## Description: VtuVISVESVARAYA TECHNOLOGICAL UNIVERSITY BELGAUM - 590018, KARNATAKA

HOBBY PROJECT REPORT ON

## “CRIMINAL FACE IDENTIFICATION SYSTEM”

**BACHELOR OF ENGINEERING IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by**

**ARUN KUMAR T[1EW20CS014] BHAVANA N[1EW20CS018]**

**CHAITHRA B C[1EW20CS025] CHANDAN B RAM[1EW20CS026]**

Under the guidance of

**Mr. Jagadeesh B N**

Assistant Professor, Dept. of CSE.

### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING EAST WEST INSTITUTE OF TECHNOLOGY

**BANGALORE-560091, KARNATAKA.**

### 2022-2023

**EAST WEST INSTITUTE OF TECHNOLOGY**- BENGALURU-560091.

(Affiliated to Visvesvaraya Technological University, Belgaum, Karnataka) DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



**CERTIFICATE**

This is to certify that **ARUN KUMAR T [1EW20CS014], BHAVANA N [1EW20CS018], CHAITHRA B C [1EW20CS025], CHANDAN B RAM [1EW20CS026]** has satisfactorily submitted Hobby Project Report titled **“CRIMINAL FACE IDENTIFICATION SYSTEM”** for **5th semester, Bachelor of Engineering in “COMPUTER SCIENCE & ENGINEERING”**, during the academic year 2022-2023.

Signature of the Coordinators Signature of HOD

**Mr. Jagadeesh B N/ Mrs. Sowmyashree S Dr. Achyutha Prasad N**

(Assistant Professor, Dept. of CS&E) **(**H.O.D, Dept. of CS&E)

## DECLARATION

**ARUN KUMAR T [1EW20CS014], BHAVANA N [1EW20CS018], CHAITHRA B C [1EW20CS025], CHANDAN B RAM [1EW20CS026]** students of fifth semester Bachelor of Engineering in the Department of Computer Science and Engineering of East West Institute of Technology, Bengalore-560091, the hobby project entitled **“CRIMINAL FACE IDENTIFICATION SYSTEM”** has been carried out by us under the supervision of **Mr. Jagadeesh B N,** Assistant Professor, **Mr. V Sunil Kumar**, Assistant Professor, **Mrs. Leena Shruthi H M,** Assistant Professor, **Mrs. Vedashree C R** **,** Assistant Professor, **Mrs. Sowmyashree S,** Assistant Professor, Department of Computer Science and Engineering, EWIT, Visvesvaraya Technological University during the academic year 2022-2023.

**Place: Bengaluru**

**Date: 19-12-2022**

**ARUN KUMAR T [1EW20CS014]**

**BHAVANA N [1EW20CS018]**

**CHAITHRA B C [1EW20CS025]**

**CHANDAN B RAM [1EW20CS026]**

## ACKNOWLEDGEMENT

The completion of project brings with great sense of satisfaction, but it is never completed without thanking the persons who are all responsible for its successful completion. First and foremost, I wish to express our deep sincere feelings of gratitude to my Institution, **EAST WEST INSTITUTE OF TECHNOLOGY**, for providing me an opportunity to do our education.

We extend our deep sense of sincere gratitude to **Dr. K Channakeshavalu**, Principal, East West Institute of Technology, Bengaluru, for having permitted to carry out the hobby project work successfully**.**

We express our heartfelt sincere gratitude to **Dr. Achyutha Prasad N**, Professor and Head of Department of Computer Science and Engineering, East West Institute of Technology, Bengaluru, for his timely suggestions and support.

We express special sincere gratitude to our project guide **Mr. Jagadeesh B N Assistant Professor, Dept. of CS&E**, East West Institute of Technology, Bengaluru, for his constant support and encouragement.

Finally, we would like to thank all the Teaching, Technical faculty and supporting staff members of Department of Computer Science and Engineering, East West Institute of Technology, Bengaluru, for their support.

**ARUN KUMAR T [1EW20CS014]**

**BHAVANA N [1EW20CS018]**

**CHAITHRA B C [1EW20CS025]**

**CHANDAN B RAM [1EW20CS026]**

## ABSTRACT

Crime preventions and criminal identification are the primary issues for the police personnel, since property and life protection are the basic concerns of the police but to combat the crime, the availability of police personnel is limited.

The goal of this project is to identify face of previously convicted persons and provide a solution with higher accuracy, better response rate and an initial step for video surveillance. Solution is proposed based on nature of criminal psychic of repeating crime or involvement in it. This system is used to track history sheeters and recognize them before and after any mischief or any unlawful activity.

In the system we are storing the image of criminal in the database along with its other detail to provide ease in data retrieval and ensuring fast deployment of results in real world. The project is built on python 3.10.8 with the use of OpenCV along with the algorithms like Haar cascade classifier, LBPH and face\_recognition etc. to store the detail of person we have used SQLite

**TABLE OF CONTENTS PAGE NO**

## DECLARATION i

[ACKNOWLEDGEMENT ii](#_bookmark0)

[ABSTRACT iii](#_bookmark1)

CHAPTER 1: Introduction 08

CHAPTER 2: Requirement Specification. 09

2.1Hardware Requirement

2.2Software Requirement

CHAPTER 3: Block Diagram. 10

CHAPTER 4: Description of Technologies 11

CHAPTER 5: Implementation. 14

CHAPTER 6: Snapshots 21

CHAPTER 7: Conclusion 26

CHAPTER 8: Reference. 27

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Title** | **Page No** |
| 3.1 | Block Diagram | 10 |
| 6.1 | Home Page | 21 |
| 6.2 | Criminal Registration Page | 21 |
| 6.3 | Face identification using Photo Match | 22 |
| 6.4 | Face detection in Video surveillance | 22 |
| 6.5 | Face detection in Video surveillance with Criminal Details. | 23 |
| 6.6 | Details of criminal can be viewed from both Photo match and video surveillance. | 23 |

**CHAPTER 1**

# INTRODUCTION

Criminal record contains details about a particular person along with photograph and personal information. To identify any history sheeter we need identification regarding that person. One of the ways is face identification. The face is our primary focus of attention in social intercourse, playing a major role in conveying identity and emotions.

Human ability to remember and recognize faces is remarkable. This system aims to provide a copy of human trait of identification along with the details of person in real time for efficient tracking of habitual criminals.

Criminal face identification system creates database of criminal and recognize the person if one’s image matches with an existing one in distributed environment. The project will be a milestone for video based face identification and for surveillance.

**CHAPTER 2**

# REQUIREMENT SPECIFICATION

In this section the various requirements that are essential for this project are specified. These requirements have to be fulfilled for successful execution of the project. The purpose, scope along with hardware and software requirements is given below:

### SYSTEM REQUIREMENTS:

**HARDWARE REQUIREMENTS:**

* **Processor :** Intel CORE i3
* **RAM :** 2/4/8 GB
* **System type :** 32/64-bit processor
* **Keyboard :** 104 keys
* **Mouse :** 2/3 buttons
* **Camera**

### SOFTWARE REQUIREMENTS:

### Microsoft Windows 7 or higher

### Python 3.8+

### OPEN-CV

### Face\_recognition

### SQLite Studio

**CHAPTER 3**

**BLOCK DIAGRAM**

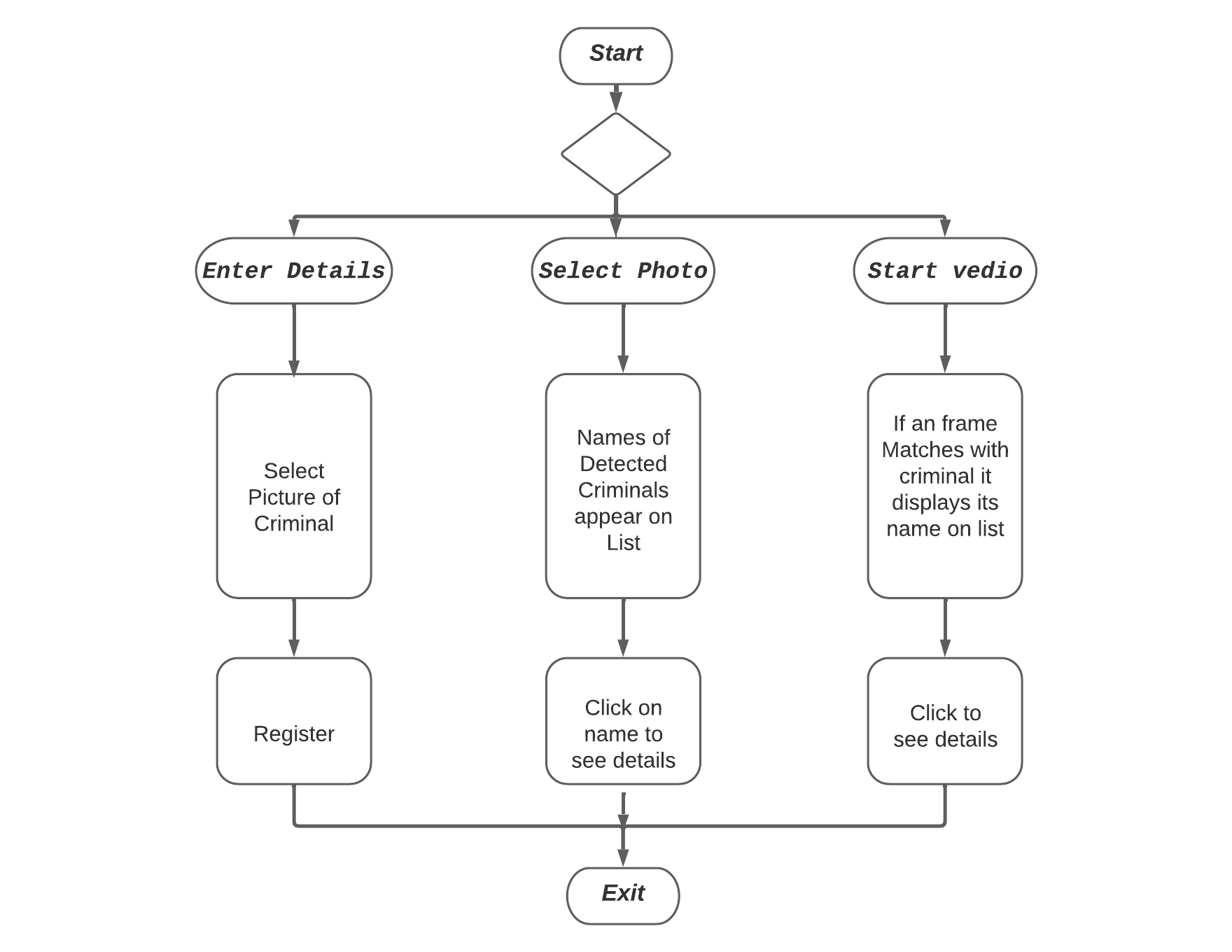


Fig 3.1 : Block Diagram

**CHAPTER 4**

# DESCRIPTION OF TECHNOLOGIES

* 1. **4.1 Description of tools**

**4.1.1 SQLite**:

SQLite is a very popular database which has been successfully used with on disk file format for desktop applications like version control systems, financial analysis tools, media cataloging and editing suites, CAD packages, record keeping programs etc. SQLite is a very light weighted database so, it is easy to use it as an embedded software with devices like televisions, Mobile phones, cameras, home electronic devices, etc. Reading and writing operations are very fast for SQLite database. It is almost 35% faster than File system. It only loads the data which is needed, rather than reading the entire file and hold it in memory. If you edit small parts, it only overwrite the parts of the file which was changed.

* 1. **4.2 Description of technologies**

**4.2.1 Python:**

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. It was initially designed by **Guido van Rossum** in 1991 and developed by Python Software Foundation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming. Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. Rather than building all of its functionality into its core, Python was designed to be highly [extensible](https://en.wikipedia.org/wiki/Extensibility) via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

**4.2.2 OpenCV:**

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source Apache 2 License. OpenCV is written in [C++](https://en.wikipedia.org/wiki/C%2B%2B) and its primary interface is in C++, but it still retains a less comprehensive though extensive older [C interface](https://en.wikipedia.org/wiki/C_(programming_language)). All of the new developments and algorithms appear in the C++ interface. There are bindings in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) and [MATLAB](https://en.wikipedia.org/wiki/MATLAB)/[OCTAVE](https://en.wikipedia.org/wiki/GNU_Octave). The API for these interfaces can be found in the online documentation. Wrappers in several programming languages have been developed to encourage adoption by a wider audience. In version 3.4, [JavaScript](https://en.wikipedia.org/wiki/JavaScript) bindings for a selected subset of OpenCV functions was released as OpenCV.js, to be used for web platforms.

**4.2.3 NumPy:**

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.  [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is [open-source software](https://en.wikipedia.org/wiki/Open-source_software) and has many contributors. NumPy targets the [C,Python](https://en.wikipedia.org/wiki/CPython" \o "CPython) [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, which is a non-optimizing [bytecode](https://en.wikipedia.org/wiki/Bytecode) [interpreter](https://en.wikipedia.org/wiki/Interpreter_(computing)). [Mathematical algorithms](https://en.wikipedia.org/wiki/List_of_algorithms#Computational_mathematics) written for this version of Python often run much slower than [compiled](https://en.wikipedia.org/wiki/Compiler) equivalents due to the absence of compiler optimization. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly [inner loops](https://en.wikipedia.org/wiki/Inner_loop), using NumPy.

**4.2.4 Tkinter:**

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and macOS installs of Python. The name Tkinter comes from Tk interface. As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete [Tcl](https://en.wikipedia.org/wiki/Tcl" \o "Tcl) [interpreter](https://en.wikipedia.org/wiki/Interpreter_(computing)) embedded in the Python [interpreter](https://en.wikipedia.org/wiki/Language_interpretation).

**4.2.5 Python Imaging Library (Pillow):**

Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

**4.2.6:** **face\_recognition module:**

face\_recognition recognize and manipulate faces from Python or from the command line with

the world’s simplest face recognition library. It is built using [dlib](http://dlib.net/)’s state-of-the-art face recognition

built with deep learning. The model has an accuracy of 99.38% on the [labelled Faces in the Wild](http://vis-www.cs.umass.edu/lfw/) benchmark. This also provides a simple face\_recognition command line tool that lets you do face recognition on a folder of images from the command line.

**4.2.6.1** **Face detection:**

We have used OpenCV which presents a Haar cascade classifier, which is used for face detection. The Haar cascade classifier uses the AdaBoost algorithm to detect multiple facial features. First, it reads the image to be detected and converts it into the gray image, then loads Haar cascade classifier to decide whether it contains a human face. If so, it proceeds to examine the face features and draw a rectangular frame on the detected face. Otherwise, it continues to test the next picture.

**4.2.6.1** **Face extraction:**

Face extraction The LBP operator is applied to describe the contrast information of a pixel to its neighborhood pixels. The original LBP operator is defined in the window of 3\*3.Using the median pixel value as the threshold of the window, it compares with the gray value of the adjacent 8 pixels. If the neighborhood pixel value is larger or equal compare to the median pixel value, the value of pixel position is marked as 1, otherwise marked as 0.

**CHAPTER 5**

# IMPLEMENTATION

# SOURCE CODE:

from tkinter import \*

import shutil

from PIL import ImageTk,Image

import sqlite3

from tkinter import filedialog

import tkinter.messagebox as tmsg

from subprocess import call

def register():

    call(["python", "registerGUI.py"])

def VideoSurveillance():

    call(["python", "surveillance.py"])

def detectCriminal():

    call(["python", "detect.py"])

root = Tk()

root.minsize(1550,1200)

root.title("CFIS- Criminal Face Identification System")

root.configure(bg="#7AC5CD")

Fullname=StringVar()

father=StringVar()

var = IntVar()

c=StringVar()

d=StringVar()

var1= IntVar()

file1=""

image=Image.open("home.png")

photo=ImageTk.PhotoImage(image)

photo\_label=Label(image=photo,width=1550,height=220,bg='white').place(x=0,y=0)

photo\_label

label\_0 = Label(root, text="CRIMINAL FACE IDENTIFICATION SYSTEM",width=38,font=("Times New Roman", 40),anchor=CENTER,bg="#FF4500",fg="white")

label\_0.place(x=200,y=300)

Button(root, text='REGISTER CRIMINAL',width=35,height=3,bg='blue',fg='white',font=("bold", 15),command=register).place(x=600,y=400)

Button(root, text='PHOTO MATCH',width=35,height=3,bg='blue',fg='white',font=("bold", 15),command=detectCriminal).place(x=600,y=500)

Button(root, text='VIDEO SURVEILLANCE',width=35,height=3,bg='red',fg='white',font=("bold", 15),command=VideoSurveillance).place(x=600,y=600)

import cv2 as cv

import numpy as np

import sqlite3

import face\_recognition as fr

import numpy as np

from tkinter import \*

from tkinter import ttk

from PIL import Image,ImageTk

import os

import imutils

import math

import winsound

# from sklearn.metrics import accuracy\_score

#####################################################################################################

class App:

    def \_\_init\_\_(self,video\_source=0):

        self.appname="CFIS- Criminal Face Identification System"

        self.window=Tk()

        self.window.title(self.appname)

        self.window.geometry('1350x720')

        self.window.state("zoomed")

        self.window["bg"]='#7AC5CD'

        self.video\_source=video\_source

        self.vid=myvideocapture(self.video\_source)

        self.label=Label(self.window,text=self.appname,font=("bold",20),bg='blue',fg='white').pack(side=TOP,fill=BOTH)

        self.canvas=Canvas(self.window,height=700,width=700,bg='#7AC5CD')

        self.canvas.pack(side=LEFT,fill=BOTH)

        self.detectedPeople=[]

        self.images=self.load\_images\_from\_folder("images")

        #get image names

        self.images\_name=[]

        for img in self.images:

            self.images\_name.append(fr.load\_image\_file(os.path.join("images",img)))

        #get their encodings

        self.encodings=[]

        for img in self.images\_name:

            self.encodings.append(fr.face\_encodings(img)[0])

        #get id from images

        self.known\_face\_names=[]

        for name in self.images:

            self.known\_face\_names.append((os.path.splitext(name)[0]).split('.')[1])

        self.face\_locations=[]

        self.face\_encodings=[]

        self.face\_names=[]

        self.process\_this\_frame=True

        print(self.known\_face\_names)

        self.faceDetect = cv.CascadeClassifier("haarcascade\_frontalface\_default.xml")

        self.recognizer = cv.face\_LBPHFaceRecognizer.create()

        self.recognizer.setRadius(5)

        # self.recognizer.read("recognizer\\training\_data.yml")

        self.Id=0

        #== showing treeview

        self.tree= ttk.Treeview(self.window, column=("column1", "column2", "column3","column4","column5"), show='headings')

        self.tree.heading("#1", text="Cr-ID")

        self.tree.column("#1", minwidth=0, width=70, stretch=NO)

        self.tree.heading("#2", text="NAME")

        self.tree.column("#2", minwidth=0, width=200, stretch=NO)

        self.tree.heading("#3", text="CRIME")

        self.tree.column("#3", minwidth=0, width=150, stretch=NO)

        self.tree.heading("#4", text="Nationality")

        self.tree.column("#4", minwidth=0, width=100, stretch=NO)

        self.tree.heading("#5", text="MATCHING %")

        self.tree.column("#5", minwidth=0, width=120, stretch=NO)

        ttk.Style().configure("Treeview.Heading",font=('Calibri', 13,'bold'), foreground="red", relief="flat")

        self.tree.place(x=710,y=50)

        self.update()

        self.window.mainloop()

    def load\_images\_from\_folder(self,folder):

        images=[]

        for filename in os.listdir(folder):

            images.append(filename)

        return images

    def doubleclick(self,event):

        item=self.tree.selection()

        itemid=self.tree.item(item,"values")

        ide=itemid[0]

        ide=(int(ide))

        self.viewdetail(ide)

    def viewdetail(self,a):

        conn = sqlite3.connect("criminal.db")

        cur = conn.cursor()

        cur.execute("SELECT \* FROM people where Id="+str(a))

        rows = cur.fetchall()

        print(rows)

        for row in rows:

            label\_n = Label(self.window, text=row[1],bg="#7AC5CD",fg='white',width=20,font=("bold", 12))

            label\_n.place(x=1130,y=400)

            label\_f = Label(self.window, text=row[3],bg="#7AC5CD",fg='white',width=20,font=("bold", 12))

            label\_f.place(x=1130,y=430)

            label\_m = Label(self.window, text=row[4],bg="#7AC5CD",fg='white',width=20,font=("bold", 12))

            label\_m.place(x=1130,y=460)

            label\_g = Label(self.window, text=row[2],bg="#7AC5CD",fg='white',width=20,font=("bold", 12))

            label\_g.place(x=1130,y=490)

            label\_r = Label(self.window, text=row[5],bg="#7AC5CD",fg='white',width=20,font=("bold", 12))

            label\_r.place(x=1130,y=520)

            label\_bl = Label(self.window, text=row[6],bg="#7AC5CD",fg='white',width=20,font=("bold", 12))

            label\_bl.place(x=1130,y=550)

            label\_b = Label(self.window, text=row[7],bg="#7AC5CD",fg='white',width=20,font=("bold", 12))

            label\_b.place(x=1130,y=580)

            label\_n = Label(self.window, text=row[8],bg="#7AC5CD",fg='white',width=20,font=("bold", 12))

            label\_n.place(x=1130,y=610)

            label\_c = Label(self.window, text=row[9],width=30,bg="#7AC5CD",font=("bold", 15),fg="red")

            label\_c.place(x=1060,y=640)

        conn.close()

        label\_name = Label(self.window, text="Name",bg="#7AC5CD",fg='yellow',width=20,font=("bold", 12))

        label\_name.place(x=930,y=400)

        label\_father = Label(self.window, text="FatherName",bg="#7AC5CD",fg='yellow',width=20,font=("bold", 12))

        label\_father.place(x=930,y=430)

        label\_mother = Label(self.window, text="MotherName",bg="#7AC5CD",fg='yellow',width=20,font=("bold", 12))

        label\_mother.place(x=930,y=460)

        label\_gender = Label(self.window, text="Gender",bg="#7AC5CD",fg='yellow',width=20,font=("bold", 12))

        label\_gender.place(x=930,y=490)

        label\_religion = Label(self.window, text="Religion",bg="#7AC5CD",fg='yellow',width=20,font=("bold", 12))

        label\_religion.place(x=930,y=520)

        label\_bloodgroup = Label(self.window, text="Blood Group",bg="#7AC5CD",fg='yellow',width=20,font=("bold", 12))

        label\_bloodgroup.place(x=930,y=550)

        label\_body = Label(self.window, text="BodyMark",bg="#7AC5CD",fg='yellow',width=20,font=("bold", 12))

        label\_body.place(x=930,y=580)

        label\_nat = Label(self.window, text="Nationality",bg="#7AC5CD",fg='yellow',width=20,font=("bold", 12))

        label\_nat.place(x=930,y=610)

        label\_crime = Label(self.window, text="Crime",bg="#7AC5CD",width=23,font=("bold", 15),fg="red")

        label\_crime.place(x=900,y=640)

        x='user.'+str(a)+".png"

        image=Image.open('images/'+x)

        image = image.resize((180,180), Image.ANTIALIAS)

        photo=ImageTk.PhotoImage(image)

        photo\_l=Label(image=photo,width=180,height=180).place(x=750,y=450).pack()

    def getProfile(self,id):

        conn=sqlite3.connect("criminal.db")

        cmd="SELECT ID,name,crime,nationality FROM people where ID="+str(id)

        cursor=conn.execute(cmd)

        profile=None

        for row in cursor:

            profile=row

            break

        conn.close()

        return profile

    def showPercentageMatch(self,face\_distance,face\_match\_threshold=0.6):

        if face\_distance > face\_match\_threshold:

            range = (1.0 - face\_match\_threshold)

            linear\_val = (1.0 - face\_distance) / (range \* 2.0)

            return linear\_val

        else:

            range = face\_match\_threshold

            linear\_val = 1.0 - (face\_distance / (range \* 2.0))

            return linear\_val + ((1.0 - linear\_val) \* math.pow((linear\_val - 0.5) \* 2, 0.2))

    def update(self):

        isTrue,frame=self.vid.getframe()

        if isTrue:

            self.photo=ImageTk.PhotoImage(image=Image.fromarray(frame))

            self.canvas.create\_image(0,0,image=self.photo,anchor=NW)

            #Resize the frame of video to 1/4 size for fast process

            small\_frame=cv.resize(frame,(0,0),fx=0.25,fy=0.25)

            #convert the image to BGR color(openCV) to RGB color(face\_recognition)

            rgb\_small\_frame=small\_frame[:,:,::-1]

            #Only process every other frame of video to save time

            if self.process\_this\_frame:

                #find all the faces and face encodings in the current frame of video

                self.face\_locations=fr.face\_locations(rgb\_small\_frame)

                self.face\_encodings=fr.face\_encodings(rgb\_small\_frame,self.face\_locations)

                self.face\_names=[]

                for face\_encoding in self.face\_encodings:

                    #See if the face is a match for known face(s)

                    matches=fr.compare\_faces(self.encodings,face\_encoding)

                    Id=0

                    face\_distances=fr.face\_distance(self.encodings,face\_encoding)

                    best\_match\_index=np.argmin(face\_distances)

                    percent=self.showPercentageMatch(face\_distances[best\_match\_index])

                    #acc = accuracy\_score(self.encodings[best\_match\_index], face\_encoding)

                    if matches[best\_match\_index]:

                        Id=self.known\_face\_names[best\_match\_index]

                    self.face\_names.append(Id)

                    profile=self.getProfile(Id)

                    confidence=str(round(percent\*100,2))+"%"

                    if profile not in self.detectedPeople and profile!=None:

                        self.detectedPeople.append(profile)

                        profilex=list(profile)

                        profilex.append(confidence)

                        profile=tuple(profilex)

                        self.tree.insert("", 'end', values=profile)

                        self.tree.bind("<Double-1>",self.doubleclick)

                        winsound.PlaySound("SystemExit", winsound.SND\_ALIAS)

                    print(profile)

            self.process\_this\_frame=not self.process\_this\_frame

        self.window.after(15,self.update)

#####################################################################################

class myvideocapture:

    def \_\_init\_\_(self,video\_source=0):

        self.vid=cv.VideoCapture(video\_source)

        if not self.vid.isOpened():

            raise ValueError("unable to open",video\_source)

        self.width=self.vid.get(cv.CAP\_PROP\_FRAME\_WIDTH)

        self.height=self.vid.get(cv.CAP\_PROP\_FRAME\_HEIGHT)

    def getframe(self):

        if self.vid.isOpened():

            ret, frame = self.vid.read()

            frame=imutils.resize(frame,height=700)

            if ret:

                return (ret, cv.cvtColor(frame, cv.COLOR\_BGR2RGB))

            else:

                return (ret, None)

        else:

            return (ret, None)

    def \_\_del\_\_(self):

        if self.vid.isOpened():

            self.vid.release()

if \_\_name\_\_=="\_\_main\_\_":

    App()

**CHAPTER 6**

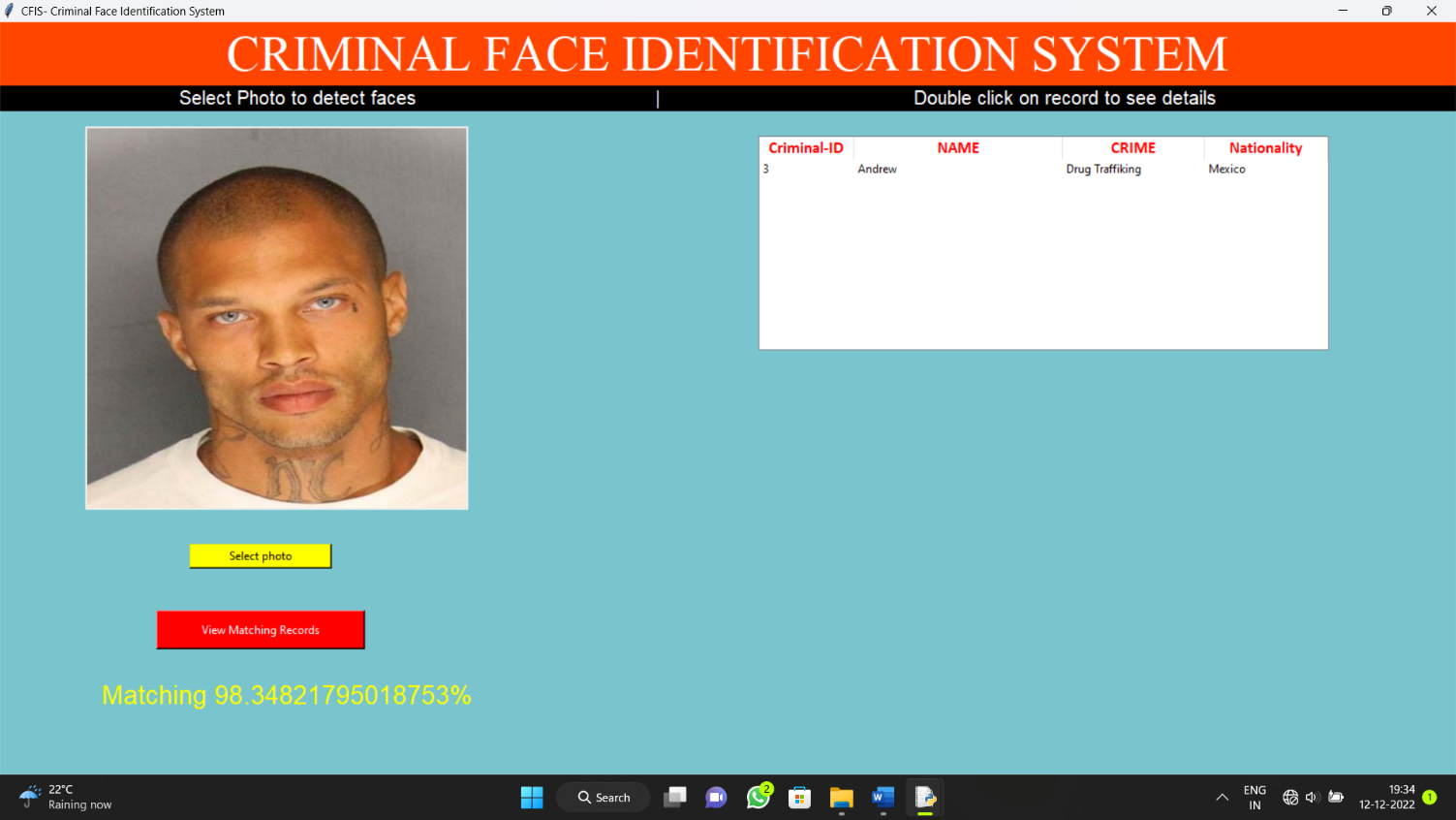
**SNAPSHOTS**

# 

# Fig 6.1: Home Page

# 

# Fig 6.2: Criminal Registration Page.



# Fig 6.3: Face identification using Photo Match.

Fig 6.4: Face detection in Video surveillance.

# Fig 6.5: Face detection in Video surveillance with Details.

# Fig 6.6: Details of criminal can be viewed from both Photo match and video surveillance.

**CHAPTER 7**

**CONCLUSION**

In this project, we are able to detect and recognize faces of the criminals in an image and in a video stream obtained from a camera in real time. We have used Haar feature-based cascade classifiers in OpenCV approach for face detection. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. Also, we have used Local Binary Patterns Histograms (LBPH) for face recognition.

The project will be a milestone for video based face identification and for surveillance systems. Several advantages of this algorithm are: Efficient feature selection, Scale and location invariant detector, instead of scaling the image itself, we scale the features such a generic detection scheme can be trained for detection of other types of objects (e.g. cars, sign boards, number plates, etc).

LBPH recognizer can recognize faces in different lighting conditions with high accuracy. Also, LBPH can recognize efficiently even if single training image is used for each person. Our application has some disadvantages like: Detector is most effective only on frontal images of faces, it can hardly cope with 45° face rotation both around the vertical and horizontal axis.

# FUTURE ENHANCEMENTS

* Light normalization may allow the threshold value to increase.
* Improvement of face recognition using specific character in the face.
* Analyze the face in 3D by using more than one camera.
* Addition of Iris based identification technology.

**CHAPTER 8**

**REFERENCES**

### Reference URL’S:

## [1]. GeeksforGeeks: https://www.geeksforgeeks.org

## [2]. W3Schools: https //www.w3schools.com

## [3]. Stackoverflow https:// www.stackoverflow.com

### Reference Books/Journals:

[1] LBPH Based Improved Face Recognition at Low Resolution

Aftab Ahmed, Jiandong Guo, Fayaz Ali, Farha Deeba and Awais Ahmed

2018 International Conference of Artificial Intelligence and Big Data, China.

[2] Criminal Identification System Using Face Detection and Recognition

Piyush Kakkar and Vibhor Sharma International Journal of Advance Research in Computer and Communication Technology Vol 7 issue 3 march 2018

[3] Learn Python the Hard Way: 3rd Edition by Zed A. Shaw