

**A PROJECT REPORT
ON
REAL TIME-BASED FACIAL RECOGNITION SYSTEM**

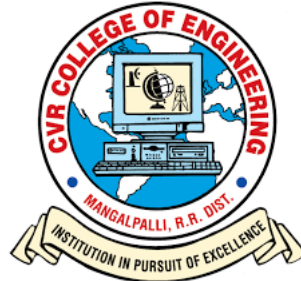
*Mini Project report submitted in the partial fulfillment
of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING**

Submitted by

K.VISHWA SAI	17B81A05V5
D.SUDHEER KUMAR	17B81A05S1
K. VISHAL REDDY	17B81A05V3

Under the guidance of
Ms. Y. SARADA DEVI
Assistant Professor, CSIT Department



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CVR COLLEGE OF ENGINEERING

(UGC Autonomous Institution)

Accredited by NBA & NAAC 'A' Grade

(Approved by AICTE & Govt. of Telangana and Affiliated to JNT University, Hyderabad)
Vastunagar, Mangalpalli (v), Ibrahimpatanam (M), R.R.Dist-501 510.Telangana.
Ph.08414-252222, 252369/79



Cherabuddi Education Society's
CVR COLLEGE OF ENGINEERING

(An Autonomous Institution)

ACCREDITED BY NBA, NAAC 'A' Grade

(Approved by AICTE & Government of Telangana and Affiliated to JNTU Hyderabad)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), R.R.District.

Web: <http://www.cvr.ac.in>, email: info@cvr.ac.in

Ph : 08414 – 252222, 252369, Office Telefax : 252396, Principal : 252396 (O)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled “**Real Time-Based Facial Recognition System**” being submitted by K. Vishwa Sai (17B81A05V5), D. Sudheer Kumar (17B81A05S1), K. Vishal Reddy (17B81A05V3) in partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering to Jawaharlal Nehru Technological University (JNTUH), Hyderabad, is a bonafide work carried out by them under my guidance and supervision. The results provided in this report have not been submitted to any other university or institution for the award of any degree.

Mrs. Y. Sarada Devi

Asst. Professor, CSIT

CVR College of Engineering

Dr. K. Venkateswara Rao

HOD, CSE

CVR College of Engineering

ACKNOWLEDGEMENT

I sincerely thank **Dr. Nayanathara K Sattiraju**, principal, CVR College of Engineering, for her cooperation and encouragement throughout the project.

I earnestly thank **Dr. K. Venkateswara Rao**, HOD, Department of CSE, CVR College of Engineering, for giving timely cooperation and taking necessary action throughout the course of our project.

I express our sincere thanks and gratitude to my project coordinator and internal guide, **Ms. Y. Sarada Devi**, Department of CSE, CVR College of Engineering, for her valuable help and encouragement throughout the project work.

Finally, I place in records my sincere appreciation and indebtedness to my parents and all the lectures for their understanding and co-operation, without whose encouragement and blessing it would not have been possible to complete this work.

K. Vishwa Sai (17B81A05V5)

D. Sudheer Kumar (17B81A05S1)

K. Vishal Reddy (17B81A05V3)

ABSTRACT

Facial recognition is a challenging research in the field of image processing and computer vision, especially for security systems based on the face image recognition. It is also one of the most interesting and important research field in the past two decades. The reason comes from the need of automatic recognition and surveillance systems. Face recognition is an important application of Image processing owing to its use in many fields. The project presented here was developed after study of various face recognition methods and their efficiencies Criminals are widely using Social Media & Social Networks to Commit the Crime and they are being part of cyberspace as well, in this regard. Hence there is a need for some software application which detects faces in real time with minimal error and maximum accuracy. Face recognition is an important application of Image processing owing to its use in many fields. The project presented here was developed after studies of various face recognition methods and their efficiencies.

TABLE OF CONTENTS

No.	Title	Page No.
	Certificate	2
	Acknowledgement	3
	Abstract	4
1	INTRODUCTION	
	1.1 Motivation	7
	1.2 Problem Statement	8
	1.3 Project Objectives	8
2	LITERATURE SURVEY	
	2.1 Existing System	9
	2.2 Limitations of existing system	9
	2.3 Our solution	9
3	SOFTWARE & HARDWARE SPECIFICATIONS	
	3.1 Hardware Requirements	10
	3.2 Software Requirements	10
4	DESIGN	
	4.0 UML Diagrams	11

	4.1 Use Case Diagram	12
	4.2 Sequence Diagram	13
	4.3 Activity Diagram	14
	4.4 System Architecture Diagram	15
5	IMPLEMENTATION & TESTING	
	5.1 Code Implementation	16
	5.2 Screenshots	29
	5.3 Testing & Test cases.	30
6	CONCLUSION & FUTURE SCOPE	
	6.1 Conclusion	31
	6.2 Future Scope	31
7	REFERENCES	32

1. INTRODUCTION

1.1 MOTIVATION

Face recognition has been a sought out after problem of biometrics and it has a variety of applications in modern life. The problems of face recognition attracts researchers working in biometrics, pattern recognition field and computer vision. Several face recognition algorithms are also used in many different applications apart from biometrics, such as video compressions, indexing etc. They can also be used to classify multimedia content, to allow fast and efficient searching for material that is of interest to the user.

An efficient face recognition system can be of great help in forensic sciences, Identification for law enforcement, surveillance, authentication for banking and security system, and giving preferential access to authorized users i.e. access control for secured areas etc. The problem of face recognition has gained even more importance after the recent increase in the terrorism related incidents.

Use of face recognition for authentication also reduces the need of remembering passwords and can provide a much greater security if face recognition is used in combination with other security measures for access control. The cost of the license for an efficient commercial Face recognition system ranges from 30,000 \$ to 150,000 \$ which shows the significant value of the problem.

Though face recognition is considered to be a very crucial authentication system but even after two decades continuous research and evolution of many face recognition algorithms, a truly robust and efficient system that can produce good results in real time and normal conditions is still not available.

1.2 PROBLEM STATEMENT

Criminals are widely using Social Media & Social Networks to Commit the Crime and they are being part of cyber space as well, in this regard. • Desired Solution: The solution should focus on developing an application to capture a photo from video recording and search for the same on official websites / social media websites / internet using an optimized facial recognition algorithm.

1.3 PROJECT OBJECTIVES

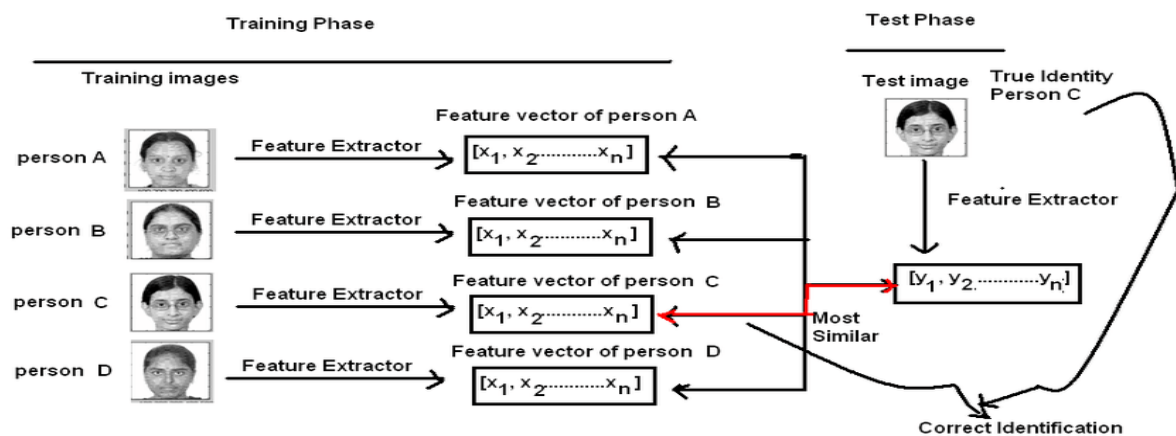
- ☐ To help in detecting criminals in public.
- ☐ To help in detecting anomaly persons in a public area
- ☐ To detect riots occurrence before hand and hence help in preventing them.

2. LITERATURE SURVEY

2.1 EXISTING SYSTEM

Principal component analysis:

Principal component analysis (PCA) is a technique used for identification of a smaller number of uncorrelated variables known as **principal components** from a larger set of data. The technique is widely used to emphasize variation and capture strong patterns in a data set.



2.2 LIMITATIONS OF THE EXISTING SYSTEM

- ☐ **Principal Components** are not as readable and interpretable as original features.
- ☐ Data standardization is must before **PCA**, you must standardize your data before implementing **PCA**, otherwise **PCA** will not be able to find the optimal **Principal Components**.

2.3 OUR SOLUTION

- Desired Solution: The solution should focus on developing an application to capture a photo from a video recording and search for the same on official websites / social media websites / internet using an optimized facial recognition algorithm to easily detect criminals.

3. SOFTWARE AND HARDWARE SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS:

- ☐ **CPU** : Core i5(2.8 GHz or above, at least quad core)
- ☐ **RAM** : 8GB minimum (16 GB recommended)
- ☐ **GPU** : NVidia graphics (minimum 3GB or more like 8GB)
(Like GeForceGTX 1080Ti)

3.2 SOFTWARE REQUIREMENTS:

- ☐ **Language** : Python
- ☐ **Frameworks** : Tensor Flow(Backend), Keras(front end)
- ☐ **Library used** : NumPy, Matplotlib, OS, pandas etc

4. DESIGN

4.0 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: A Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

4.1USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed

for which actor. Roles of the actors in the system can be depicted

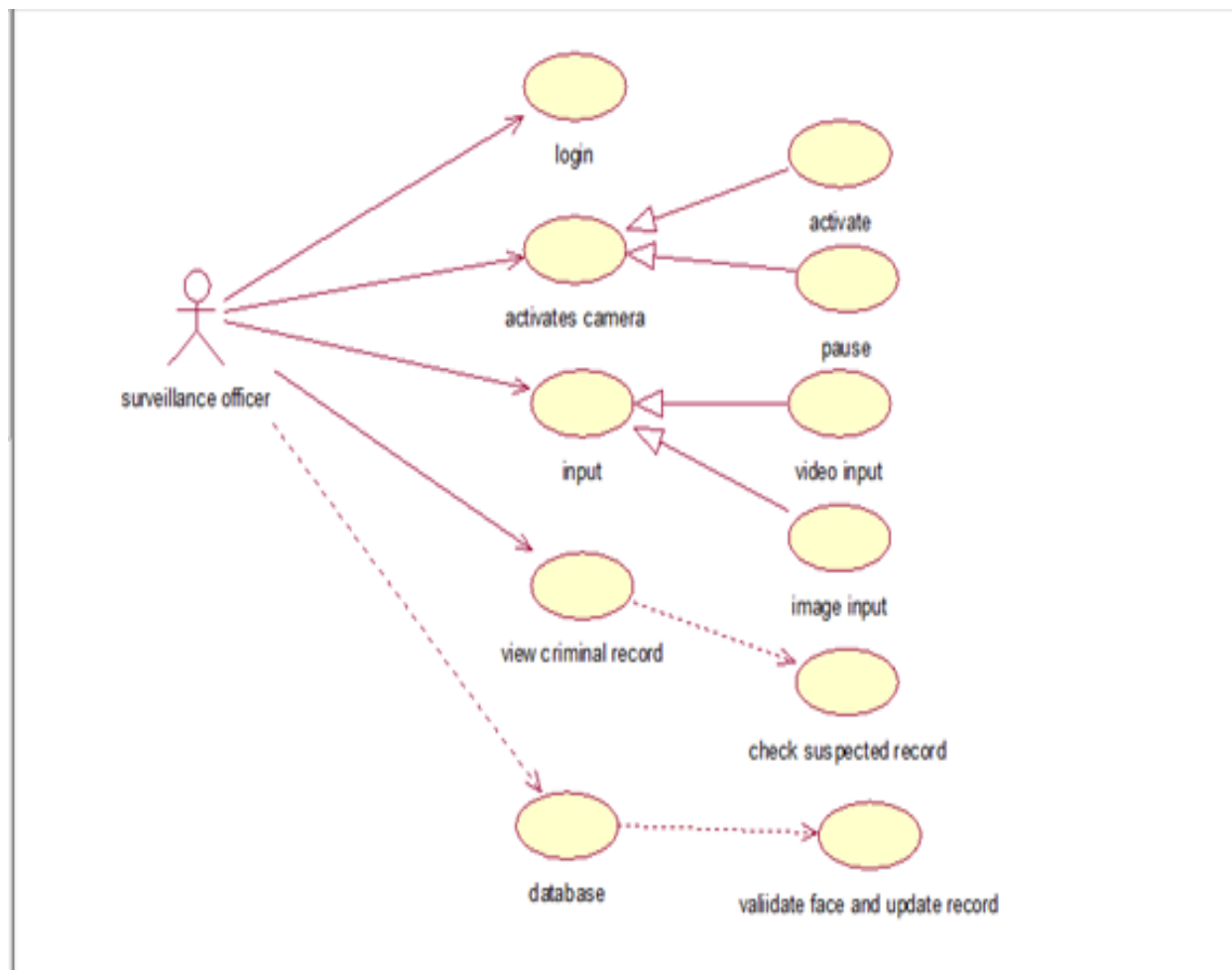


Figure 4.1 Use Case Diagram of Real Time Based Facial Recognition System

4.2 SEQUENCE DIAGRAM:

-

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a

Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

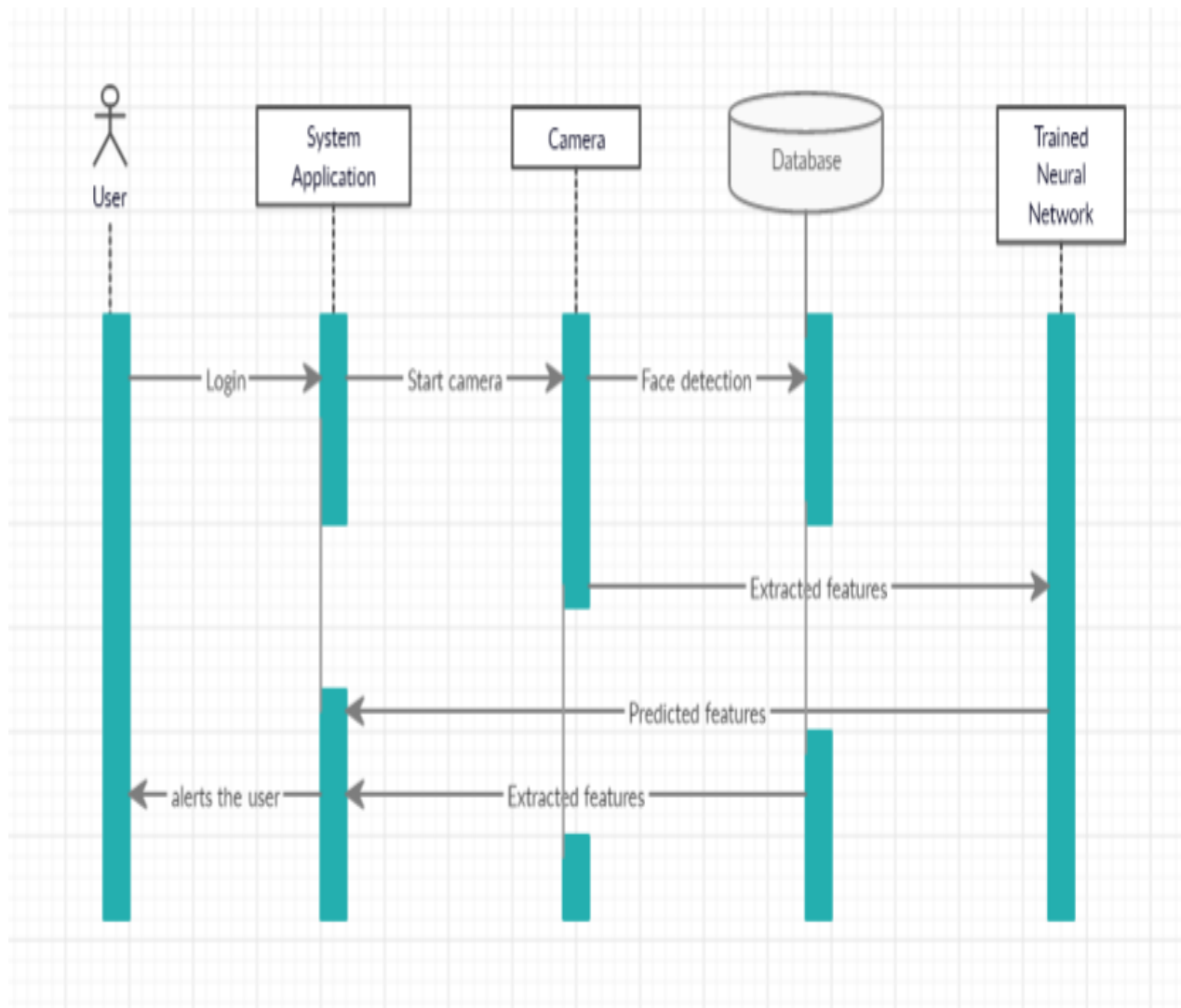


Figure 4.2 Sequence Diagram of Real-Time Based Facial recognition System

4.3 ACTIVITY DIAGRAM:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another as shown in below.

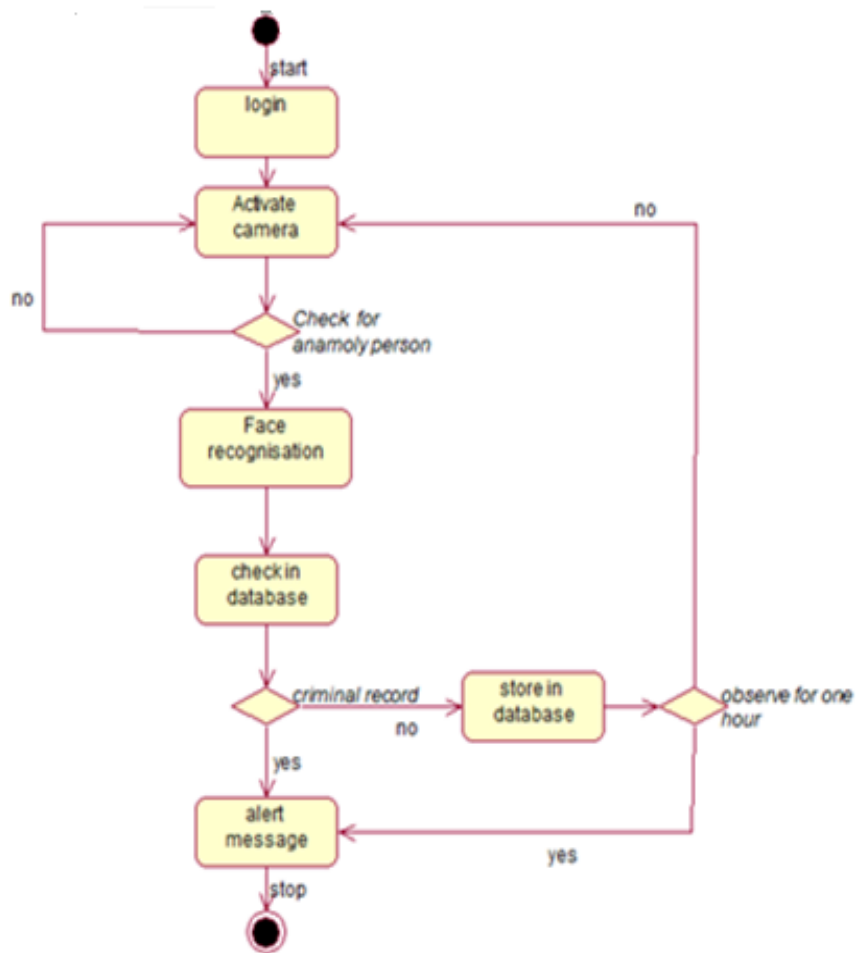


Figure 4.3 Activity Diagram Of real-time based facial recognition system

4.5 SYSTEM ARCHITECTURE DIAGRAM:

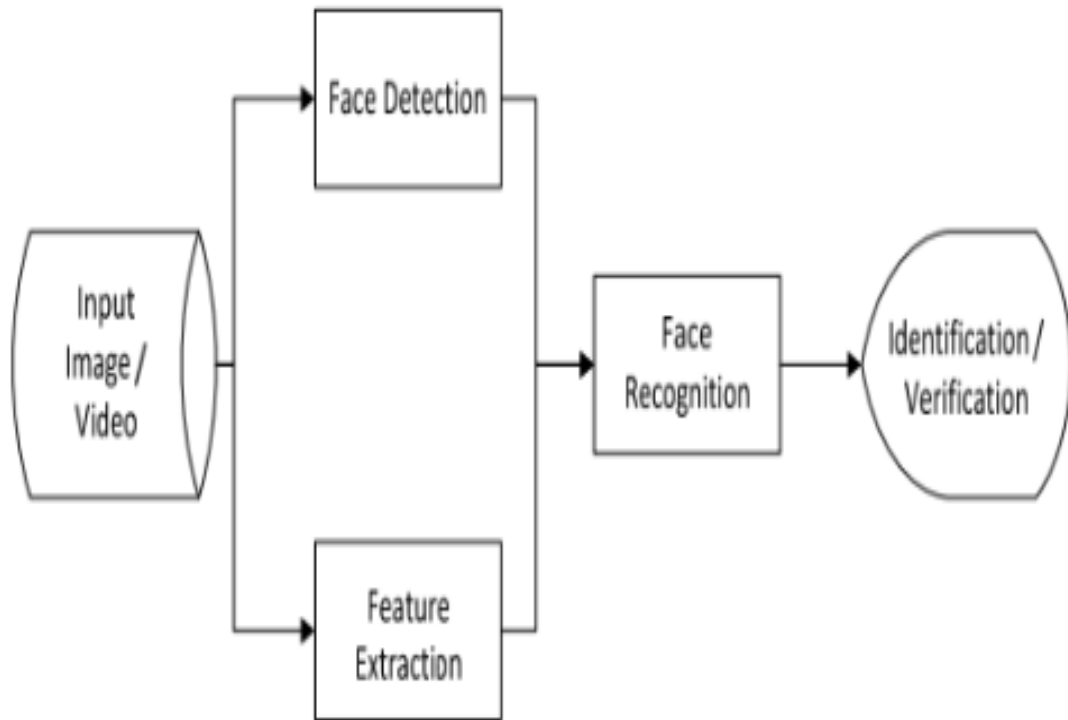


Figure 4.5 System Architecture Diagram of Real-Time Based Facial Recognition System

5. IMPLEMENTATION & TESTING

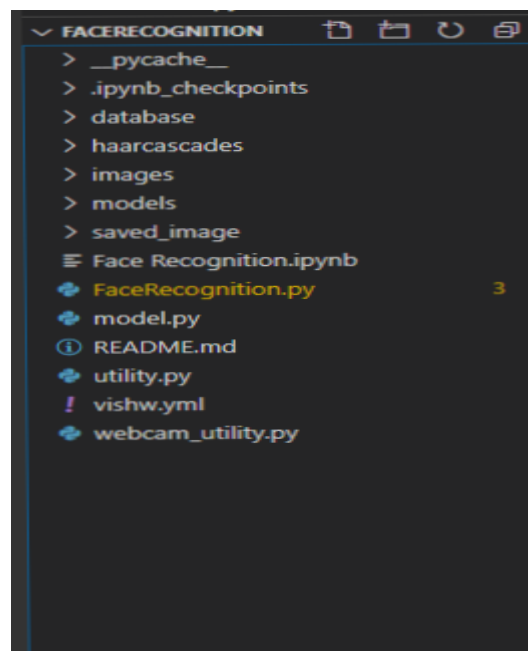
5.1 CODE IMPLEMENTATION

The goal of the coding or programming phase is to translate the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and that performs the computation specified by the design.

The coding phase affects both testing and maintenance. The goal of coding is not to reduce the implementation cost but the goal should be to reduce the cost of later phases. In other words, the goal is not to simplify the job of programmer. Rather the goal should be to simplify the job of the tester and maintainer

Code of the program

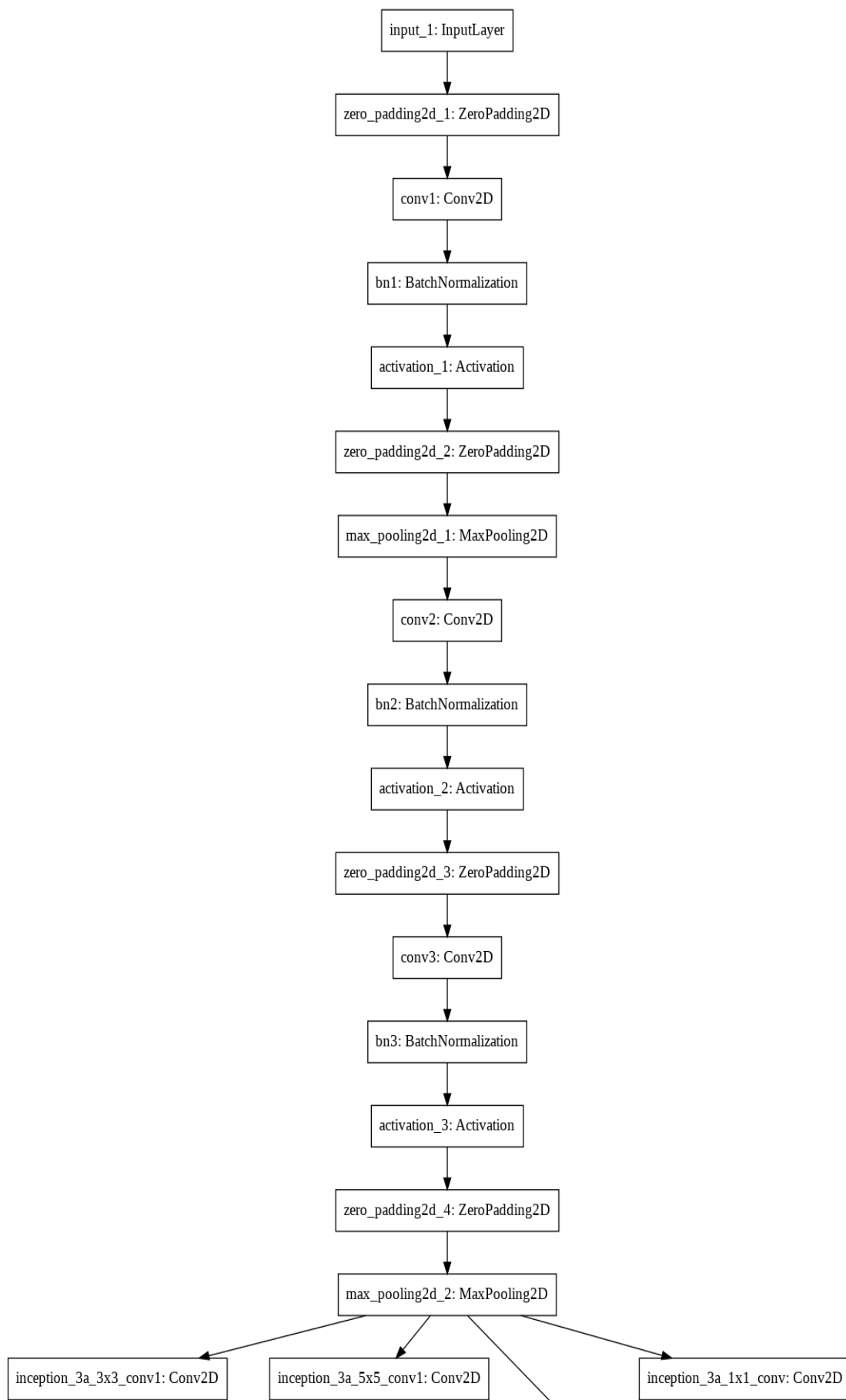
The code of the application is designed in the directory format as shown below

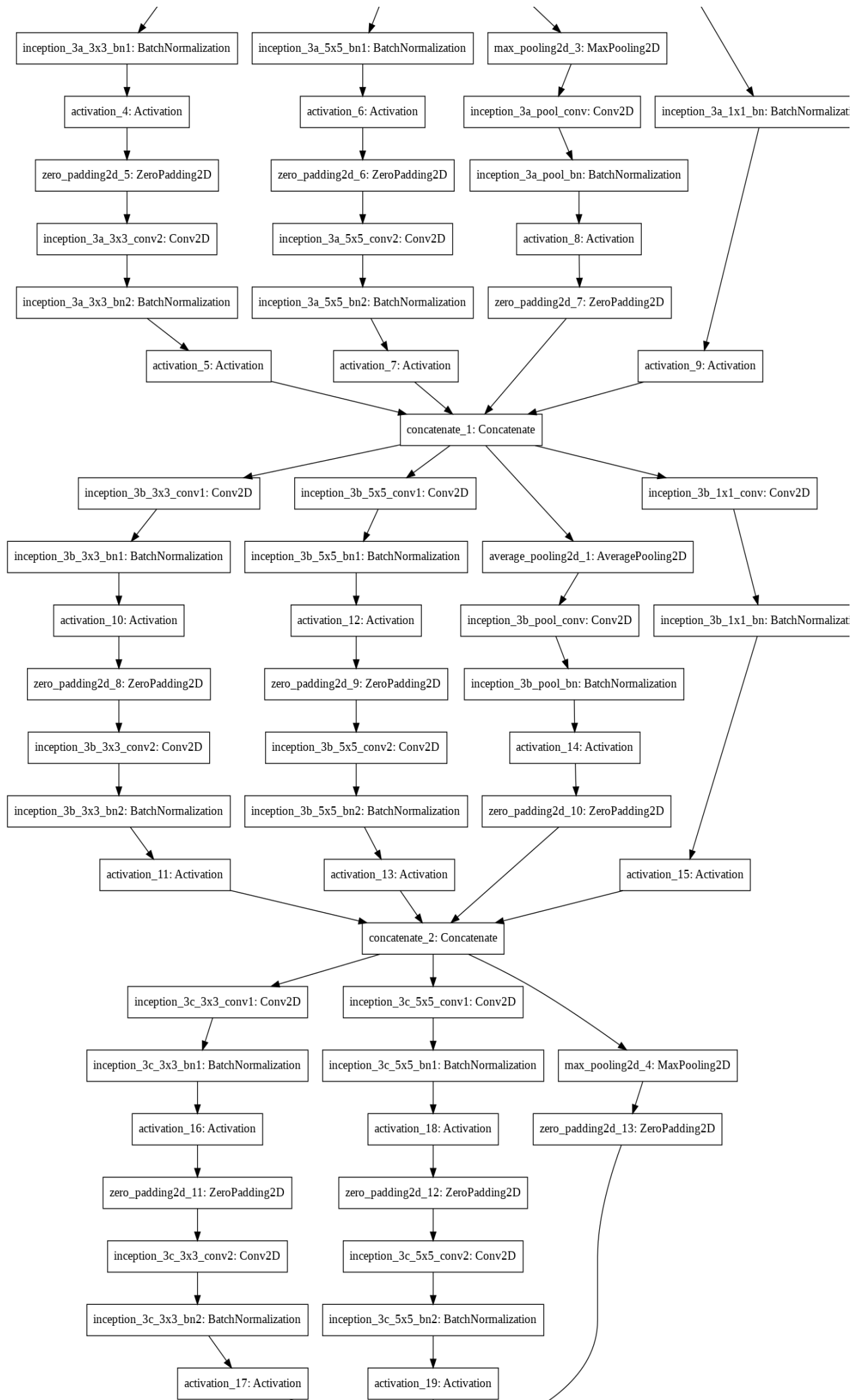


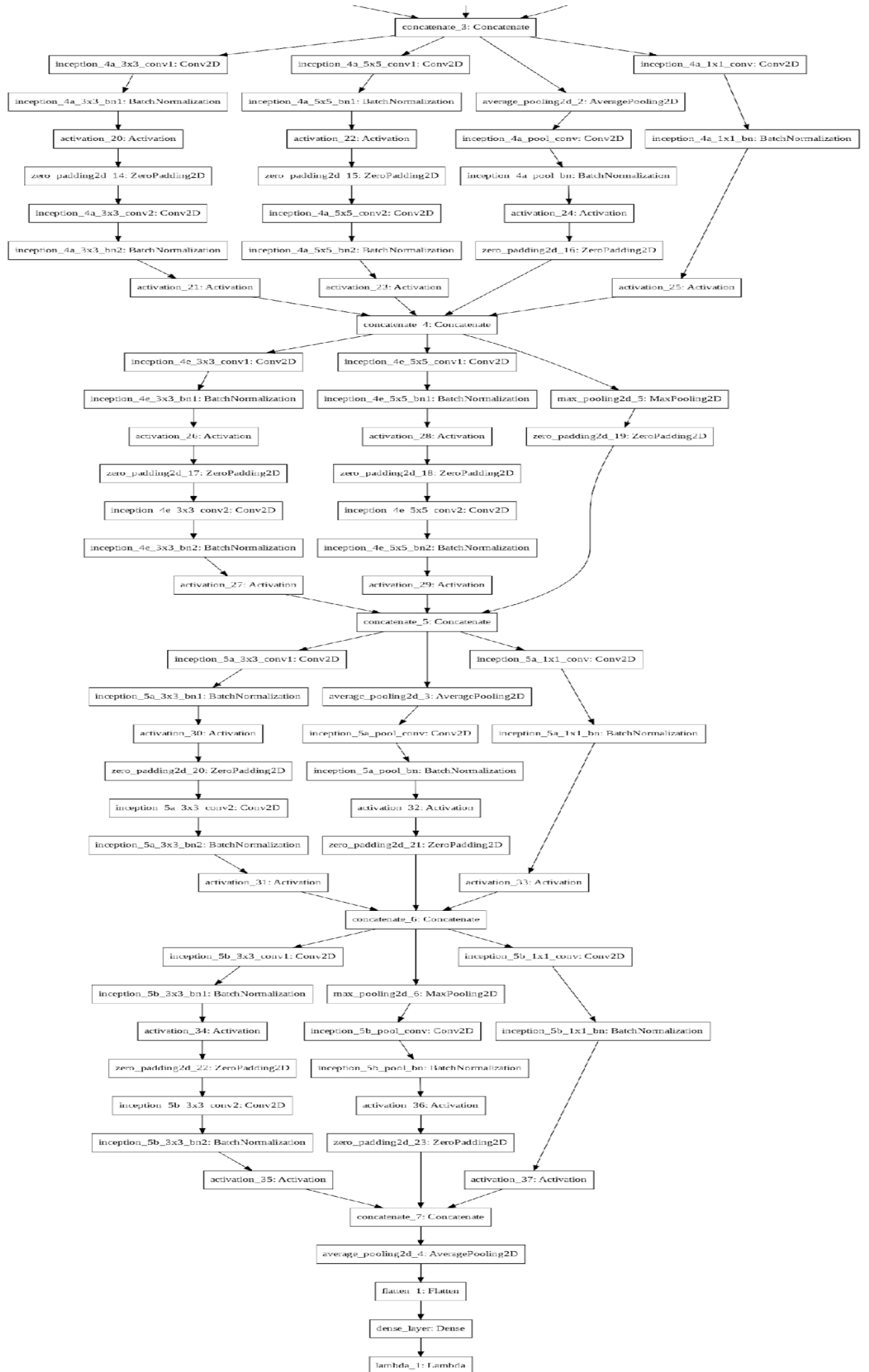
As you can see our application has many folders such as database, model, haarcascades, images.

The database folder has the pickled file which contains the faces which are already recognized by the system. The model folder has the pertained face net model file.

The model architecture of the pertained model is shown below







The application as the architecture diagram show has 2 phases .The first is face detection .This is done using Haar cascade file in haar cascade folder and then verification is done by the code which we have written

The code of the program

a) The application.py (the main file)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, ZeroPadding2D, Activation, Input, concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import MaxPooling2D, AveragePooling2D
from tensorflow.keras.layers import Concatenate
from tensorflow.keras.layers import Lambda, Flatten, Dense
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.layers import Layer
from tensorflow.keras import backend as K
from tensorflow.keras.models import load_model
K.set_image_data_format('channels_first')

import pickle
import cv2
import os.path
import os
import numpy as np
from numpy import genfromtxt
import pandas as pd
import tensorflow as tf
from utility import *
from webcam_utility import *

# ## Model
# The model makes an encoding vector consisting of 128 numbers for the input image. Two encodings are compared and if the two encodings are similar then we say that the two images are of the same person otherwise they are different.
# The model uses **Triplet loss function**. The aim is to minimize this function.

# triplet loss function
# y_pred - list containing three objects:
#     anchor(None, 128) -- encodings for the anchor images
#     positive(None, 128) -- encodings for the positive images
#     negative(None, 128) -- encodings for the negative images
def triplet_loss(y_true, y_pred, alpha = 0.2):
    anchor, positive, negative = y_pred[0], y_pred[1], y_pred[2]

    # triplet formula components
    pos_dist = tf.reduce_sum( tf.square(tf.subtract(anchor, positive)) )
    neg_dist = tf.reduce_sum( tf.square(tf.subtract(anchor, negative)) )
    basic_loss = pos_dist - neg_dist + alpha
```

```

loss = tf.maximum(basic_loss, 0.0)

return loss

# ### Loading the Model
# The model outputs a vector of 128 numbers which represent encoding for the given input i
mage. We will be using this encoding vector for comparing two images.
# ##### Input
# -
This network takes as input 96x96 RGB image as its input. Specifically, inputs a tensor of sh
ape $(m, n\_C, n\_H, n\_W)$ , where $n\_C$ = channel.
#
# ##### Output
# -
A matrix of shape  $*(m, 128)*$  where the 128 numbers are the encoding values for $ith$ im
age.
# load the model
def load_FRmodel():
    FRmodel = load_model('models/model.h5', custom_objects={'triplet_loss': triplet_loss})
    return FRmodel

# We will create a database of registered. For this we will use a simple dictionary and map ea
ch registered user with his/her face encoding.
# initialize the user database
# initialize the user database
def ini_user_database():
    # check for existing database
    if os.path.exists('database/user_dict.pickle'):
        with open('database/user_dict.pickle', 'rb') as handle:
            user_db = pickle.load(handle)
    else:
        # make a new one
        # we use a dict for keeping track of mapping of each person with his/her face encoding
        user_db = {}
        # create the directory for saving the db pickle file
        os.makedirs('database')
        with open('database/user_dict.pickle', 'wb') as handle:
            pickle.dump(user_db, handle, protocol=pickle.HIGHEST_PROTOCOL)
    return user_db

# adds a new user face to the database using his/her image stored on disk using the image pat
h
def add_user_img_path(user_db, FRmodel, name, img_path):
    if name not in user_db:
        user_db[name] = img_to_encoding(img_path, FRmodel)
        # save the database
        with open('database/user_dict.pickle', 'wb') as handle:
            pickle.dump(user_db, handle, protocol=pickle.HIGHEST_PROTOCOL)
        print('User ' + name + ' added successfully')

```

```

else:
    print('The name is already registered! Try a different name.....')

# adds a new user using image taken from webcam
def add_user_webcam(user_db, FRmodel, name):
    # we can use the webcam to capture the user image then get it recognized
    face_found = detect_face(user_db, FRmodel)

    if face_found:
        #resize_img("saved_image/1.jpg","images")
        if name not in user_db:
            add_user_img_path(user_db, FRmodel, name, "saved_image/1.jpg")
        else:
            print('The name is already registered! Try a different name.....')
    else:
        print('There was no face found in the visible frame. Try again.....')

# deletes a registered user from database
def delete_user(user_db, name):
    popped = user_db.pop(name, None)

    if popped is not None:
        print('User ' + name + ' deleted successfully')
        # save the database
        with open('database/user_dict.pickle', 'wb') as handle:
            pickle.dump(user_db, handle, protocol=pickle.HIGHEST_PROTOCOL)
    elif popped == None:
        print('No such user !!')

# ### Putting everything together
# For making this face recognition system we are going to take the input image, find its encoding and then
# see if there is any similar encoding in the database or not. We define a threshold value to decide whether the two images are similar
# or not based on the similarity of their encodings.
def find_face(image_path, database, model, threshold=0.6):
    # find the face encodings for the input image
    encoding = img_to_encoding(image_path, model)

    min_dist = 99999
    # loop over all the recorded encodings in database
    for name in database:
        # find the similarity between the input encodings and claimed person's encodings using L2 norm
        dist = np.linalg.norm(np.subtract(database[name], encoding))
        # check if minimum distance or not
        if dist < min_dist:
            min_dist = dist
            identity = name

```

```

if min_dist > threshold:
    print("User not in the database.")
    identity = 'Unknown Person'
else:
    print("Hi! " + str(identity) + ", L2 distance: " + str(min_dist))

return min_dist, identity

# for doing face recognition
def do_face_recognition(user_db, FRmodel, threshold=0.7, save_loc="saved_image/1.jpg"):
    # we can use the webcam to capture the user image then get it recognized
    face_found = detect_face(user_db, FRmodel)

    if face_found:
        resize_img("saved_image/1.jpg", "images")
        find_face("saved_image/1.jpg", user_db, FRmodel, threshold)
    else:
        print('There was no face found in the visible frame. Try again.....')

def main():
    FRmodel = load_FRmodel()
    print('\n\nModel loaded...')

    user_db = ini_user_database()
    print('User database loaded')

    ch = 'y'
    while(ch == 'y' or ch == 'Y'):
        user_input = input(
            '\nEnter choice \n1. Realtime Face Recognition\n2. Recognize face\n3. Add or Delete\n4. Quit\n')

        if user_input == '1':
            os.system('cls' if os.name == 'nt' else 'clear')
            detect_face_realtime(user_db, FRmodel, threshold=0.6)

        elif user_input == '2':
            os.system('cls' if os.name == 'nt' else 'clear')
            # we can use the webcam to capture the user image then get it recognized
            do_face_recognition(user_db, FRmodel, threshold=0.6,
                               save_loc="saved_image/1.jpg")

        elif user_input == '3':
            os.system('cls' if os.name == 'nt' else 'clear')
            print(
                '1. Add user using saved image path\n2. Add user using Webcam\n3. Delete user\n'
            )

```

```

add_ch = input()
name = input('Enter the name of the person\n')

if add_ch == '1':
    img_path = input(
        'Enter the image name with extension stored in images/\n')
    add_user_img_path(user_db, FRmodel, name, 'images/' + img_path)
elif add_ch == '2':
    add_user_webcam(user_db, FRmodel, name)
elif add_ch == '3':
    delete_user(user_db, name)
else:
    print('Invalid choice...\n')

elif user_input == '4':
    return

else:
    print('Invalid choice...\nTry again?\n')

ch = input('Continue ? y or n\n')
# clear the screen
os.system('cls' if os.name == 'nt' else 'clear')

if __name__ == '__main__':
    main()

```

2) Utility files

a) Utility.py

```

import tensorflow as tf
import numpy as np
import os
import cv2
from numpy import genfromtxt
from tensorflow.keras.layers import Conv2D, ZeroPadding2D, Activation, Input, concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import MaxPooling2D, AveragePooling2D
import h5py

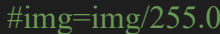
def img_to_encoding(image_path, model):
    img1 = cv2.imread(image_path, 1)
    # resize the image to 96 x 96
    img1 = cv2.resize(img1, (96, 96))
    print(img1.shape)

```



```

img = img1[...::-1]
print(img.shape)

img = np.around(np.transpose(img, (2,0,1))/255.0, decimals=12)
img=img/255.0
print(img.shape)
x_train = np.array([img])
print(x_train.shape)
embedding = model.predict_on_batch(x_train)
return embedding

# loads and resizes an image
def resize_img(image_path, save_path):
    img = cv2.imread(image_path, 1)
    img = cv2.resize(img, (96, 96))
    cv2.imwrite(image_path, img)

```

b) Webcam_utility.py

```

# for taking images from webcam
import cv2
import time
from utility import *
import os.path

def detect_face(database, model):
    save_loc = r'saved_image/1.jpg'
    capture_obj = cv2.VideoCapture(0)
    capture_obj.set(3, 640) # WIDTH
    capture_obj.set(4, 480) # HEIGHT

    face_cascade = cv2.CascadeClassifier(
        r'haarcascades/haarcascade_frontalface_default.xml')

    # whether there was any face found or not
    face_found = False

    # run the webcam for given seconds
    req_sec = 3
    loop_start = time.time()
    elapsed = 0

    while(True):
        curr_time = time.time()
        elapsed = curr_time - loop_start
        if elapsed >= req_sec:
            break

    # capture object frame-by-frame

```

```

ret, frame = capture_obj.read()
# mirror the frame
frame = cv2.flip(frame, 1, 0)

# Our operations on the frame come here
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
# detect face
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

# Display the resulting frame
for (x, y, w, h) in faces:
    # required region for the face
    roi_color = frame[y-90:y+h+70, x-50:x+w+50]
    # save the detected face
    cv2.imwrite(save_loc, roi_color)
    # draw a rectangle bounding the face
    cv2.rectangle(frame, (x-10, y-70),
                  (x+w+20, y+h+40), (15, 175, 61), 4)

# display the frame with bounding rectangle
cv2.imshow('frame', frame)

# close the webcam when 'q' is pressed
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

# release the capture_object
capture_obj.release()
cv2.destroyAllWindows()

img = cv2.imread(save_loc)
if img is not None:
    face_found = True
else:
    face_found = False

return face_found

# detects faces in realtime from webcam feed

def detect_face_realtime(database, model, threshold=0.7):
    text = "
    font = cv2.FONT_HERSHEY_SIMPLEX
    save_loc = r'saved_image/1.jpg'
    capture_obj = cv2.VideoCapture(0)
    capture_obj.set(3, 640) # WIDTH
    capture_obj.set(4, 480) # HEIGHT

    face_cascade = cv2.CascadeClassifier(

```

```

r'haarcascades/haarcascade_frontalface_default.xml')
print('***** Enter "q" to quit *****')
prev_time = time.time()
while(True):

    # capture_object frame-by-frame
    ret, frame = capture_obj.read()
    # mirror the frame
    frame = cv2.flip(frame, 1, 0)

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # detect face
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    # Display the resulting frame
    for (x, y, w, h) in faces:
        # required region for the face
        roi_color = frame[y-90:y+h+70, x-50:x+w+50]

        # save the detected face
        cv2.imwrite(save_loc, roi_color)

        # keeps track of waiting time for face recognition
        curr_time = time.time()

        if curr_time - prev_time >= 3:
            img = cv2.imread(save_loc)
            if img is not None:
                resize_img(save_loc)

            min_dist, identity, registered = find_face_realttime(
                save_loc, database, model, threshold)

            if min_dist <= threshold and registered:
                # for putting text overlay on webcam feed
                text = 'Welcome ' + identity
                print('Welcome ' + identity + '!')
            else:
                text = 'Unknown user'
                print('Unknown user' + ' detected !')
                print('distance:' + str(min_dist))
            # save the time when the last face recognition task was done
            prev_time = time.time()

        # draw a rectangle bounding the face
        cv2.rectangle(frame, (x-10, y-70),
                      (x+w+20, y+h+40), (15, 175, 61), 4)
        cv2.putText(frame, text, (50, 50), font, 1.8, (158, 11, 40), 3)

    # display the frame with bounding rectangle

```

```

cv2.imshow('frame', frame)

# close the webcam when 'q' is pressed
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

# release the capture_object
capture_obj.release()
cv2.destroyAllWindows()

# checks whether the input face is a registered user or not
def find_face_realtime(image_path, database, model, threshold):
    # find the face encodings for the input image
    encoding = img_to_encoding(image_path, model)
    registered = False
    min_dist = 99999
    identity = 'Unknown Person'
    # loop over all the recorded encodings in database
    for name in database:
        # find the similarity between the input encodings and claimed person's encodings using
        # L2 norm
        dist = np.linalg.norm(np.subtract(database[name], encoding))
        # check if minimum distance or not
        if dist < min_dist:
            min_dist = dist
            identity = name

    if min_dist > threshold:
        registered = False
    else:
        registered = True
    return min_dist, identity, registered

```

5.2 SCREENSHOTS

The opening of the application looks like

```
Model loaded...
User database loaded

Enter choice
1. Realtime Face Recognition
2. Recognize face
3. Add or Delete user
4. Quit
```

Now the User chooses which ever option he likes..

Here we want to demo the user presses 3(Add or Delete user)

```
1. Add user using saved image path
2. Add user using Webcam
3. Delete user
```

5.2 TESTING & TESTCASES

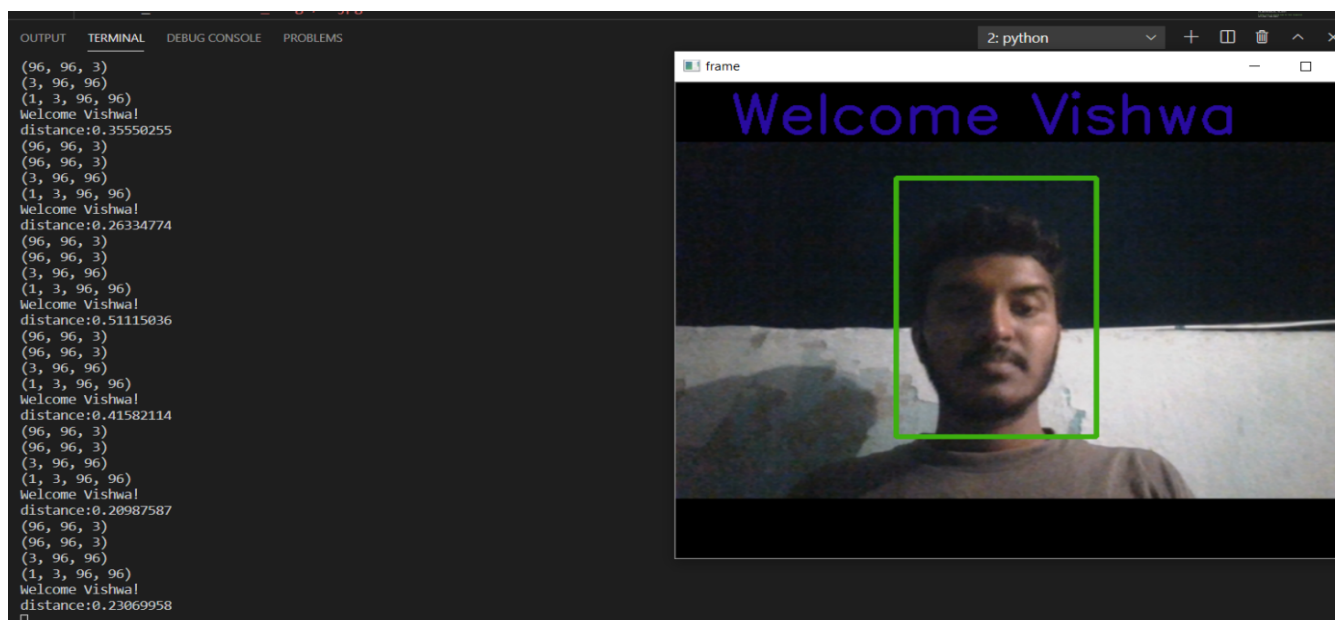
Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

```
AVX2
2020-06-29 23:52:11.633210: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with properties:
name: GeForce GTX 1050 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.62
pciBusID: 0000:01:00:0
totalMemory: 4.00GiB freeMemory: 3.30GiB
2020-06-29 23:52:11.651310: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2020-06-29 23:52:13.046001: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-06-29 23:52:13.057902: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988] 0
2020-06-29 23:52:13.062973: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0: N
2020-06-29 23:52:13.067890: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 301
2 MB memory) -> physical GPU (device: 0, name: GeForce GTX 1050 Ti, pci bus id: 0000:01:00:0, compute capability: 6.1)

Model loaded...
***** Enter "q" to quit *****
(96, 96, 3)
(96, 96, 3)
(3, 96, 96)
(1, 3, 96, 96)
Unknown user detected !
distance:0.66309
(96, 96, 3)
(96, 96, 3)
(3, 96, 96)
(1, 3, 96, 96)
Unknown user detected !
distance:0.66407734
(96, 96, 3)
(96, 96, 3)
(3, 96, 96)
(1, 3, 96, 96)
Unknown user detected !
distance:0.64415705
(96, 96, 3)
(96, 96, 3)
(3, 96, 96)
```

5.2.1 The user is not registered hence it shows the output as unknown user

Once the User is added using add or delete user the system then recognises the person



5.2.2 The system recognising the user and displaying on the screen

6. CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

- ☐ In our project user friendly coding has been adopted.
- ☐ This project on Face recognition has given us an opportunity to explore more about face recognition.
- ☐ This project provided us with the pros and cons of many recognition systems and the trade-off associated with them.

6.2 FUTURE SCOPE

This Real time solution can be further extended for applications such as:

- ☐ Anomaly detection tasks such as detecting terrorists from group of people.
- ☐ Taking attendance of students in colleges using biometric cameras.
- ☐ Restricted access to restricted places.

7.REFERENCES

[1] <https://towardsdatascience.com/one-shot-learning-face-recognition-using-siamese-neural-network-a13dcf739e>

[2] **Andrew Ng explanation of Siamese networks and one shot learning**

https://www.youtube.com/watch?v=96b_weTZb2w

[3] **OpenCV tutorials**

https://docs.opencv.org/master/d9/df8/tutorial_root.html

[4]**Principal component analysis (PCA)**

<https://www.geeksforgeeks.org/ml-principal-component-analysispca/>

[5]**Convolution Neural Networks**

<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>