**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**

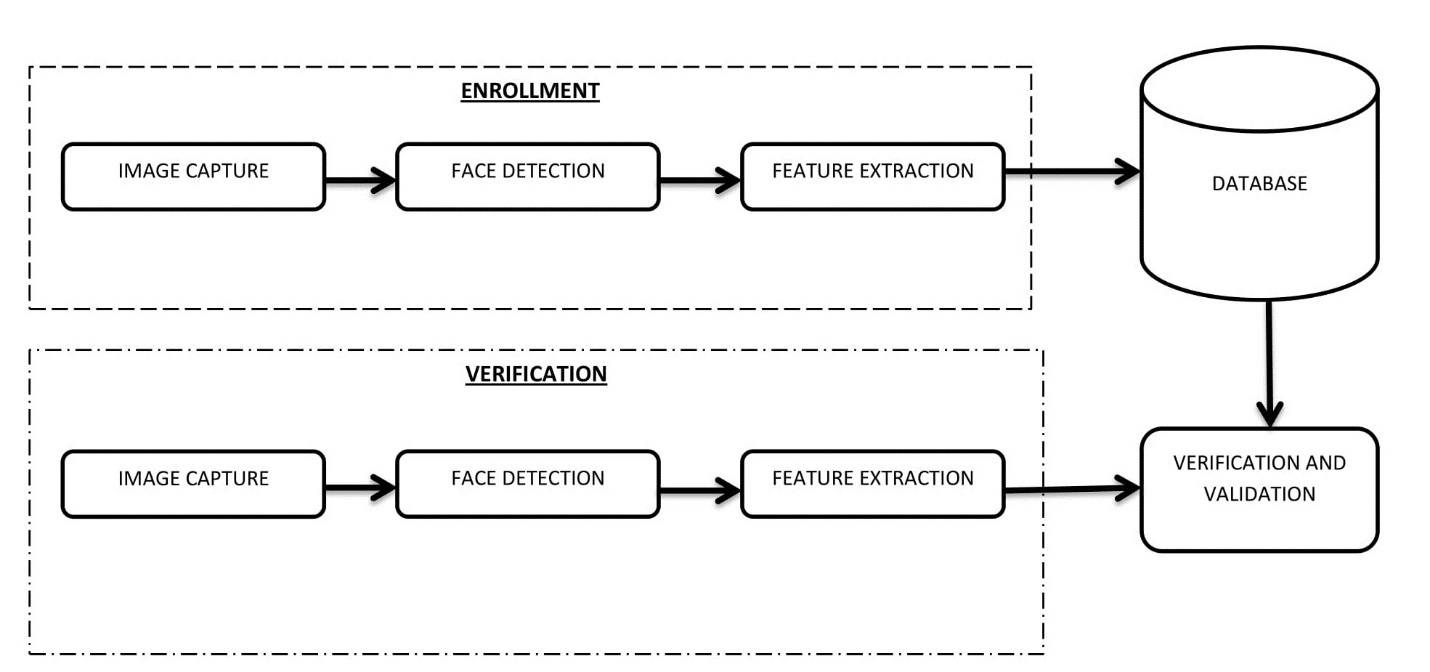
**Software Specification**

* 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**

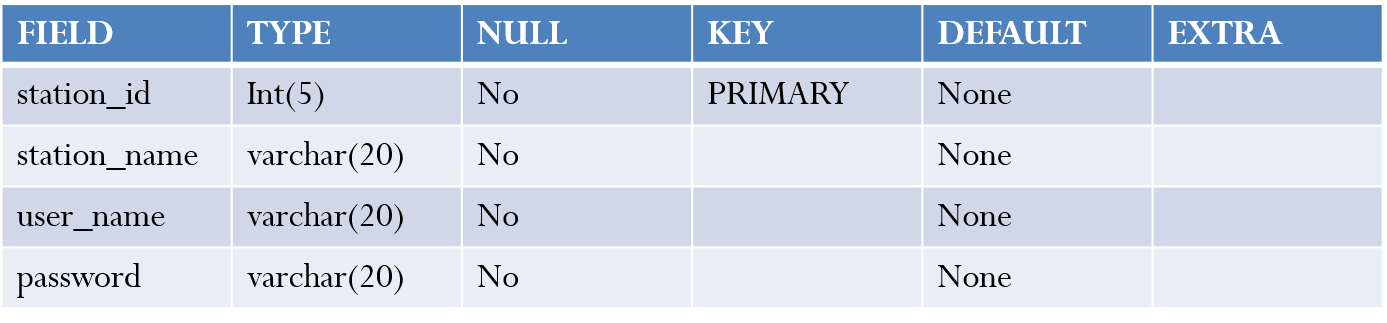
* Face recognition
* Verification
* Time stamp checking
* Ticket deactivation

**STATION MASTER MODULE**

**Station Master Functions**

* Station Validation
  + Station Username
  + Password
* Ticketing
  + Name
  + Destination
  + Fare calculation
  + Face Detection, Capture & Storing

**Station Table**



**Passenger Table**

Table

Description automatically generated

**Use-Case Diagram**

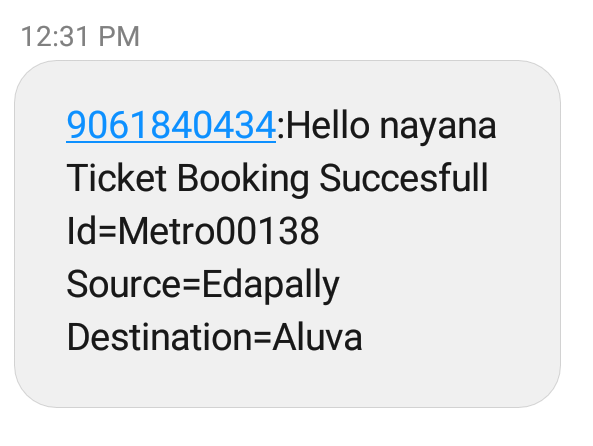
**Diagram

Description automatically generated**

**SCREENSHOTS**

Graphical user interface, application

Description automatically generated



Graphical user interface, text, application, chat or text message

Description automatically generated

A picture containing text, person, orange

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

**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](https://www.amazon.com/dp/B06W9JVHRX?tag=uuid10-20" \t "_blank)