**Problem Statement: -**

An attendance system using facial recognition that automatically recognizes persons and add their entry and creates logs and can also register new persons and add their face data dynamically into the detection information.

**Scope: -**

Can be used for access control for different organizations at entry doors and gates. Can be used at schools, colleges and other educational institutes as well.

**Limitations: -**

Facial recognition is dependent on accuracy, lighting conditions and other such factors. Can be used together with biometrics and other systems for increased security.

The facial recognition accuracy is also affected by the quality of the camera being used.

**Requirements: -**

**Hardware Requirements: -**

* A windows or mac 64 bit OS is required for the installation of Visual Studio C++ Build Tools
* At least 4 GB of ram is required for the proper functioning of a fast detection and retraining of the model
* A camera is required could be inbuilt camera or an external camera that has a appropriate clarity.

**Software Requirements: -**

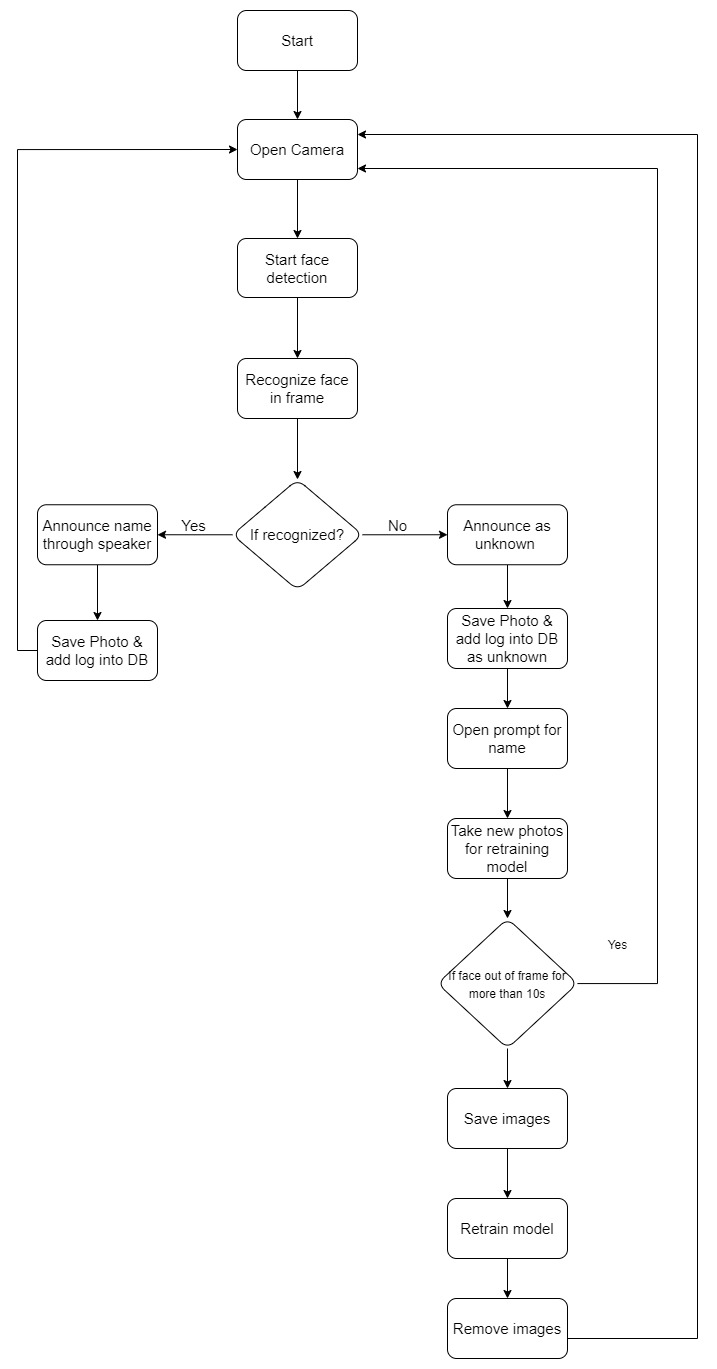
**General Requirements: -**

* The haarcascade.xml file that is the xml file is to be downloaded from the opencv website.
* Microsoft Visual C++ build tools. Desktop development for C++. Required for dlib. cmake installation required for windows

**Python libraries: -**

* Facial-recognition and classification using {cmake (3.25.0v) , dlib (19.24.0v), face-recognition (1.3.0v}
* Camera access and face detection and image manipulation { cv2} (4.6.0.66v)
* Accessing database {mysql-connector } (2.2.9v)
* Text to speech {gtts } (requires internet) (2.3.0v)
* Play the saved mp3 audio {playsound} (1.2.2v)

**Description: -**



The main screen opens the camera and then facial recognition starts immediately. If the detected face is inside the drawn frame, then that frame is sent to the detection.

The recognition module compares the face data received with the already existing data and return the persons name if the faced is matched else it returns the name as unknown.

If the name is detected then the photo log is saved with the filename as “<name>\_<time\_of\_detection>.jpg” and entry is logged into the database as well.

If the person is not detected then the person’s name is saved as unknown into the image logs and database entry then they are shown the prompt to enter the name.

Once the name is entered then the camera opens to take 5 photos to send the images to re-train the model again to add the person into the recognized persons data.

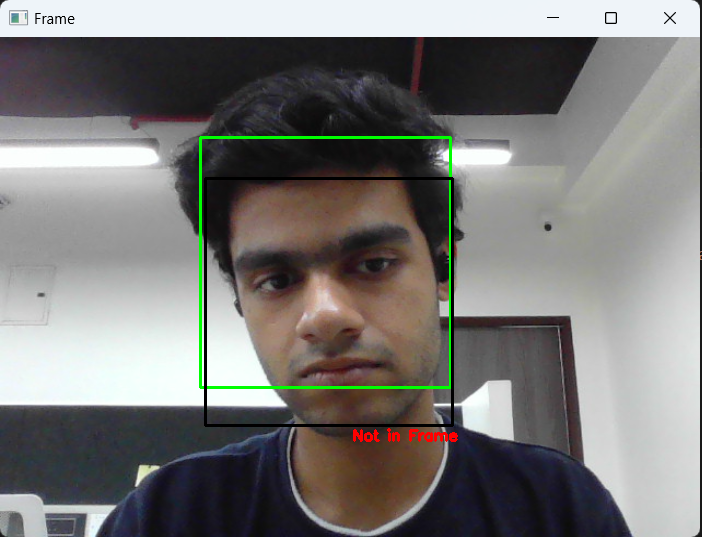
The program detects faces infinitely until explicitly closed.

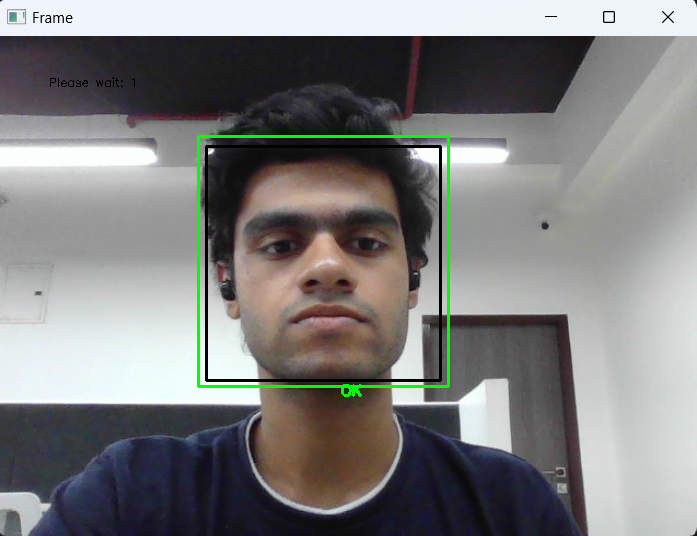
**File description: -**

The files in init\_data are the initial data required which are the haarcascade.xml file, the gui files for confirmation and name entry and the encodings.pickle file and the capturing frame parameters in frame\_box.json.

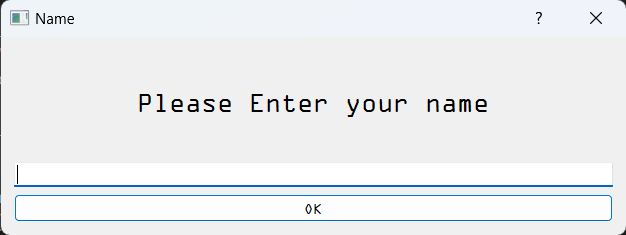
* **haarcasacde.xml :-** The haarcascade.xml is used by the opencv library to detect faces in the images passed through the camera.
* **encodings.pickle :-** The pickle file contains data about the face encodings and their respective names. Every time a new face is detected and the corresponding photos are taken the face\_recognition library compares the encodings of the detected face and the face encodings present in the pickle file and therefore returns the best match and their corresponding name stored in the pickle file. The same file is also updated when a new face is to be registered.
* **frame\_box.json :-** The frame\_box file contains the height and width of the area of rectangle drawn. Any face that inside this box is only sent for detection. This value can be changed based on the output screen of the device.
* **GUI** files are created using the PyQt5 designer and they are imported into the python code. Used for showing popups on confirmation and registration and also input the name of the new registers.

**Screenshots: -**

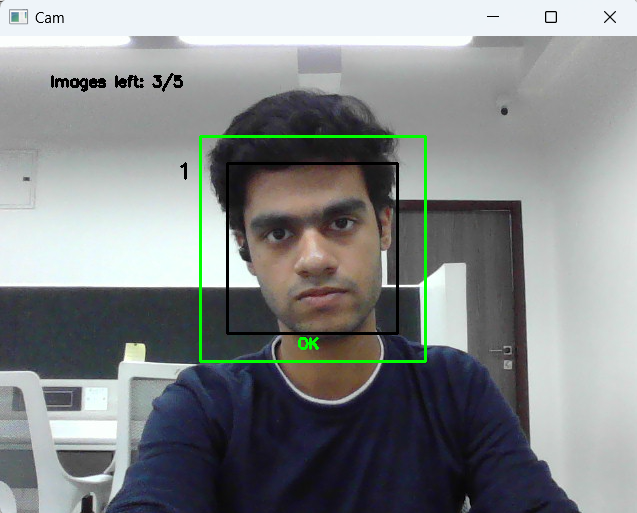
**Faces not in frame not sent for detection**

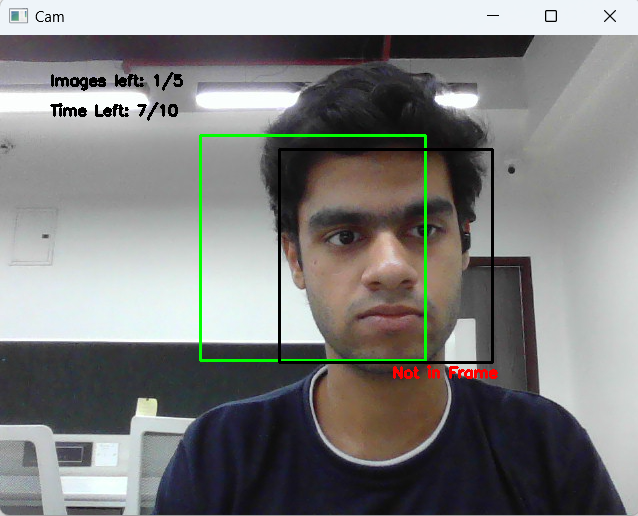


**Face in frame captured as an image and sent for detection**

****

**Name prompt if face not detected**

**Capturing new face for retraining the model**

****

**Timeout starts if face out of frame during retraining**