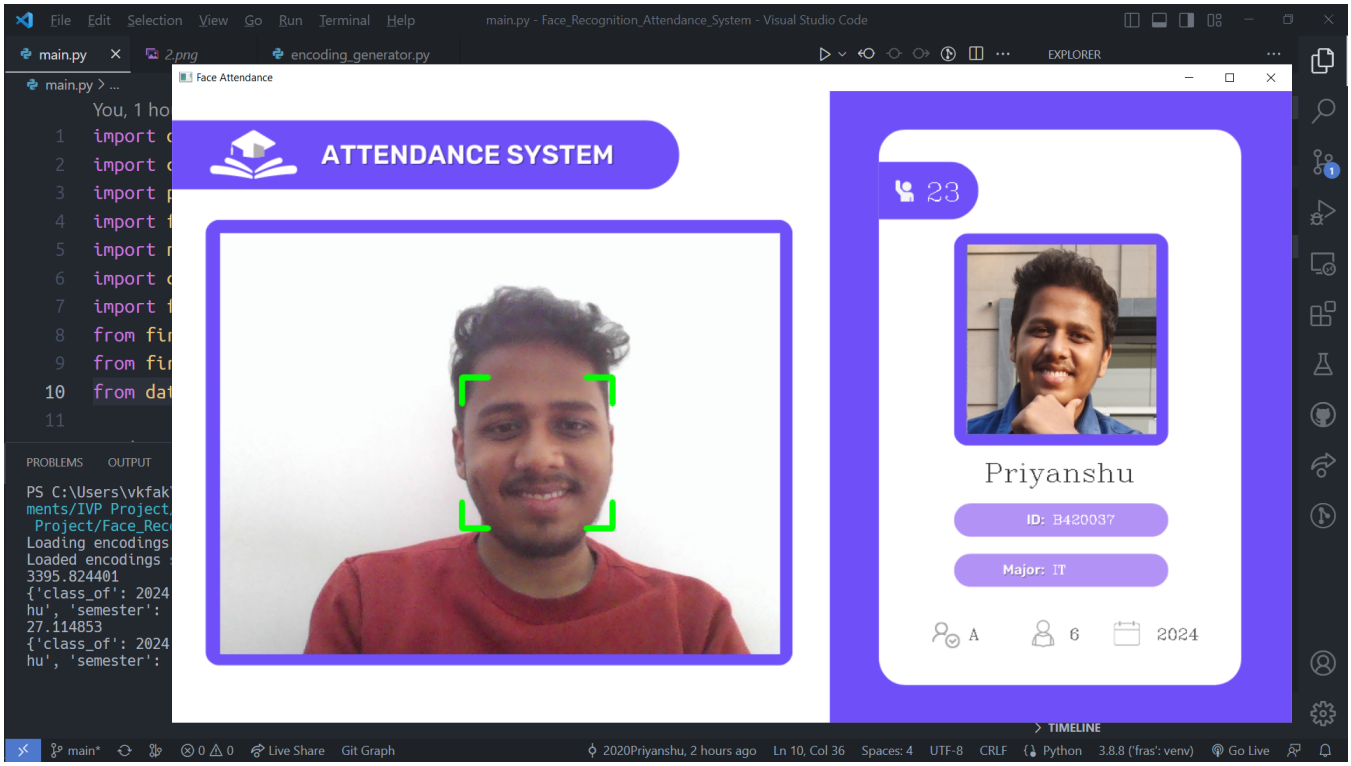


When a face is shown, it compares and matches the face. When the face matched, the program fetched the student info (along with the image from the database) and displayed it as shown.

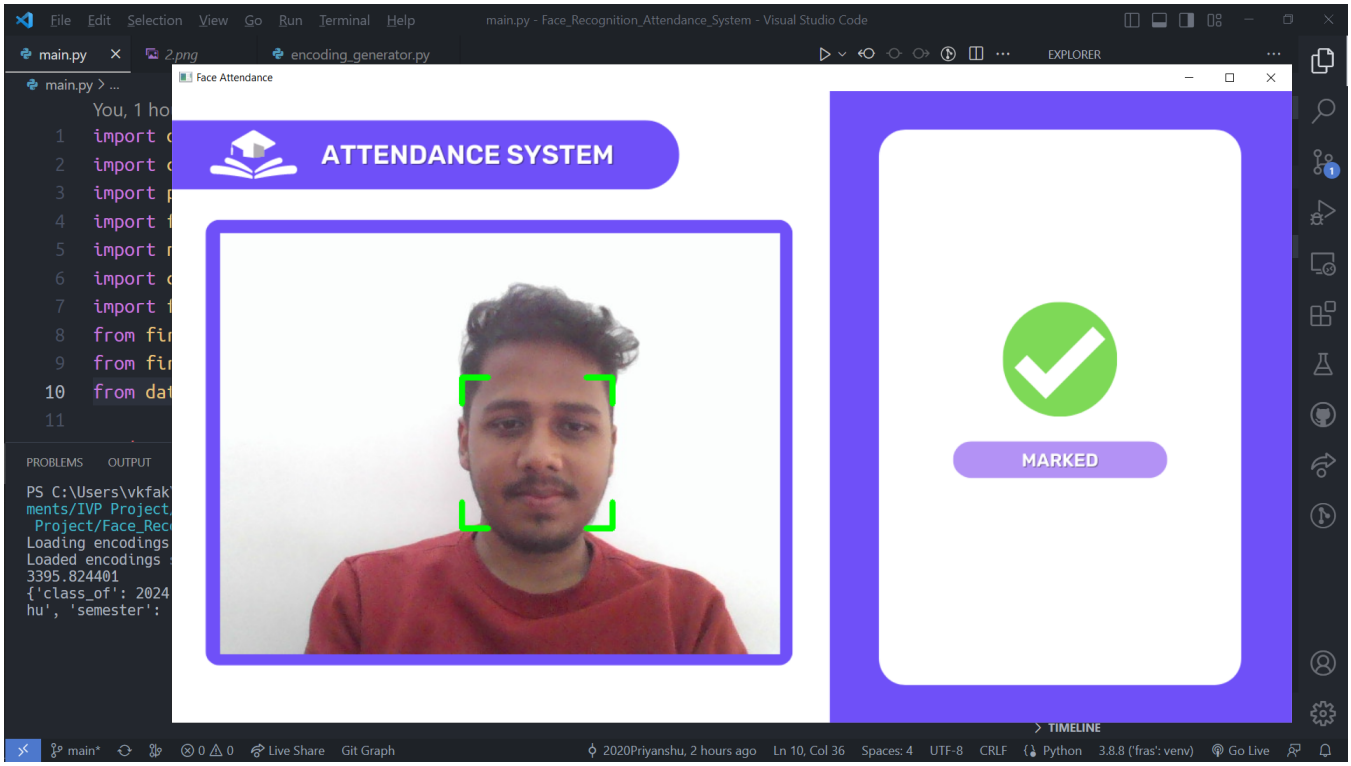


After a certain amount of time, the program shows marked if the matched face is still lingering around.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

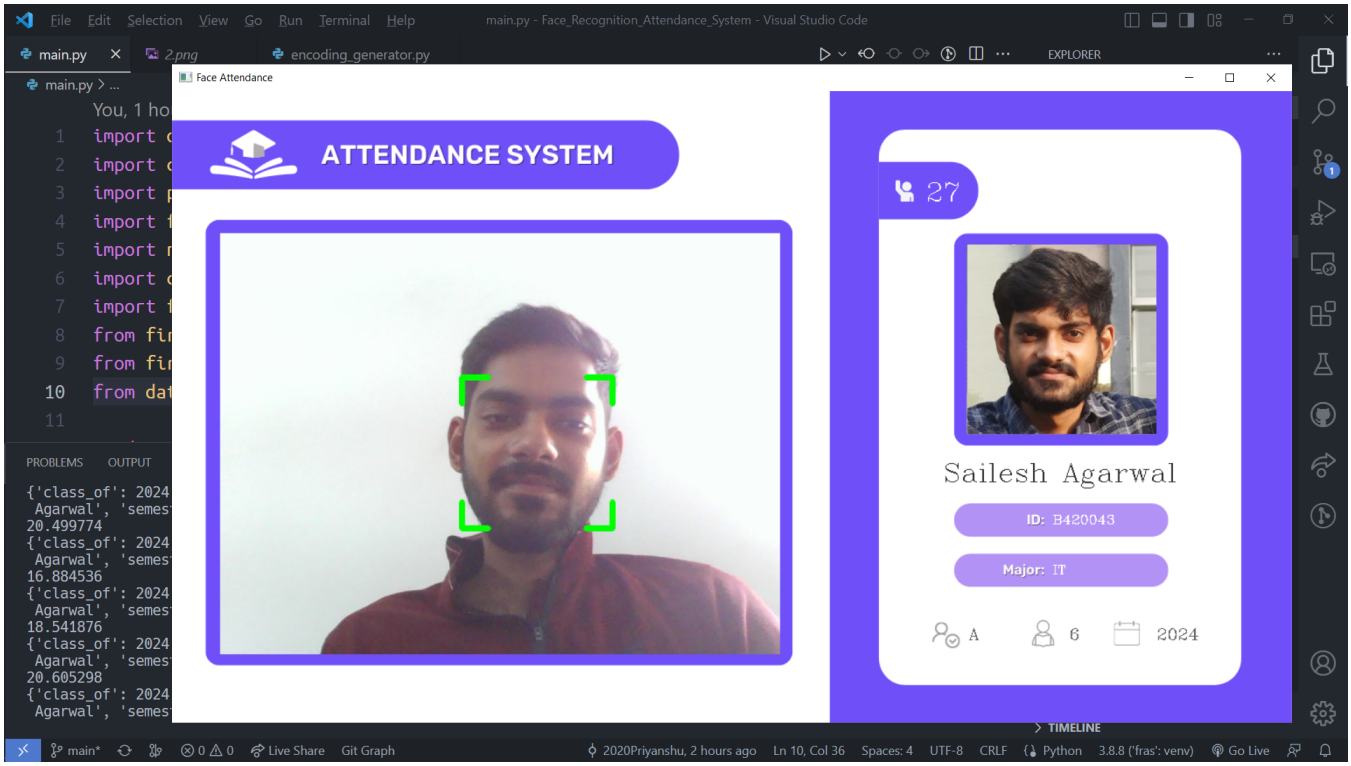


This snap shows the program is identifying faces and marking attendance correctly.

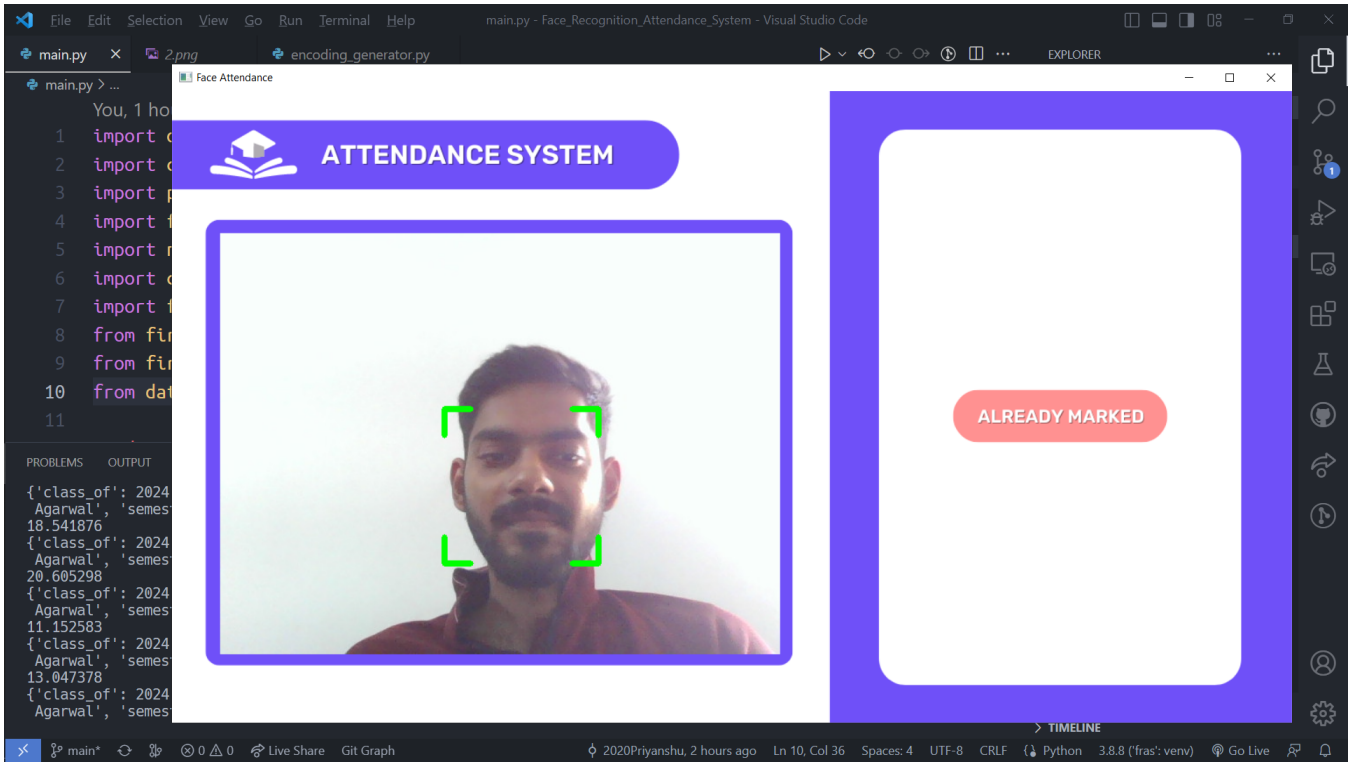


The student was still there, so the program showed “marked”.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System



The attendance of the last member in the student database.



Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

If the same student shows her/his face again within 20 seconds of last attendance, the attendance system shows “already marked”.

In conclusion, the "Face Recognition Attendance System" project is a successful implementation of face recognition technology in the domain of image and video processing. The system is capable of detecting faces, recognizing them, and marking attendance accordingly. It has proved to be an efficient and reliable method for attendance management in various institutions.

Note: All of the resources and source codes can be found in the following github repository link which is also provided in the document’s footer:
GitHub Repository Link

7. Alternative approach

The significant part of the project that is face recognition, is done by the module “face_recognition”. So as an alternative approach we tried to build our own DL Model using transfer learning. We used VGG16 and trained it on our dataset which contained 4 labels namely,

- VIRAT KOHLI
- SHARAPOVA
- FEDRAR
- MESSI

The model is built using libraries like Tensorflow, Numpy,Opencv, Image etc. The VGG16 has 14,815,044 params out of which 100,356 where trainable and rest were frozen.The dataset contained 137 images for 4 classes in the training set and 137 images for 4 classes in the test set.

```
=================================================================
Total params: 14,815,044
Trainable params: 100,356
Non-trainable params: 14,714,688

_________________________________________________________________
Found 137 images belonging to 4 classes.
Found 137 images belonging to 4 classes.
```

The code for the model development and the model usage is provided below:

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

```
# re-size all the images to this
IMAGE_SIZE = [224, 224]

train_path = 'dataset/train'
valid_path = 'dataset/test'

# add preprocessing layer to the front of VGG
vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

# don't train existing weights
for layer in vgg.layers:
  layer.trainable = False



  # useful for getting number of classes
folders = glob('dataset/train/*')


# our layers - you can add more if you want
x = Flatten()(vgg.output)
# x = Dense(1000, activation='relu')(x)
prediction = Dense(len(folders), activation='softmax')(x)

# create a model object
model = Model(inputs=vgg.input, outputs=prediction)

# view the structure of the model
model.summary()
```

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

```
# tell the model what cost and optimization method to use
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)


from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory('dataset/train',
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')

test_set = test_datagen.flow_from_directory('dataset/test',
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
# fit the model
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=5,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
# loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# accuracies
# plt.plot(r.history['acc'], label='train acc')
# plt.plot(r.history['val_acc'], label='val acc')
# plt.legend()
# plt.show()
# plt.savefig('AccVal_acc')
```

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

```
##Now we open the webcam and capture the face from it and recognition the photo!

# Face Recognition

# Importing the libraries
from PIL import Image
from keras.applications.vgg16 import preprocess_input
import base64
from io import BytesIO
import json
import random
import cv2
from keras.models import load_model
import numpy as np
import tensorflow as tf
import numpy as np

from keras.preprocessing import image
model =  tf.keras.models.load_model('facefeatures_new_model_new_4_classes.h5')

# Loading the cascades
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

```
def face_extractor(img):
    # Function detects faces and returns the cropped face
    # If no face detected, it returns the input image

    #gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(img, 1.3, 5)

    if faces is ():
        return None

    # Crop all faces found
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)
        cropped_face = img[y:y+h, x:x+w]

    return cropped_face
```

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

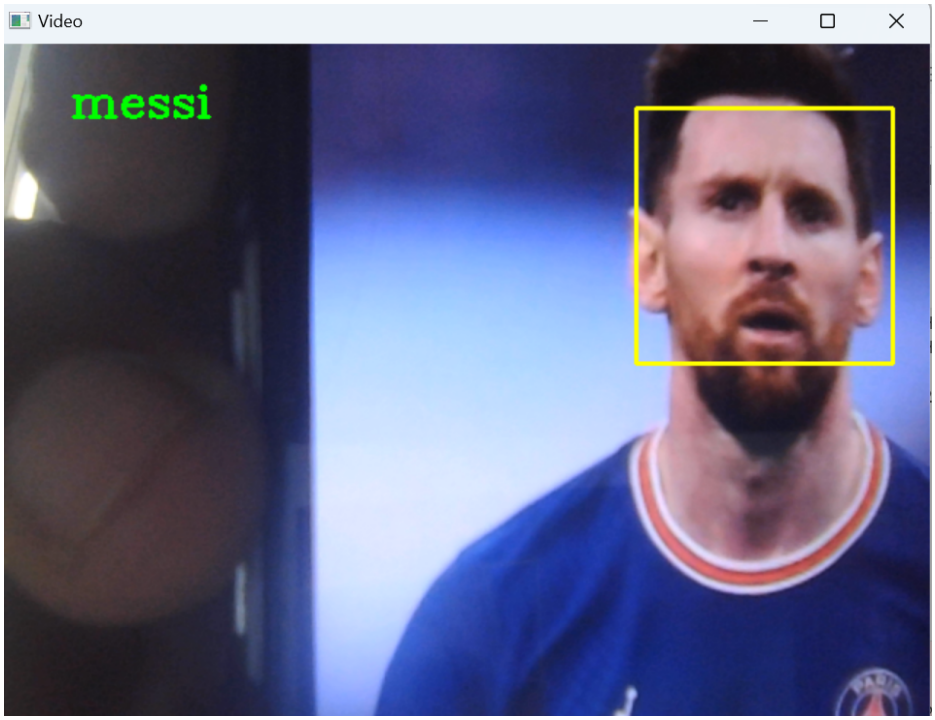
```
# Doing some Face Recognition with the webcam
video_capture = cv2.VideoCapture(0)
while True:
    _, frame = video_capture.read()
    #canvas = detect(gray, frame)
    #image, face =face_detector(frame)

    face=face_extractor(frame)
    if type(face) is np.ndarray:
        face = cv2.resize(face, (224, 224))
        im = Image.fromarray(face, 'RGB')
           #Resizing into 128x128 because we trained the model with this image size.
        img_array = np.array(im)
                    #Our keras model used a 4D tensor, (images x height x width x channel)
                    #So changing dimension 128x128x3 into 1x128x128x3
        img_array = np.expand_dims(img_array, axis=0)
        pred = model.predict(img_array)
        print(pred)

        name="None matching"
        if(np.argmax(pred[0])==0):
            name='messi'
        elif(np.argmax(pred[0])==1):
            name='sharapova'
        elif(np.argmax(pred[0])==2):
            name='fedrar'
        elif(np.argmax(pred[0])==3):
            name='virat'
```

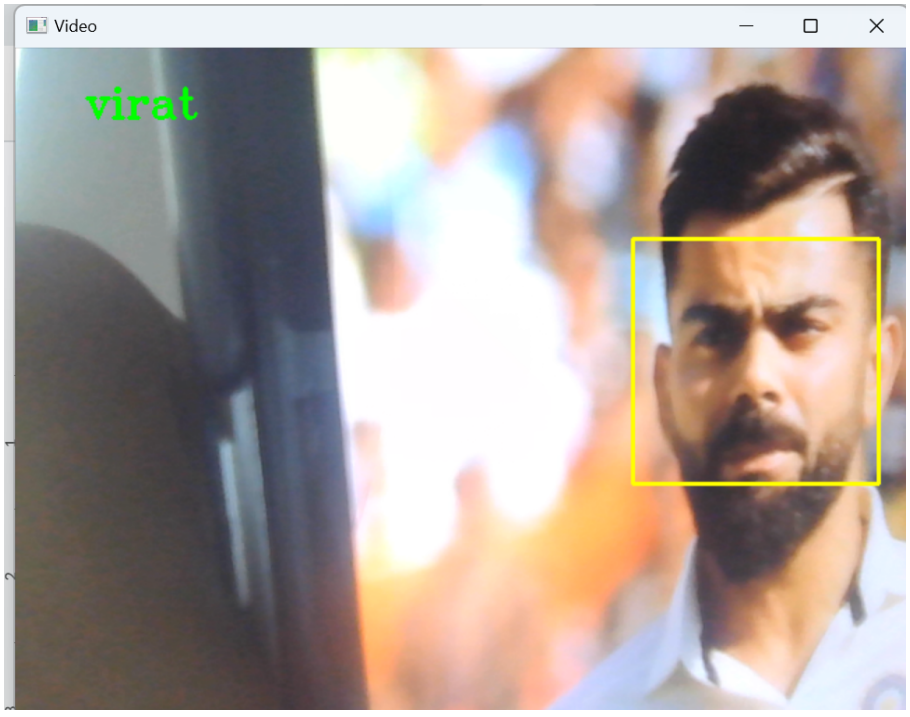
```
        cv2.putText(frame,name, (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
        cv2.putText(frame,name, (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
    else:
        cv2.putText(frame,"No face found", (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
    cv2.imshow('Video', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
video_capture.release()
cv2.destroyAllWindows()
```

GOAL OF THE MODEL:The model is capturing iamge from the system webcam as an input and providing the label as an output.

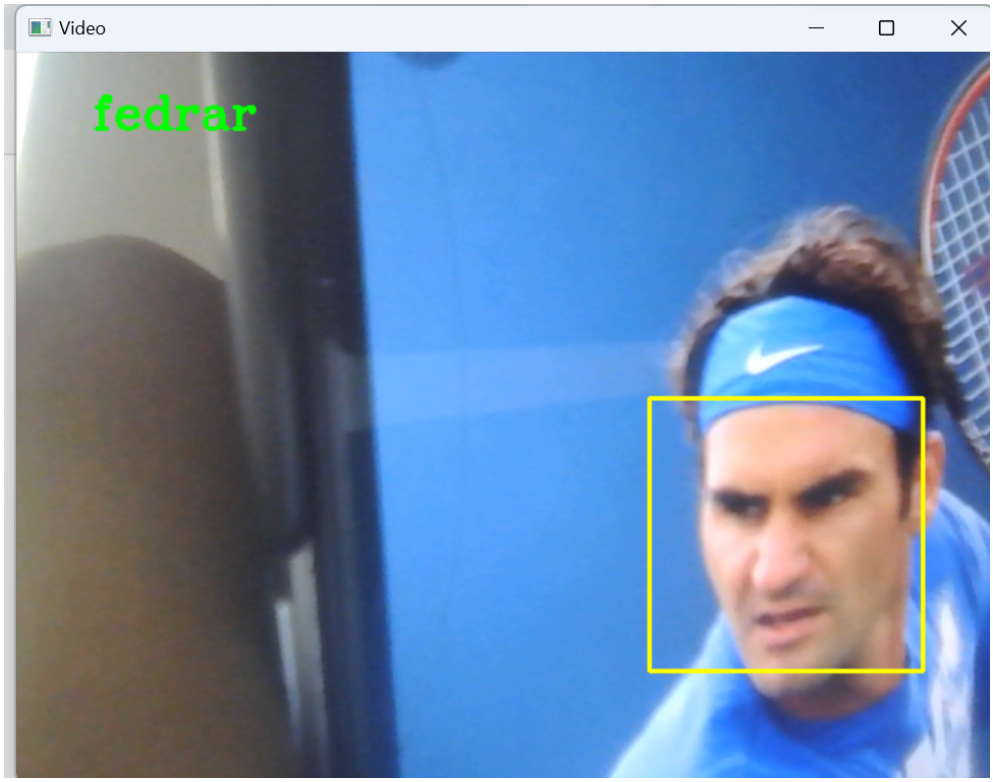
The screenshots of the output is added below:

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System





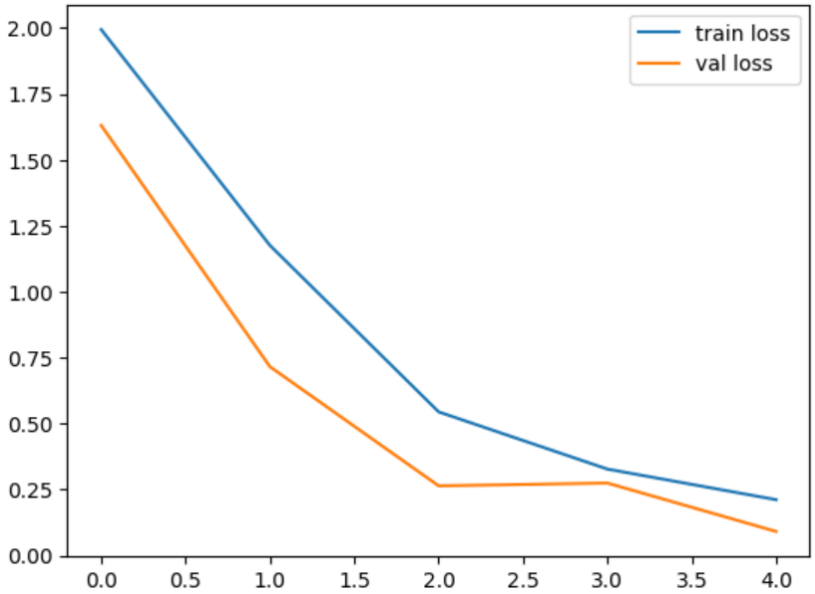
Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System



The training accuracy of the model is above 90% and validation accuracy is over 99%

```
Epoch 5/5
5/5 [==============================] - 78s 17s/step - loss: 0.2120 - accuracy: 0.9197 - val_loss: 0.0915 - val_accuracy: 0.9927
```

The graph for the loss function is provided below:



Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

The model “facefeatures_new_model_new_4_classes.h5” can be downloaded from the link of the git repository that is mentioned in the report earlier.

8. Contribution by Individual Members

Akash Parida

ID: B420004

Responsibilities:

1. Set up the web cam and the image background
2. Added time-intervals between mode changes of the web cam using number of frames captured and seconds elapsed. Overlaid the student information text on the background image of the web cam
3. Added documentation to the codes and tweaked the code-structure for better readability and uniformity

Priyanshu

ID: B420037

Responsibilities:

1. Setting up and configuring the Firebase project
2. Implementing Firebase database to store attendance data in a structured format by creating and maintaining data models for the Students Attendance.
3. Integrating the Firebase Realtime Database and Storage buckets with the encoding generator to ensure proper storage and retrieval of encoded images and attendance data.

Sailesh Agrawal

ID: B420043

Responsibilities:

1. Implementing the face recognition algorithm for recognizing the face and pushing it on to the database.
2. Developing and implementing an alternative Face Recognition Model based on transfer learning
3. Training and testing the algorithm of alternative model to improve accuracy.
4. Developing the image encoding generator using facial landmarks detector of face_recognition library. The encoding generated was a 128-dimensional encoding of the face.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System

9. References

- What Is FacialRecognition? https://aws.amazon.com/what-is/facial-recognition/#:~:text=It%20works%20by%20identifying%20and,large%20collection%20of%20existing%20images.
- Face Recognition Module https://pypi.org/project/face-recognition/
- Shukla, A., & Shrivastava, R. K. (2019). Automated attendance system using face recognition. International Journal of Recent Technology and Engineering, 8(1C3), 1013-1017.
- Kumari, A., & Patel, V. M. (2021). A review on face recognition based attendance system. International Journal of Advanced Science and Technology, 30(1), 290-300.

Github link: https://github.com/Zad-100/Face_Recognition_Attendance_System