A Project Based Learning Report

On

Face recognition using GUI

Submitted to CMR Engineering College

In Partial Fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

ASHISH SINGH	**(208R1A0505)**
RAJKUMAR	**(208R1A0506)**
SHRAVAN	**(208R1A0519)**
VINAY KUMAR	**(208R1A0517)**

Under the guidance of

Neha Bajaj

Assistant Professor, Department of CSE

Department of Computer Science & Engineering

CMR ENGINEERING COLLEGE

(Accredited by NBA, Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

2022-2023

CMR ENGINEERING COLLEGE

(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, Hyderabad-501 401

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled **"Face recognition using GUI"** is a bonafide work carried out by

ASHISH SINGH	**(208R1A0505)**
RAJKUMAR	**(208R1A0506)**
SHRAVAN	**(208R1A0519)**
VINAY KUMAR	**(208R1A0517)**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE ANDENGINEERING** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	Head of the Department
Neha Bajaj	**Dr. Sheo Kumar**
Assistant Professor	Professor& HOD
Department of CSE,	Department of CSE,
CMREC, Hyderabad	CMREC, Hyderabad

DECLARATION

This is to certify that the work reported in the present project entitled "**Face recognition using GUI**" is a record of bonafide work done by me in the Department of Computer Science and Engineering, CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by me and not copied from any other source. I submit my project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

ASHISH SINGH	**(208R1A0505)**
RAJKUMAR	**(208R1A0506)**
SHRAVAN	**(208R1A0519)**
VINAY KUMAR	**(208R1A0517)**

CONTENTS

TITLE **PAGE NO**

ABSTRACT

Face recognition is the process of identifying or verifying the identity of an individual using their facial features. With the advancement of technology, face recognition systems have become popular in various fields, including security, surveillance, and biometrics. In this project, we propose a face recognition system using a Graphical User Interface (GUI) that allows for user-friendly interaction.

The system employs a combination of facial feature extraction and machine learning algorithms to recognize faces in real-time. The system begins by capturing a live video feed from a camera and extracting facial features from the video frames. These features are then compared to a database of known faces using a machine learning algorithm, which is trained to recognize facial features and patterns.

The GUI provides an intuitive user interface for the system, allowing users to interact with the system and view the results of the face recognition process. The system displays the recognized face along with the name or ID of the person in real-time.

The proposed system has several potential applications, including security systems for public places, attendance systems for schools or businesses, and access control systems for restricted areas. Overall, the face recognition system using GUI provides a fast, reliable, and user-friendly Solution for face recognition

1.0 INTRODUCTION

About the Project

"Face recognition is a technology that has become increasingly popular in recent years, with applications in security, social media, and more. With the rise of GUI-based software development, it has become easier for developers to create face recognition systems that are user-friendly and accessible. This documentation aims to provide a comprehensive guide to creating a face recognition system using a GUI. The system allows users to detect and recognize faces, as well as perform a variety of related tasks. In this documentation, we will explain how the system works, its main features, and provide step-by-step instructions for installation and usage."

Face recognition is a computer technology that enables the automatic identification and verification of individuals based on their facial features. This technology has gained significant attention in recent years due to its potential applications in various domains, including security, surveillance, marketing, and entertainment.

In this documentation, we present a face recognition system that utilizes a Graphical User Interface (GUI) to enhance its usability and accessibility. The system is designed to be easy to use and can be operated by users with little to no technical expertise.

The primary objective of this system is to accurately identify individuals by comparing their facial features with a database of known faces. To achieve this goal, the system employs a deep learning-based approach, which has demonstrated superior performance compared to traditional methods.

In the following sections, we provide a detailed description of the system's architecture, functionality, and implementation. Additionally, we discuss the various technologies and algorithms used in the development of the system, as well as the performance metrics used to evaluate its effectiveness.

1.1 REQUIREMENTS

1.1.1 Software Requirements

- Operating System : Windows 11/10/8/7 or Linux
- Tool used : Visual Studio code Editore
- Python : python3.9.7

1.1.2 Minimum hardware requirements

- RAM : 4GB
- System Architecture : 64-bit
- Hard Disk Space : 3GB
- Processor : Intel Atom or Intel i3 core

2.0 BACKGROUND

The purpose of creating face recognition technology is to provide an automated and accurate way of identifying individuals by analyzing their facial features. This technology has various applications, including security and surveillance systems, access control, law enforcement, and identification verification in various industries such as banking, healthcare, and government. It can also be used for personal devices such as Smartphone's and laptops to improve their security features. The ultimate goal is to provide a fast and efficient way of recognizing individuals, reducing the need for manual identification, and improving overall security.

The proposed system is a face recognition application that uses a Graphical User Interface (GUI) to interact with the user. The system aims to provide accurate identification of individuals based on their facial features. The system is designed to recognize and match the input image with the images in the database and return the most likely match.

The proposed system uses a machine learning algorithm, specifically the pre-trained Haar Cascade classifier, to detect faces in the input image. Once the faces are detected, the system extracts the facial features, such as the eyes, nose, and mouth, using the dlib library. These features are then used to create a face embedding, which is a mathematical representation of the face. The system uses a pre-trained deep learning model, specifically the FaceNet model, to match the face embedding with the images in the database. The system returns the most likely match, which is displayed to the user through the GUI.

Overall, the proposed system aims to provide a reliable and accurate method of identifying individuals based on their facial features, which can be used in various applications such as security systems, attendance tracking, and personalized services.

Overview of the face recognition system and its applications

1. Introduction to the software tools and libraries used in the development of the system
2. Detailed explanation of the algorithms and techniques used for face detection, feature extraction, and recognition
3. Description of the database used for training and testing the system
4. Implementation details of the face recognition system, including code snippets and screenshots of the GUI
5. Evaluation of the system's performance in terms of accuracy, speed, and robustness
6. Future work and potential improvements for the face recognition system

The above-mentioned scope will help to provide a comprehensive understanding of the face recognition system and its functionalities. It will also enable users to replicate and enhance the system as per their requirements..

What is OpenCV?

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It was initially developed by Intel in 1999 and has since been used extensively in the fields of image and video processing, object detection, and recognition, among others. OpenCV provides developers with a set of tools and functions for image and video analysis and processing, including various algorithms for edge detection, image filtering, object recognition, and motion estimation. It is written in C++, but has bindings for several other programming languages, including Python, Java, and MATLAB. OpenCV is widely used in academic research, commercial applications, and hobby projects due to its robustness, flexibility, and ease of use.

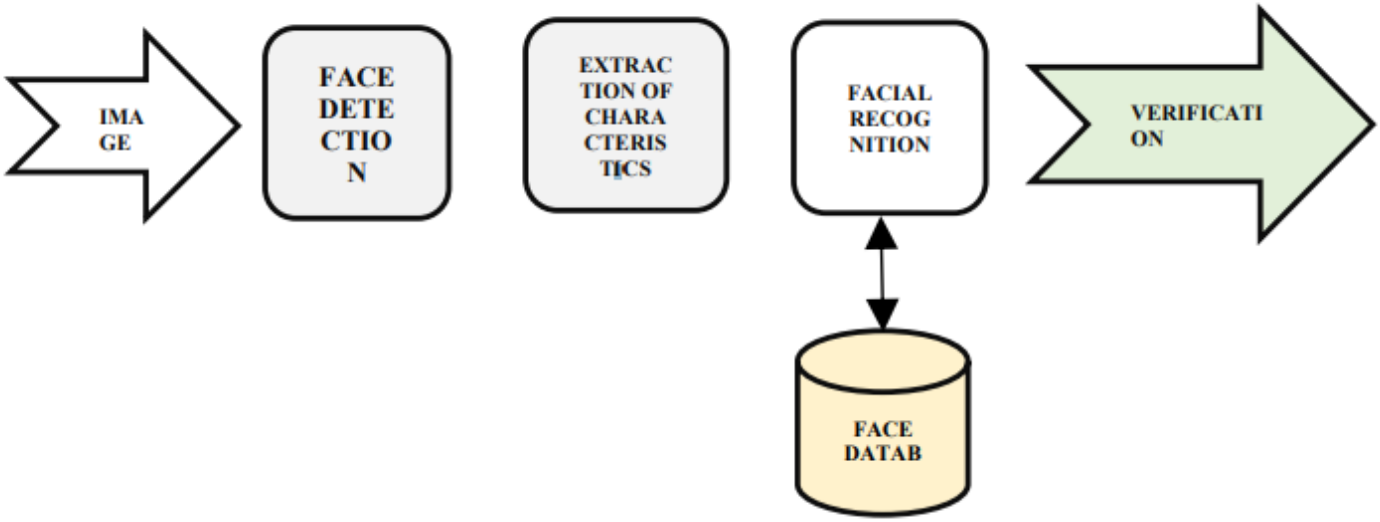
How This Programs Going to Works?

This code *is* written in Python using tkinter and OpenCV library. It is a Face Recognition System which takes images and recognizes faces. The code has a GUI interface with two input fields, one for ID and one for name.

How to Capture Face?

There is a button "Take Images" which is used to take 60 Images of every new user for training the model.

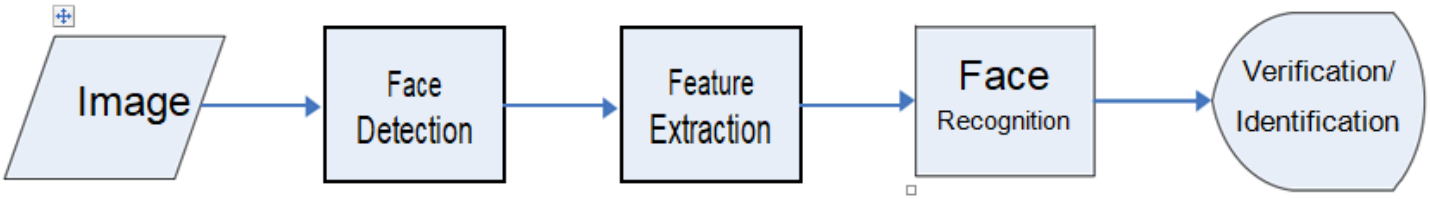
How your program runs. Explain about internal structure?



For this, facial recognition systems depend on models that offer and provve to have low error rates, which characterizes the accuracy of these systems. Accuracy is relevant to ensure the success of system s based on facial recognition technologies, because the closer to the 100% accuracy rate, thhe better is the result of the system

2.1 Structure and Procedure

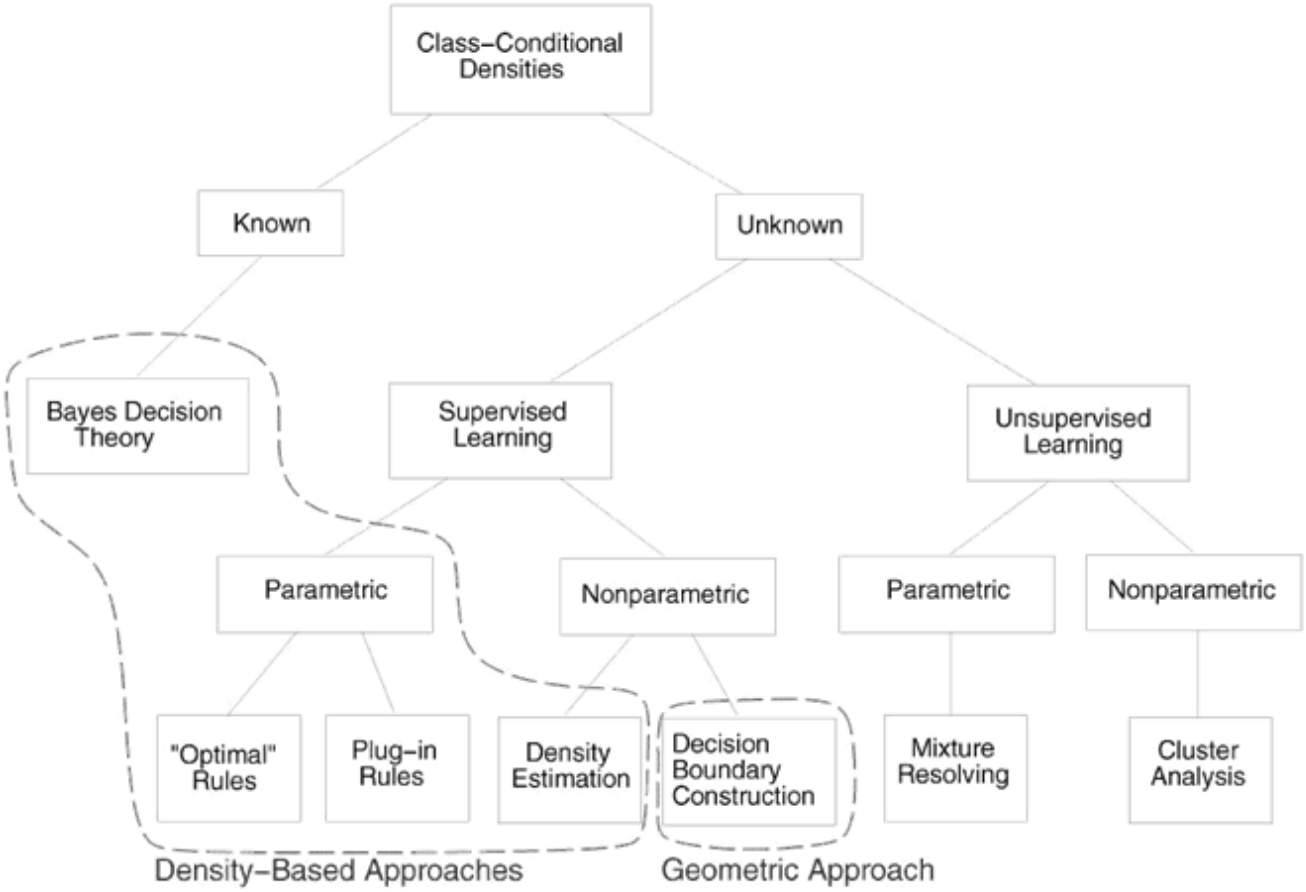
Given a picture taken from a digital camera, we'd like to know if there is any person inside, where his/her face locates at, and who he/she is. Towards this goal, we generally separate the face rec-ognition procedure into three steps: **Face Detection**, **Feature Extraction**, and **Face Recognition** (shown at Fig. 1).



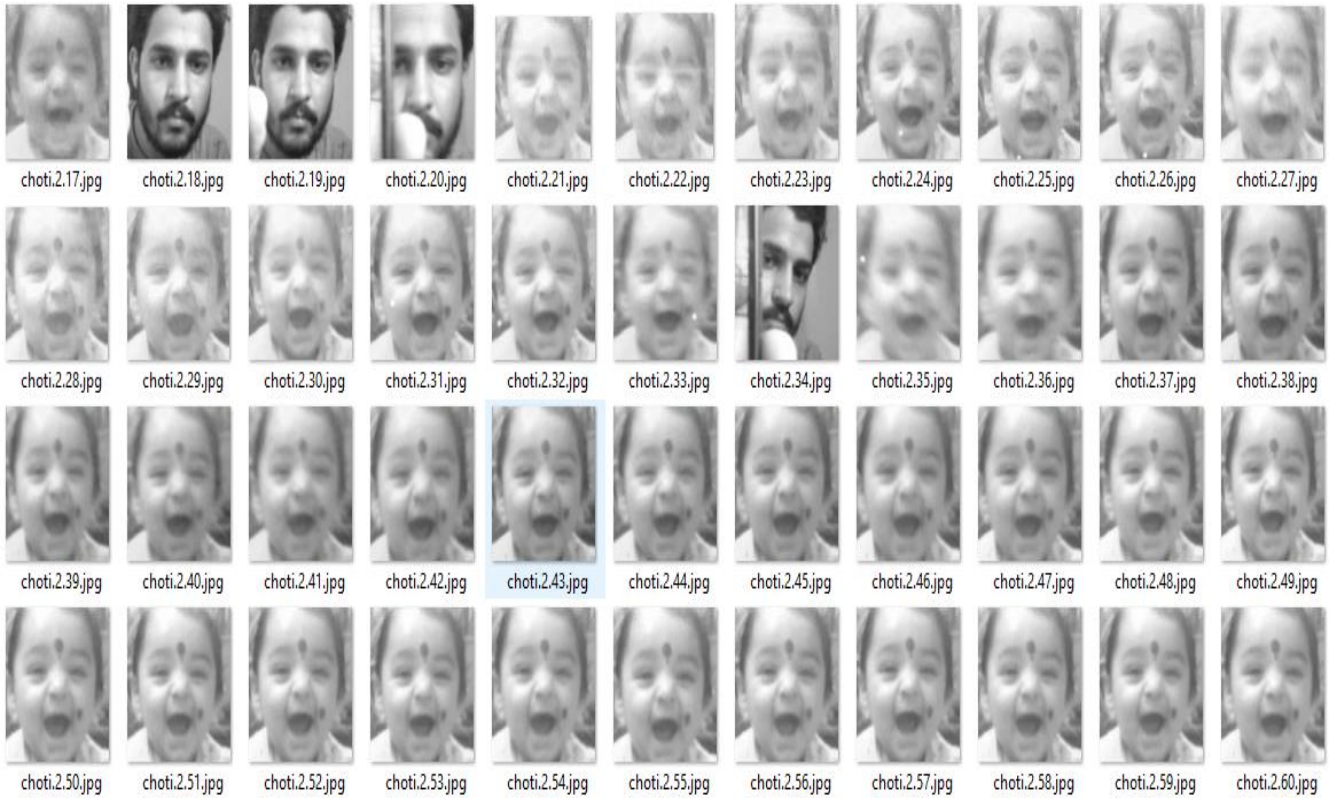
That is a common pipeline for face recognition systems:

1. Image: The input to the system is an image that contains one or more faces.
2. Face Detection: The system uses computer vision algorithms to detect the location of the face or faces within the image. This involves identifying the facial landmarks and features, such as the eyes, nose, and mouth.
3. Feature Extraction: Once the face is detected, the system extracts features from the face. These features can include measurements of the distances between facial landmarks, the shape of the face, and the texture of the skin. These features are then used to create a unique template for each face.
4. Face Recognition: The system compares the template created from the input image to templates in a database of known faces. This involves a similarity calculation, which measures how closely the features in the input template match the features in the database templates.
5. Verification/Identification: Depending on the application, the system will either verify that the input face matches a specific person's face or attempt to identify the person in the input image by searching through a database of known faces. The system will output the result of the verification or identification process, indicating whether a match was found or not.

2.2 Various approaches in pattern recognition



2.3 Training Data



3.0 CODE BASE

We will Create 2 Scripts.

```
# importing libraries

import tkinter as tk
from tkinter import Message, Text
import cv2
import os
import shutil
import csv
import numpy as np
from PIL import Image, ImageTk
import pandas as pd
import datetime
import time
import tkinter.ttk as ttk
import tkinter.font as font
from pathlib import Path
from cv2 import face
# import cv2.face



window = tk.Tk()
window.title("Face_Recogniser")
window.configure(background='white')
window.grid_rowconfigure(0, weight=1)
window.grid_columnconfigure(0, weight=1)
message = tk.Label(
    window, text="Face-Recognition-System",
    bg="green", fg="white", width=50,
    height=3, font=('times', 30, 'bold'))

message.place(x=200, y=20)

lbl = tk.Label(window, text="No.",
            width=20, height=2, fg="green",
            bg="white", font=('times', 15, ' bold '))
lbl.place(x=400, y=200)

txt = tk.Entry(window,
            width=20, bg="white",
            fg="green", font=('times', 15, ' bold '))
txt.place(x=700, y=215)

lbl2 = tk.Label(window, text="Name",
```

```
                width=20, fg="green", bg="white",
                height=2, font=('times', 15, ' bold '))
lbl2.place(x=400, y=300)

txt2 = tk.Entry(window, width=20,
                bg="white", fg="green",
                font=('times', 15, ' bold '))
txt2.place(x=700, y=315)

# The function below is used for checking
# whether the text below is number or not ?


def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass

    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass

    return False
# Take Images is a function used for creating
# the sample of the images which is used for
# training the model. It takes 60 Images of
# every new user.


def TakeImages():

    # Both ID and Name is used for recognising the Image
    Id = (txt.get())
    name = (txt2.get())

    # Checking if the ID is numeric and name is Alphabetical
    if(is_number(Id) and name.isalpha()):
        # Opening the primary camera if you want to access
        # the secondary camera you can mention the number
        # as 1 inside the parenthesis
        cam = cv2.VideoCapture(0)
        # Specifying the path to haarcascade file
        harcascadePath = "data\haarcascade_frontalface_default.xml"
```

```
# Creating the classier based on the haarcascade file.
detector = cv2.CascadeClassifier(harcascadePath)
# Initializing the sample number(No. of images) as 0
sampleNum = 0
while(True):
    # Reading the video captures by camera frame by frame
    ret, img = cam.read()
    # Converting the image into grayscale as most of
    # the processing is done in gray scale format
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # It converts the images in different sizes
    # (decreases by 1.3 times) and 5 specifies the
    # number of times scaling happens
    faces = detector.detectMultiScale(gray, 1.3, 5)

    # For creating a rectangle around the image
    for (x, y, w, h) in faces:
        # Specifying the coordinates of the image as well
        # as color and thickness of the rectangle.
        # incrementing sample number for each image
        cv2.rectangle(img, (x, y), (
            x + w, y + h), (255, 0, 0), 2)
        sampleNum = sampleNum + 1
        # saving the captured face in the dataset folder
        # TrainingImage as the image needs to be trained
        # are saved in this folder
        cv2.imwrite(
            "TrainingImage\ "+name + "."+Id + '.' + str(
                sampleNum) + ".jpg", gray[y:y + h, x:x + w])
        # display the frame that has been captured
        # and drawn rectangle around it.
        cv2.imshow('frame', img)
    # wait for 100 milliseconds
    if cv2.waitKey(100) & 0xFF == ord('q'):
        break
    # break if the sample number is more than 60
    elif sampleNum > 60:
        break
# releasing the resources
cam.release()
# closing all the windows
cv2.destroyAllWindows()
# Displaying message for the user
res = "Images Saved for ID : " + Id + " Name : " + name
# Creating the entry for the user in a csv file
row = [Id, name]
with open(r'UserDetails\UserDetails.csv','a+') as csvFile:
```

```
            writer = csv.writer(csvFile)
            # Entry of the row in csv file
            writer.writerow(row)
        csvFile.close()
        message.configure(text=res)
    else:
        if(is_number(Id)):
            res = "Enter Alphabetical Name"
            message.configure(text=res)
        if(name.isalpha()):
            res = "Enter Numeric Id"
            message.configure(text=res)

# Training the images saved in training image folder


def TrainImages():
    # Local Binary Pattern Histogram is an Face Recognizer
    # algorithm inside OpenCV module used for training the image dataset
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    # Specifying the path for HaarCascade file
    harcascadePath = "data\haarcascade_frontalface_default.xml"
    # creating detector for faces
    detector = cv2.CascadeClassifier(harcascadePath)
    # Saving the detected faces in variables
    faces, Id = getImagesAndLabels("TrainingImage")
    # Saving the trained faces and their respective ID's
    # in a model named as "trainer.yml".
    recognizer.train(faces, np.array(Id))
    recognizer.save("TrainingImageLabel\Trainer.yml")
    # Displaying the message
    res = "Image Trained"
    message.configure(text=res)


def getImagesAndLabels(path):
    # get the path of all the files in the folder
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faces = []
    # creating empty ID list
    Ids = []
    # now looping through all the image paths and loading the
    # Ids and the images saved in the folder
    for imagePath in imagePaths:
        # loading the image and converting it to gray scale
        pilImage = Image.open(imagePath).convert('L')
        # Now we are converting the PIL image into numpy array
        imageNp = np.array(pilImage, 'uint8')
```

```
        # getting the Id from the image
        Id = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(Id)
    return faces, Ids
# For testing phase


def TrackImages():
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    # Reading the trained model
    recognizer.read("TrainingImageLabel\Trainer.yml")
    harcascadePath = "data\haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath)
    # getting the name from "userdetails.csv"
    df = pd.read_csv(r"UserDetails\UserDetails.csv")
    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    while True:
        ret, im = cam.read()
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(gray, 1.2, 5)
        for(x, y, w, h) in faces:
            cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)
            Id, conf = recognizer.predict(gray[y:y + h, x:x + w])
            if(conf < 50):
                # print(df.loc[df['A'] == Id]['Name'].values)
                print(df)
                aa = df.loc[df['Id'] == Id]['Name'].values
                tt = str(Id)+"-"+aa
            else:
                Id = 'Unknown'
                tt = str(Id)
            if(conf > 75):
                noOfFile = len(os.listdir("ImagesUnknown"))+1
                cv2.imwrite("ImagesUnknown\Image" +
                            str(noOfFile) + ".jpg", im[y:y + h, x:x + w])
            cv2.putText(im, str(tt), (x, y + h),
                        font, 1, (255, 255, 255), 2)
        cv2.imshow('im', im)
        if (cv2.waitKey(1) == ord('q')):
            break
    cam.release()
    cv2.destroyAllWindows()


takeImg = tk.Button(window, text="Sample",
```

```
                    command=TakeImages, fg="white", bg="green",
                    width=20, height=3, activebackground="Red",
                    font=('times', 15, ' bold '))
takeImg.place(x=200, y=500)
trainImg = tk.Button(window, text="Training",
                    command=TrainImages, fg="white", bg="green",
                    width=20, height=3, activebackground="Red",
                    font=('times', 15, ' bold '))
trainImg.place(x=500, y=500)
trackImg = tk.Button(window, text="Testing",
                    command=TrackImages, fg="white", bg="green",
                    width=20, height=3, activebackground="Red",
                    font=('times', 15, ' bold '))
trackImg.place(x=800, y=500)
quitWindow = tk.Button(window, text="Quit",
                    command=window.destroy, fg="white", bg="green",
                    width=20, height=3, activebackground="Red",
                    font=('times', 15, ' bold '))
quitWindow.place(x=1100, y=500)
window.mainloop()
```

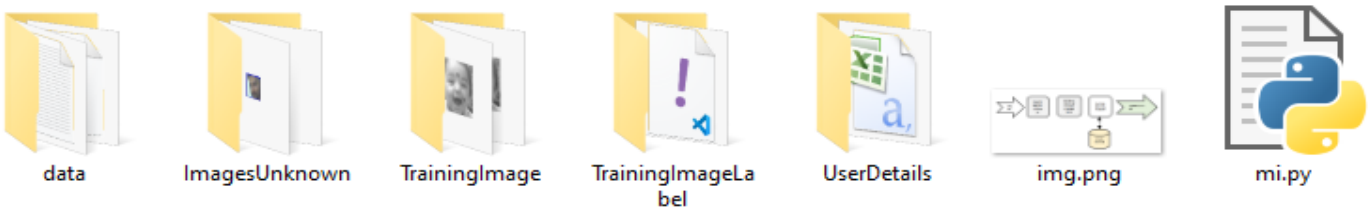
3.1 Working Process of code

1. The code has a GUI interface with two input fields, one for ID and one for name. Then there is a button "Take Images" which is used to take 60 Images of every new user for training the model.
2. The function TakeImages is used for creating the sample of the images which is used for training the model. It takes 60 Images of every new user.
3. The is_number function is used for checking whether the text below is a number or not.
4. The program opens the primary camera and captures video frame by frame. Then, it converts the image into grayscale as most of the processing is done in gray scale format. It then converts the images in different sizes (decreases by 1.3 times) and 5 specifies the number of times scaling happens. Then, it creates a rectangle around the image. Then it specifies the coordinates of the image as well as the color and thickness of the rectangle. Then it saves the captured face in the dataset folder, TrainingImage as the images need to be trained are saved in this folder. Finally, it displays the frame that has been captured and drawn a rectangle around it.

3.2 Libraries to be installed

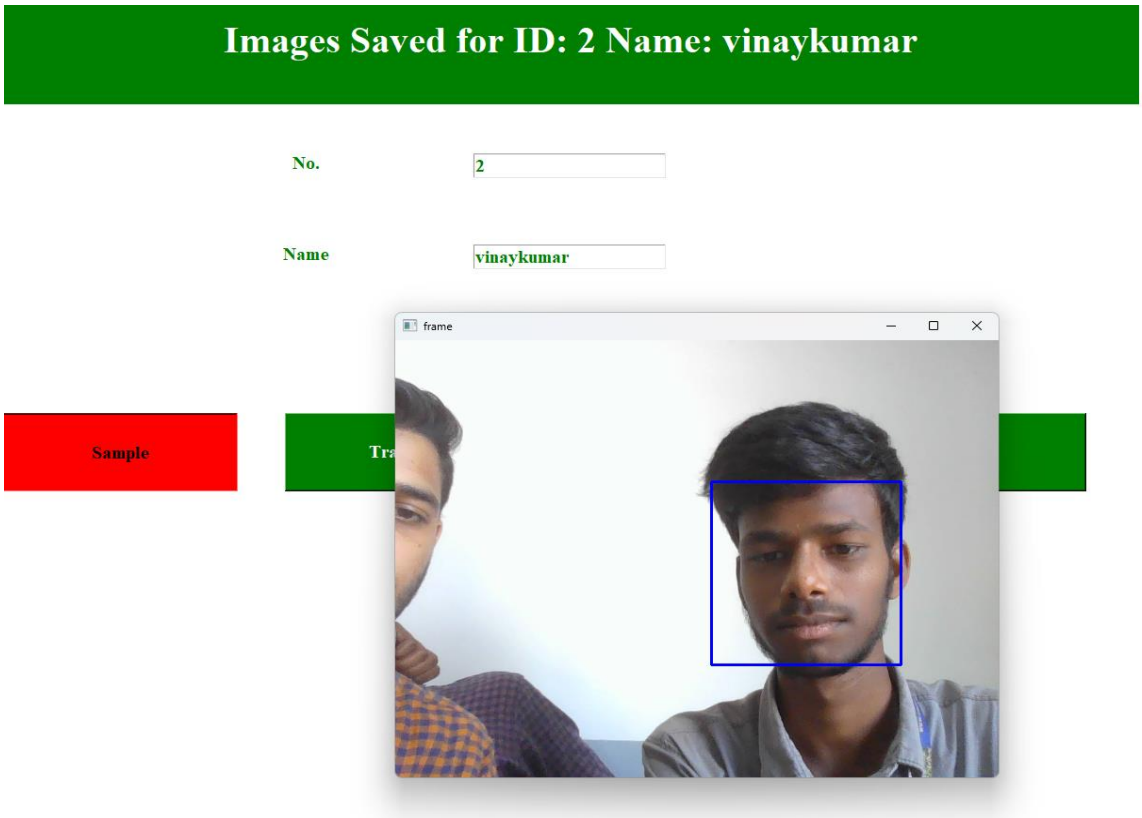
- **tkinter**: For building the GUI (Graphical User Interface).
- **cv2** (OpenCV): For image processing and computer vision tasks.
- **os**, **shutil**, **csv**, **pathlib**: For handling file and directory operations.
- **numpy**: For numerical operations on arrays.
- **PIL (Python Imaging Library)**: For image processing tasks.
- **datetime**, **time**: For time-related operations.
- **ttk**, **font**: For styling the GUI widgets.

3.3 Folder Structure

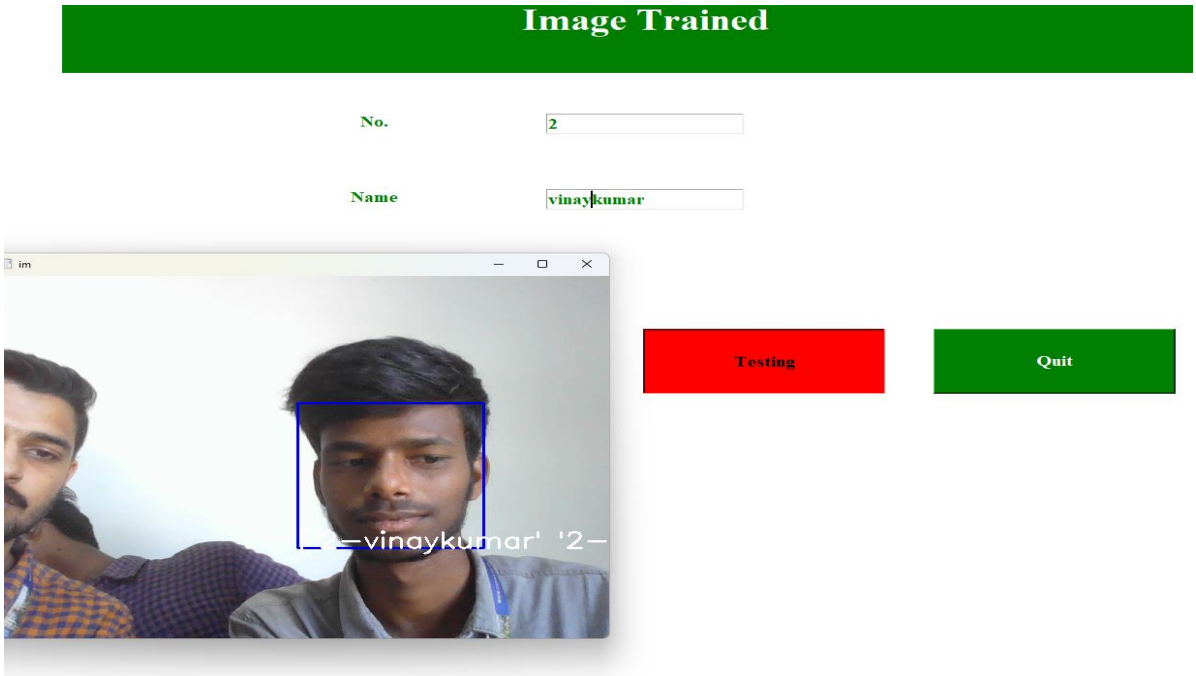


OUTPUT

Sample and Training



Testing the Image



5.0 CONCLUSIONS

Facial recognition is a technology that uses machine learning algorithms to identify and verify people's faces in digital images or videos. It can be used in various fields, including security, marketing, and entertainment.
In security, facial recognition can be used for access control, surveillance, and law enforcement. For example, it can be used to verify the identity of people entering a secure area or to track the movements of criminals.
In marketing, facial recognition can be used for targeted advertising and personalized experiences. For example, a store could use facial recognition to identify a customer and provide personalized recommendations or coupons based on their previous purchases.
In entertainment, facial recognition can be used for games, virtual reality experiences, and even in movies. For example, it can be used to capture facial expressions of actors and transfer them to animated characters in real-time..

6.0 REFERENCES

The following are all the links visited during the preparation of this literature survey:

1. Karpathy, A. (2015). The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/

2. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735– 1780.

   https://doi.org/10.1162/neco.1997.9.8.1735

3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org/

4. Colah, C. (2015). Understanding LSTM Networks.

   http://colah.github.io/posts/2015-08-Understanding-LSTMs/

5. Brownlee, J. (2018). How to Develop a Word-Level Neural Language Model and Use it to Generate Text. Machine Learning Mastery. https://machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-in-keras/