INTRODUCTION

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images.Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene. Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process.A reliable face-detection approach based on the genetic algorithm and the eigen-face technique

Time and convenience have great importance in human life. We are trying to make metro rail ticketing system more efficient without paper. As face is one of the easiest ways to distinguish the individual identity of a person, we are planning to use Face recognition to implement a Paperless ticketing system.

OBJECTIVE

To make Metro rail ticketing system more efficient and convenient without paper.

FACE RECOGNITION

A biometric is a biometric software application capable of uniquely identifying or verifying a person by comparing and analyzing pattern based on the persons facial contours.

SYSTEM ANALYSIS

Existing System

Paper tickets with barcodes are used as entry

And exit pass for metro rail transport.

Drawbacks

- Inconvenience for safe keeping of paper tickets
- Wastage of paper
- Time consuming

Proposed System

A paperless ticketing system using face recognition by identifying and analyzing the shape of a person's face. Each face has approximately 80 unique nodal points which distinguishes one from another.

Advantages

- Convenient
- Eco-friendly
- Time saving

SYSTEM CONFIGURATION

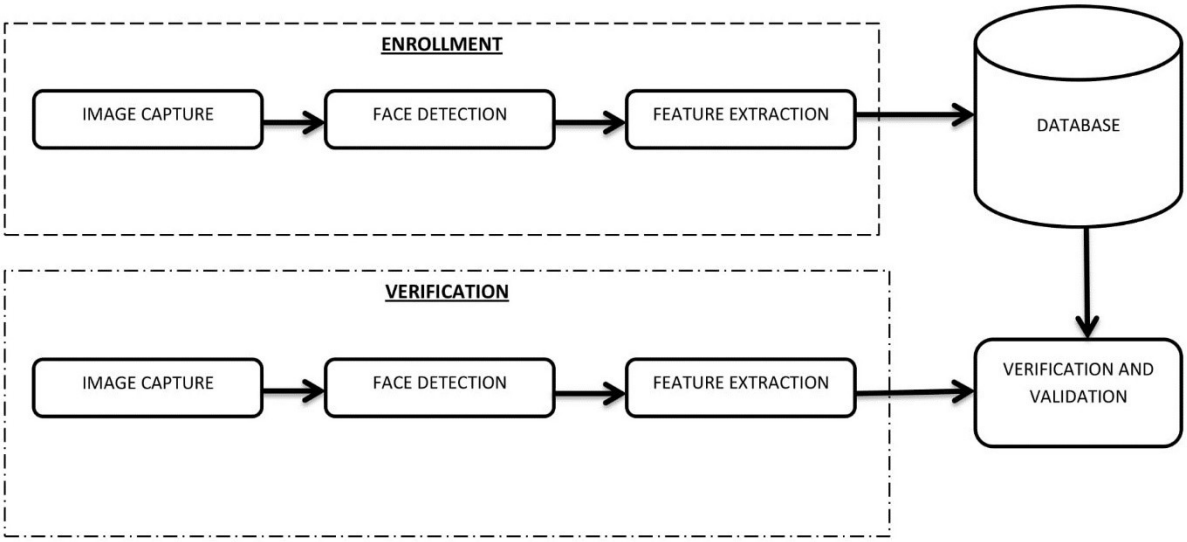
Software Specification

- OS : MS Windows7 or higher versions/ Ubuntu 14.04
- Opencv
- Python 3.7
- Mysql

Hardware Specification

- Intel Core 2 Duo/Quad or higher and 64bit processor
- Hard Disk capacity of 1- 4TB, 4 GB RAM or more
- Input devices: keyboard, digital camera
- Output device : computer monitor

SYSTEM ARCHITECTURE



MODULES

Administrator

- Station master management
- Database management
- Rules and regulations
- Fare rate calculation

Station master

- Ticket reservation
    - Face capturing
    - Details collection
        - » Name
        - » Contact details - phone number
        - » From station
        - » Destination
- Ticket cancellation
    - Database update

Entrance Verification

- Face recognition
- Verification
- Time stamp allocation
- Ticket activation

Exit Validation

- Face recognition
- Verification
- Time stamp checking
- Ticket deactivation

STATION MASTER MODULE

Station Master Functions

- <u>Station Validation</u>
    - Station Username
    - Password

- Ticketing
    - Name
    - Destination
    - Fare calculation
    - Face Detection, Capture & Storing

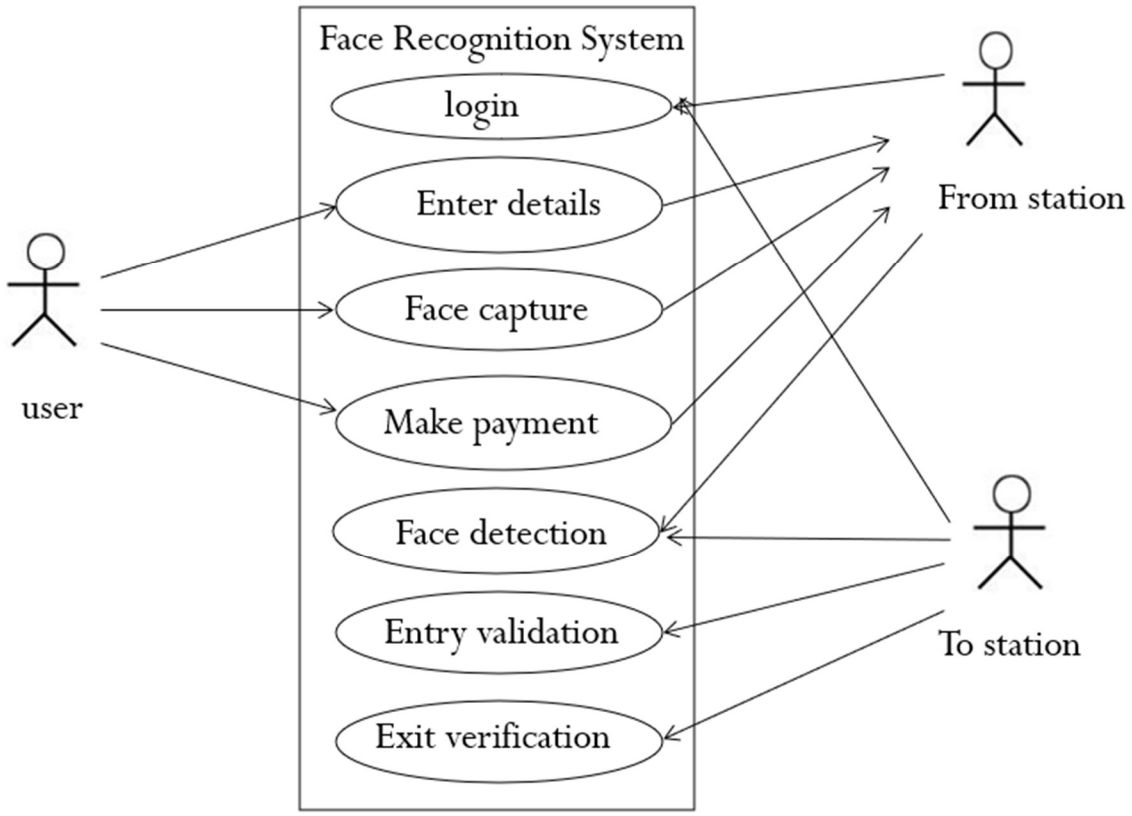
Station Table

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
station_id	Int(5)	No	PRIMARY	None	
station_name	varchar(20)	No		None	
user_name	varchar(20)	No		None	
password	varchar(20)	No		None	

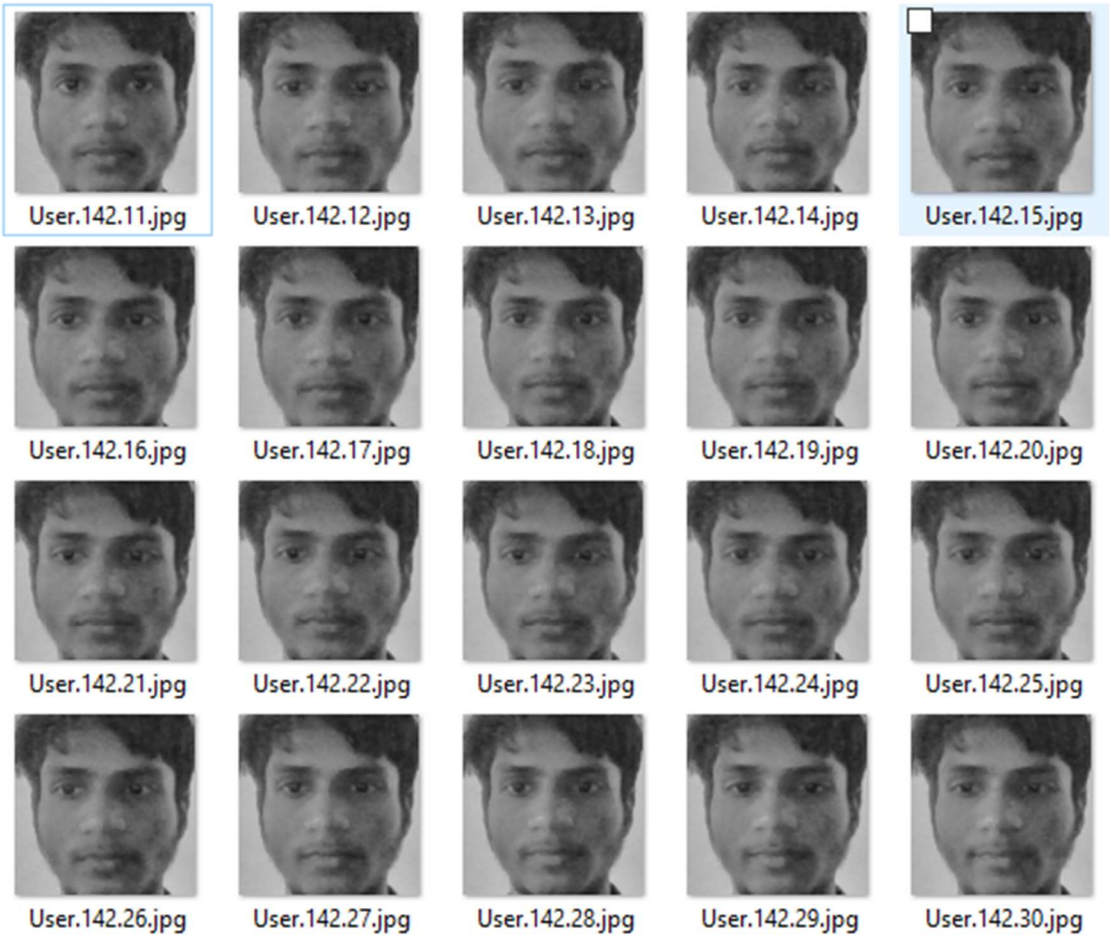
Passenger Table

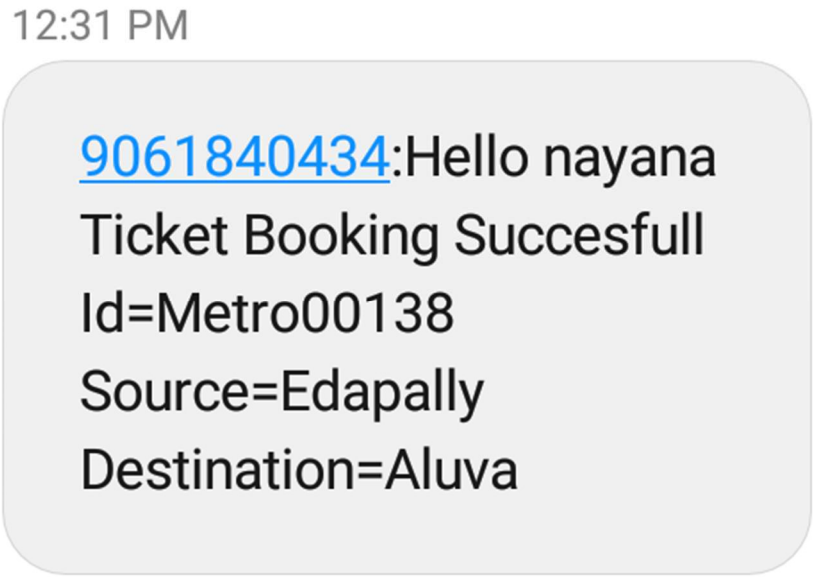
FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
passenger_id	int(5)	No	PRIMARY	None	AUTO_INCREMENT
name	varchar(20)	No		None	
Contact_no	Int(10)	No		None	
from_station	int(5)	No		None	
to_station	int(5)	No		None	
date	datetime	No		None	
status	int(5)	No		None	

Use-Case Diagram



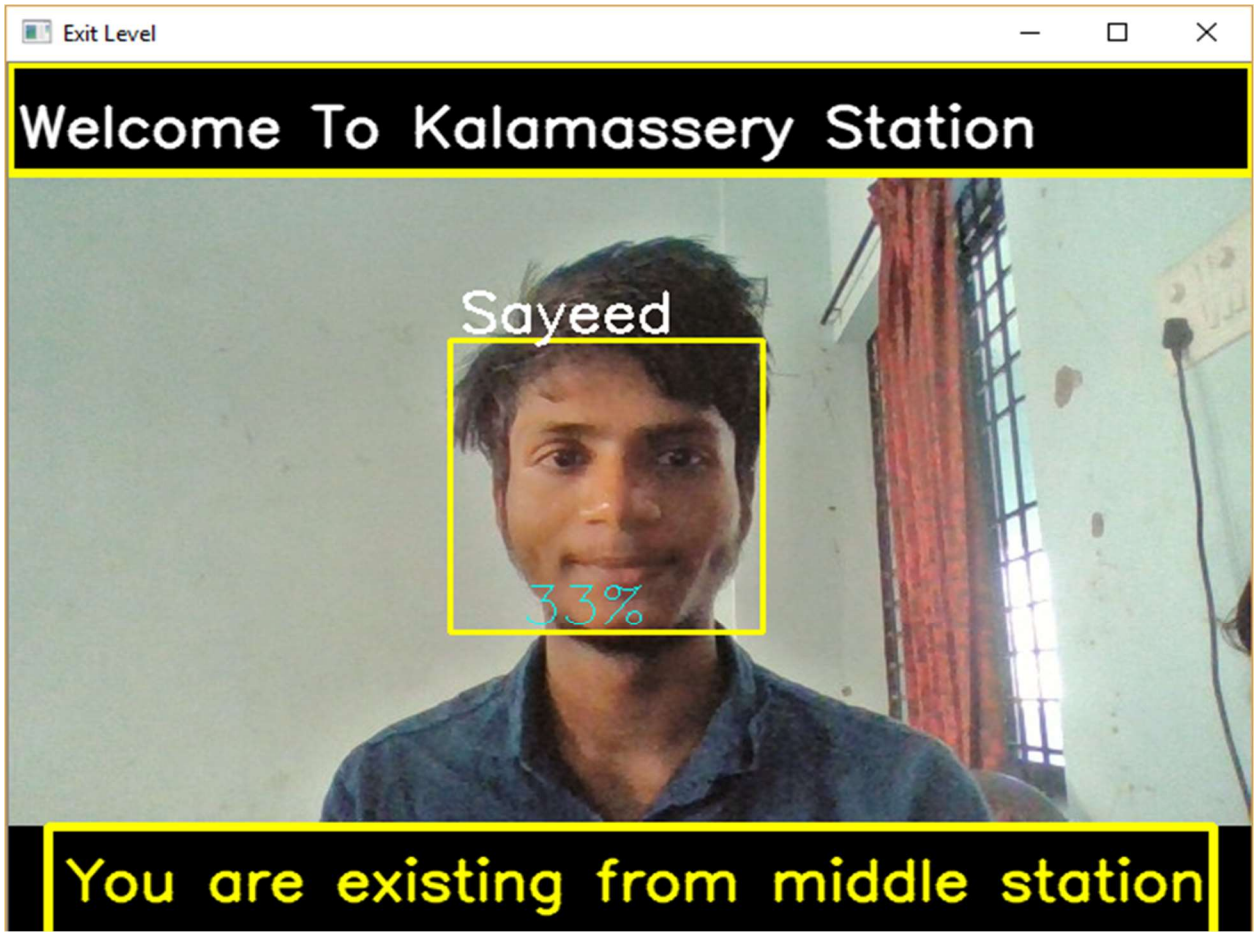
SCREENSHOTS

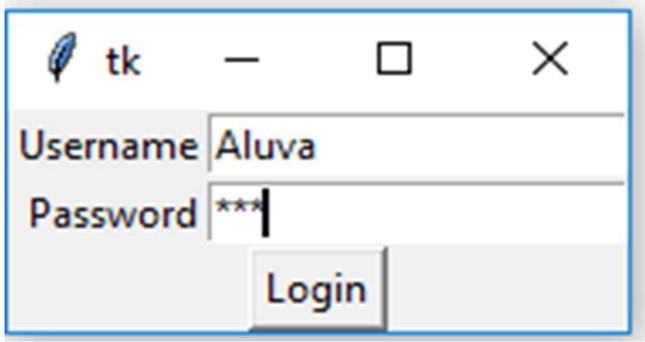


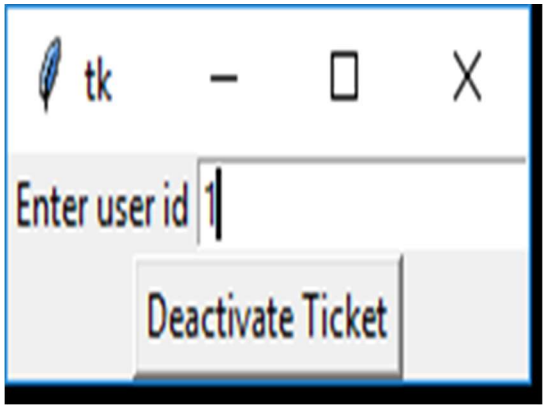


11:08 AM

9061840434:Hello Sayeed
Ticket Activated Succesfull
Time=2018-11-30 11:09:10




tk
Username
Aluva
Password
***
Login


tk
Enter user id
1
Deactivate Ticket

IMPLEMENTATION

Login.py

```
from tkinter import *
import tkinter as tk
import tkinter.messagebox as tm
import mysql.connector
import os

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  passwd="local",
  database="python"
)
mycursor = mydb.cursor()


class LoginFrame(Frame):
    def __init__(self, master):
        super().__init__(master)

        self.label_username = Label(self, text="Username")
        self.label_password = Label(self, text="Password")

        self.entry_username = Entry(self)
        self.entry_password = Entry(self, show="*")

        self.label_username.grid(row=0, sticky=E)
        self.label_password.grid(row=1, sticky=E)
        self.entry_username.grid(row=0, column=1)
        self.entry_password.grid(row=1, column=1)

        self.checkbox = Checkbutton(self, text="Keep me logged in")
        self.checkbox.grid(columnspan=2)

        self.logbtn = Button(self, text="Login", command=self._login_btn_clicked)
        self.logbtn.grid(columnspan=2)


        self.pack()
    def page1(self):
        #page2text.pack_forget()
        #page1text.pack()

        #page1 = Tk() # Opens new window
```

```
    #page1.title('Ticket Cancel')
    #.geometry('1050x650')
    os.system("ticket_cancel.py")

def page2(self):
    '''page2 = Tk() # Opens new window
    page2.title('Ticket VIew')
    page2.geometry('950x650')

    label_head = Label(page2, text="Welcome")
    label_userid = Label(page2, text="Enter the User id")
    entry_userid = Entry(page2)

    label_userid.grid(row=1, sticky=E)
    entry_userid.grid(row=1, column=1)
    label_head.grid(row=0,column=5)
    logbtn = Button(page2, text="Submit", command=self.ticketview)
    logbtn.grid(columnspan=2)
    #label_userid.pack()
    #entry_userid.pack()
    #logbtn.pack(side="left")'''
    os.system("ticketsview.py")

def ticketview(self):
    #ticketview = Tk()
    userid = self.entry_userid.get()
    print(userid)
    query = "SELECT * FROM customer WHERE ID = %s"
    uid = (userid,)
    mycursor.execute(query, uid)
    myresult = mycursor.fetchall()
    print(myresult)
    #ticketview.mainloop()

def page3(self):
    os.system("ticket_cancel.py")


def _login_btn_clicked(self):
    # print("Clicked")
    username = self.entry_username.get()
    password = self.entry_password.get()
    # print(username, password)
    sql3 = "SELECT * FROM station WHERE user_name = %s and password=%s"
    #p1 = Page1(self)
    login = (username,password,)
```

```
        mycursor.execute(sql3, login)
        myresult = mycursor.fetchall()
        validate=len(myresult)
        if validate==1:
            for x in myresult:
                station_id=x[0]
                station_name=x[1]
            r = Tk() # Opens new window
            r.title(station_name+ 'Station')
            r.geometry('1050x650') # Makes the window a certain size
            rlbl = Label(r, text='\n Welcome '+station_name+' Station') # "logged in" label
            page1btn = Button(r, text="Ticket Cancellation", command=self.page1)
            page2btn = Button(r, text="Ticket View", command=self.page2)
            page3btn = Button(r, text="Fine Calculation", command=self.page3)
            #page4btn = Button(r, text="Ticket View", command=self.page4)
            page1btn.pack(side="left")
            page2btn.pack(side="left")
            page3btn.pack(side="left")
            #page4btn.pack(side="left")
            rlbl.pack() # Pack is like .grid(), just different
            r.mainloop()
        else:
            tm.showerror("Login error", "Incorrect username")


root = Tk()
lf = LoginFrame(root)
root.mainloop()
```

ticket_booking.py

```
from json import detect_encoding

import cv2
import os
import numpy as np
import mysql.connector
from tabulate import tabulate

from texttable import Texttable
import datetime
from PIL import Image

# Path for face image database
path = 'dataset'
```

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");


#####
# function to get the images and label data
def getImagesAndLabels(path):

    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []

    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
        img_numpy = np.array(PIL_img,'uint8')

        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)

        for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)

    return faceSamples,ids


########


mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="python"
)
mycursor = mydb.cursor()
#def logged(station_id,station_name):
 # print ("welcome",station_name)


def logged(station_id,station_name):
    cam = cv2.VideoCapture(0)
    cam.set(3, 640) # set video width
    cam.set(4, 480) # set video height
    while(True):
```

```
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# For each person, enter one numeric face id
while True:
    face_name = input('\n Enter Name : ')
    if face_name.isalpha():
        break
    print ("\n[INFO] Please enter valid name")
print("\n")
if station_id!=12:
  print("To North :\n")
  mycursor = mydb.cursor()
  sql3 = "SELECT * FROM station where id>=%s"
  station=(station_id,)
  mycursor.execute(sql3,station)
  myresult = mycursor.fetchall()
  #for i in range(a,12):
  for x in myresult:
    #list.append("->")
    print("("+str(x[0])+")" +x[1], end=" -> ")
      #break
  print("**Finished**", end=" ")
  print("\n")
if station_id!=1:
  print("To South :\n")
  mycursor = mydb.cursor()
  sql3 = "SELECT * FROM station where id<=%s"
  station=(station_id,)
  mycursor.execute(sql3,station)
  myresult = mycursor.fetchall()
  #for i in range(a,12):
  for x in reversed(myresult):
    #list.append("->")
    print("("+str(x[0])+")" +x[1], end=" -> ")
      #break
  print("**Finished**", end=" ")
while(True):
    to_station = input('\n\nEnter To Station : ')
    if to_station.isdigit()and int(to_station)<=12:
        if int(to_station)==int(station_id):
            print("\n[INFO] Both Source And Destination Cannot Be Same")
        else:
            break
    else:
        print("\n[INFO] Please enter valid station id")
```

```
mycursor = mydb.cursor()
dest="SELECT name from station where id=%s"
de=(to_station,)
mycursor.execute(dest, de)
myresult = mycursor.fetchall()
for xy in myresult:
   to_station_name = xy[0]
mycursor = mydb.cursor()
sql = "INSERT INTO customer(name, fromstation,tostation) VALUES (%s, %s, %s)"
val = (face_name, station_id, to_station)
mycursor.execute(sql, val)
mydb.commit()
#print(mycursor.rowcount, "record inserted.")
face_id=mycursor.lastrowid
#a=int(station_id)
#b=int(to_station)
no=abs(int(station_id)-int(to_station))
fare=str(no*10)
date1=str(datetime.date.today())
t = Texttable()
t.add_rows([['WELCOME TO KOCHI METRO \n\n '+station_name+' Station \t'], ['Id:
Metro00'+str(face_id)+'\t\tDate : '+date1], ['\nName : '+face_name.capitalize()+'  \n\nTo Station :
'+str(to_station_name)+'\n'], ['Total Fare   \t:  '+fare+' Rs' ]])
print (t.draw())



print("\n [INFO] Initializing face capture. Look the camera and wait ...")
# Initialize individual sampling face count
count = 0

while(True):

   ret, img = cam.read()
   img = cv2.flip(img, 1) # flip video image vertically
   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
   faces = face_detector.detectMultiScale(gray, 1.3, 5)

   #while faces:
      #print ("hai")
   #else:
     #  print ("not")

   for (x,y,w,h) in faces:
```

```
            cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
            count += 1

            # Save the captured image into the datasets folder
            cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg",
gray[y:y+h,x:x+w])

            cv2.imshow('image', img)

        k = cv2.waitKey(100) & 0xff # Press 'ESC' for exiting video
        #print(k)
        if k == 27:
            break
        elif count >= 30: # Take 30 face sample and stop video
             break

    # Do a bit of cleanup
    print("\n [INFO] Image Captured Successfully")
    #cam.release()
    print ("\n [INFO] Training faces. It will take a few seconds. Wait ...")
    faces,ids = getImagesAndLabels(path)
    recognizer.train(faces, np.array(ids))

    # Save the model into trainer/trainer.yml
    recognizer.write('trainer/trainer.yml')
    # Print the numer of faces trained and end program
    print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.unique(ids))))

    exit_key=input("Press Enter to continue or q to loggout...")
    if exit_key=='q':
        print("You have been successfully logged out successfully!")
        cam.release()
        cv2.destroyAllWindows()
        break




print("**********************WELCOME TO KOCHI
METRO**********************")
while(True):
    user_name = input('\n Enter user name :  ')
    password = input('\n Enter password :  ')
    sql3 = "SELECT * FROM station WHERE user_name = %s and password=%s"
```

```
    login = (user_name,password,)
    mycursor.execute(sql3, login)
    myresult = mycursor.fetchall()
    validate=len(myresult)
    if validate==1:
      for x in myresult:
        station_id=x[0]
        station_name=x[1]
        print
("\n**********************Welcome",station_name,"**********************")
        logged(station_id,station_name)
      break
    else:
      print("\n[INFO] Please enter valid username or password"
```

ticket_view.py

```
from tkinter import *
import tkinter as tk
import tkinter.messagebox as tm
import pymysql

page2 = Tk() # Opens new window
page2.title('Ticket VIew')
page2.geometry('950x650')
page2.configure(background="light blue")
page2.grid_rowconfigure(0, weight=1)
page2.grid_columnconfigure(0, weight=1)

lbl= tk.Label(page2, text="UserId",width=10  ,height=1  ,fg="white"  ,bg="grey" ,font=('times',
15, ' bold ') )
lbl.place(x=50, y=100)

textbox = tk.Entry(page2,width=20  ,bg="white" ,fg="green",font=('times', 15))
textbox.place(x=180, y=100)
lb2= tk.Label(page2, text="",width=50  ,height=3  ,fg="white"  ,bg="light blue" ,font=('times',
15, ' bold ') )
lb2.place(x=50, y=250)


def ticketview():
    mydb = pymysql.connect("localhost","root","","python")
    mycursor = mydb.cursor()

    userid = int(textbox.get())
    print(userid)
    mycursor.execute("SELECT * FROM customer WHERE ID = %d" %(userid))
```

```
    myresult = mycursor.fetchall()
    print(myresult)
    lb2.configure(text= myresult)
    mydb.close()

takedata = tk.Button(page2, text="Submit", command=ticketview ,fg="white"  ,bg="grey"
,width=10  ,height=1, activebackground = "yellow" ,font=('times', 15, ' bold '))
takedata.place(x=50, y=150)
page2.mainloop()
```

CONCLUSION

We are trying to create a convenient and time saving ticketing model for metro rail ticketing with the help of face recognition technique.

REFERENCE

[1]https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

[2] https://www.youtube.com/watch?v=5yPeKQzCPdI

[3] Facial Recognition Technology A Clear and Concise Reference

[4] https://pypi.org/project/face-recognition/

[5] https://www.youtube.com/watch?v=Ax6P93r32KU

[6] The 2018-2023 World Outlook for Facial Recognition