INTEGRITY DRIVEN FACIAL RECOGNITION GUARDIAN USING DEEP LEARNING MODELS AND IDENTITY VERIFIER

A PROJECT REPORT
(U19ADS801)

Submitted by

AKSHAYA A (1920110001)
INDHRA V (1920110015)
PRIYAADHARSHINI V S (1920110036)
MADHAVAN K S (1920110703)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

SONA COLLEGE OF TECHNOLOGY, SALEM-5
(Autonomous)

ANNA UNIVERSITY: CHENNAI 600 025

07 MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**INTEGRITY DRIVEN FACIAL RECOGNITION GUARDIAN USING DEEP LEARNING MODELS AND IDENTITY VERIFIER"** is the bonafide work of "**AKSHAYA A (1920110001), INDHRA V (1920110015), PRIYAADHARSHINI V S (1920110036), MADHAVAN K S (1920110703)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. J. Akilandeswari	Mr. J L Aldo Stalin
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	Assistant Professor
Department Of Information Technology	Department Of Information Technology
Sona College of Technology	Sona College of Technology
Salem- 636 005.	Salem- 636 005.

Submitted for Project viva voce examination held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost, we thank to **power of almighty** for showing us inner peace and for all blessings. Special gratitude to our parents, for showing their support and love always.

We express our sincere thanks to chairman **Sri.C.VALLIAPPA** and principal **Dr.S.R.R.SENTHILKUMAR** for providing adequate facilities to complete the project.

We are immensely grateful to our Head of Information Technology Department, **Dr.J.AKILANDESWARI** for the continue encouragement to complete the project.

We express our heartfelt thanks to our project supervisor **J L ALDO STALIN** for his valuable guidance and fruitful discussions throughout the course of the project work.

We express our sincere thanks to our project coordinator **C SANTHOSH KUMAR** for guiding us throughout the project reviews and encouraging us to follow the phases of project without any obstacles.

We feel proud in sharing this success with all our department Faculty, Staff members and friends who helped directly and indirectly to completing this project successfully.

ABSTRACT

Ensuring the safety of students and providing them with a platform to report complaints is paramount. Thus, the project aims to create a user-friendly website tailored for students to easily lodge complaints. Leveraging facial recognition technology, the system will ensure secure access to the complaint portal. Emphasis will be placed on simplicity in both website design and the complaint submission process to enhance usability. Additionally, stringent security measures will be implemented to safeguard student data and privacy throughout the system.The goal of this project is to identify the face of individuals with prior disciplinary issues and provide a solution with higher accuracy and a better response rate. This system serves as an initial step for video surveillance and aims to recognize individuals before and after any incidents of misconduct or rule-breaking.

In this system, images of individuals with a history of misconduct are stored in a database along with other relevant details, facilitating easy data retrieval and ensuring quick deployment of results in real-world scenarios. The project is developed using Python 3.5, leveraging OpenCV along with algorithms such as the Haar cascade classifier, LBPH, and face recognition. To store individual details, SQLite is utilized.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	FULL FORMS
SFIS	Student Face Identification System
LBP	Local Binary Patterns
SQLite	Structured Query Language Lite
PIL	Python Imaging Library
LBPH	Local Binary Patterns Histograms
DOE	Date of Entry
DOB	Date of Birth

CHAPTER 1

INTRODUCTION

The record contains details about a particular person along with photographs 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. The human ability to remember and recognize faces is remarkable. This system aims to provide a copy of human traits of identification along with the details of the person in real-time for efficient tracking of habitual students. A student face identification system creates a database of students and recognizes the person if one's image matches an existing one in a distributed environment. The project will be a milestone for video-based face identification and surveillance.

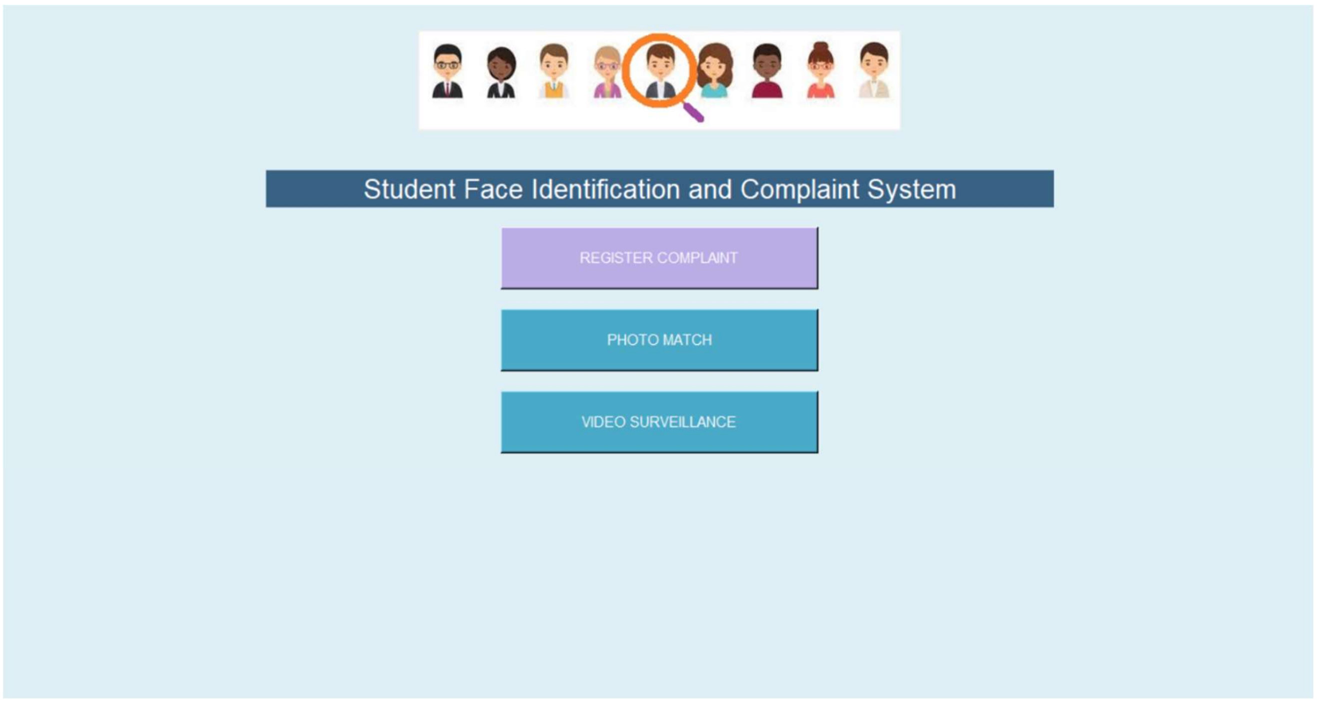


Fig 1.1 Home page

In today's rapidly evolving digital landscape, ensuring the safety and security of educational institutions is paramount. Traditional methods of identification and surveillance are being supplemented with cutting-edge technologies to create more efficient and effective systems. One such technology is student face identification systems, which leverage the remarkable human ability to remember and recognize faces. This project aims to develop a real-time face identification system for tracking habitual students, providing a comprehensive database and surveillance solution.

Face identification holds significant significance as it offers accurate and efficient recognition of individuals, crucial for enhancing security on campus. However, implementing such systems comes with its challenges, including ensuring accuracy, addressing privacy concerns, scalability, and ethical considerations. The proposed system architecture encompasses face detection, feature extraction, face matching, database management, real-time monitoring, and integration with existing security systems.

The implementation and deployment involve stages like data collection, model training, system integration, testing, validation, deployment, and ongoing maintenance. Student face identification systems represent a significant advancement in campus security and surveillance, offering real-time identification and tracking of individuals to create safer learning environments. Addressing challenges and deploying systems responsibly are crucial for realizing the potential benefits of face identification technology in educational institutions.

Significance of Face Identification:

The significance of face identification lies in its ability to provide accurate and efficient recognition of individuals. Unlike other forms of biometric identification, such as fingerprints or iris scans, faces are easily observable and non-intrusive, making them suitable for use in various environments. Additionally, the human brain is highly adept at recognizing faces, with studies showing that individuals can remember thousands of faces with remarkable accuracy.

Traditional methods of identification

Traditional methods of identification, such as ID cards and manual checks, have limitations in terms of efficiency and accuracy. Moreover, these methods are susceptible to forgery and misuse. Student face identification systems offer a sophisticated alternative, leveraging computer vision algorithms to analyze and recognize facial features in real-time. By creating a database of student faces and employing surveillance cameras, these systems can track the movement of individuals, identify unauthorized persons, and enhance overall security on campus.

Student face identification systems represent a significant advancement in campus security and surveillance. By harnessing the power of computer vision and deep learning, these systems offer real-time identification and tracking of individuals, enhancing safety and efficiency in educational environments. While challenges such as accuracy, privacy, and ethical considerations must be addressed, the potential benefits of implementing face identification systems are substantial. With careful planning, responsible deployment, and ongoing maintenance, these systems can play a vital role in creating safer and more secure learning environments for students and staff.

CHAPTER 2

LITERATURE REVIEW

2.1 Face detection [1]

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 **.**

2.1.1 Haar features [2]

A simple rectangular Haar-like feature can be defined as the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image. This modified feature set is called the 2rectangle feature. Faces are scanned and searched for Haar features of the current stage. The weight and size of each feature and the features themselves are generated using a machine learning algorithm from AdaBoost.

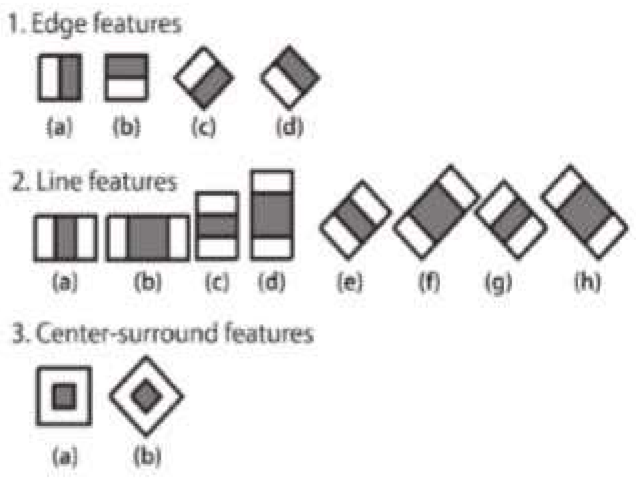


Fig 2.1 Some common Haar Feature (original paper)

2.2 Face extraction [3]

The LBP operator is applied to describe the contrast information of a pixel to its neighbourhood 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 neighbourhood pixel value is larger or equal compared to the median pixel value, the value of pixel position is marked as 1, otherwise marked as 0. The function is defined as shown in equation 1. It can be illustrated in Figure 2.2.

$$N(x) = \begin{Bmatrix} 1, x \geq 0 \\ 0, x < 0 \end{Bmatrix}$$

eq. (1)

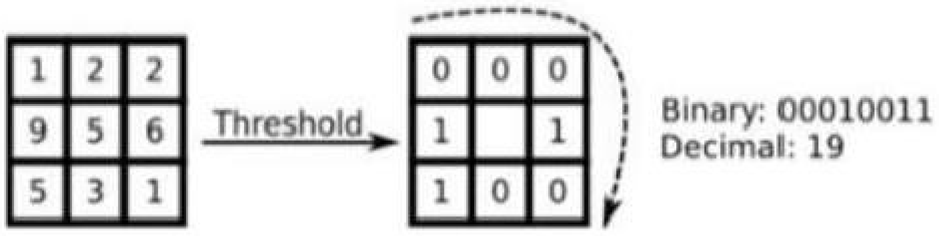


Fig 2.2 original LBP operator

2.3 Student Face Recognition System [4]

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.

There are a lot of advantages to use SQLite for this application:

1) Lightweight: 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.

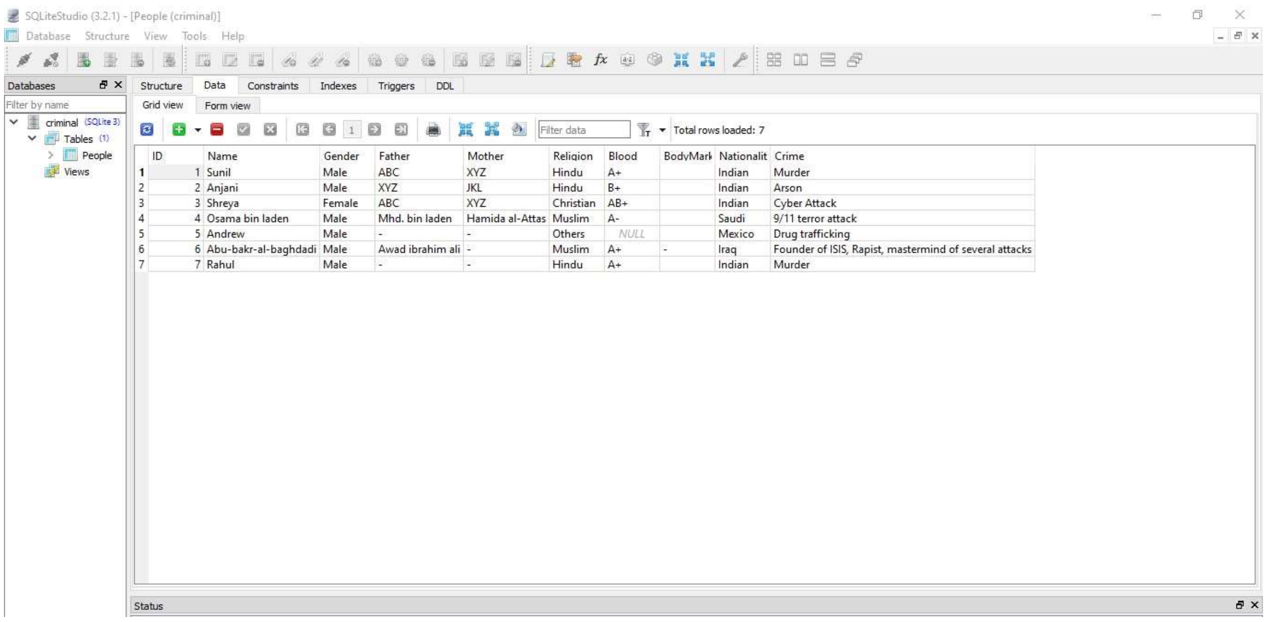
2) Better Performance: Reading and writing operations are very fast for SQLite database. It is almost 35% faster than File system. It only loads the data which is needed, rather than reading the entire file and hold it in memory. If you edit small parts, it only overwrite the parts of the file which was changed.

3) No Installation Needed: SQLite is very easy to learn. You don't need to install and configure it. Just download SQLite libraries in your computer and it is ready for creating the database.

4) Reliable: It updates your content continuously so, little or no work is lost in a case of power failure or crash.

5) Portable: SQLite is portable across all 32-bit and 64-bit operating systems and big- and little-endian architectures.

6) Reduce Cost and Complexity: It reduces application cost because content can be accessed and updated using concise SQL queries instead of lengthy and error-prone procedural queries.


SQLiteStudio (3.2.1) - [People (criminal)]
Database Structure View Tools Help
Databases
Filter by name
criminal (SQLite 3)
Tables (1)
People
Views
Structure Data Constraints Indexes Triggers DDL
Grid view Form view
Filter data
Total rows loaded: 7
Status

	ID	Name	Gender	Father	Mother	Religion	Blood	BodyMark	Nationalit	Crime
1	1	Sunil	Male	ABC	XYZ	Hindu	A+		Indian	Murder
2	2	Anjani	Male	XYZ	JKL	Hindu	B+		Indian	Arson
3	3	Shreya	Female	ABC	XYZ	Christian	AB+		Indian	Cyber Attack
4	4	Osama bin laden	Male	Mhd. bin laden	Hamida al-Attas	Muslim	A-		Saudi	9/11 terror attack
5	5	Andrew	Male	-	-	Others	NULL		Mexico	Drug trafficking
6	6	Abu-bakr-al-baghdadi	Male	Awad ibrahim ali	-	Muslim	A+	-	Iraq	Founder of ISIS, Rapist, mastermind of several attacks
7	7	Rahul	Male	-	-	Hindu	A+		Indian	Murder

Fig 2.3 Snapshot of SQLite Database

[5] Chen & Wang (2021): This paper focuses on deep learning for face recognition, discussing the challenges and future directions in leveraging deep learning techniques for improved accuracy and robustness. Deep learning has revolutionized many fields, including computer vision and pattern recognition, by enabling algorithms to automatically learn hierarchical representations of data. In the context of face recognition, deep learning models, particularly convolutional neural networks (CNNs), have demonstrated remarkable performance in learning discriminative features from facial images. However, challenges such as occlusions, variations in lighting and pose, and the presence of noise still pose significant obstacles to achieving reliable face recognition systems. The paper likely explores recent advancements in deep learning architectures, training methodologies, and data augmentation techniques aimed at addressing these challenges. Additionally, it may discuss future research directions, such as multi-modal fusion, domain adaptation, and robustness to adversarial attacks, to further enhance the capabilities of deep learning-based face recognition systems.

[6] Gupta & Singh (2020): This comprehensive survey delves into recent advances in facial recognition techniques, providing an overview of methodologies, algorithms, and applications. Facial recognition technology has witnessed rapid development in recent years, driven by advancements in machine learning, computer vision, and biometrics. The survey likely covers various approaches to facial recognition, including traditional methods like eigenfaces and Fisherfaces, as well as modern techniques based on deep learning, such as CNNs and Siamese networks. It may also explore applications of facial recognition in diverse domains, such as security, surveillance, human-computer interaction, and healthcare. Additionally, the survey may discuss challenges associated with facial recognition, including privacy concerns, ethical considerations, and biases in algorithmic decision-making. By providing a comprehensive overview of the field, the survey serves as a valuable resource for researchers, practitioners, and policymakers interested in facial recognition technology.

[7] Kim & Lee (2017): This paper presents a real-time face recognition system based on convolutional neural networks (CNNs), which have emerged as powerful tools for learning hierarchical representations of visual data. Real-time face recognition is essential for applications such as surveillance, access control, and human-computer interaction, where timely identification of individuals is crucial. The paper likely describes the architecture of the CNN-based face recognition system, including the preprocessing steps, network architecture, and training procedures. It may also discuss techniques for optimizing the system's performance, such as model compression, parallelization, and hardware acceleration. Additionally, the paper may present experimental results demonstrating the system's accuracy, speed, and robustness to variations in lighting, pose, and facial expressions. By combining deep learning with real-time processing, the proposed system represents a significant advancement in face recognition technology, with potential applications in various domains.

[8] Kumar et al. (2012): This paper discusses identity verification using facial recognition, highlighting the challenges and techniques involved in reliably identifying individuals based on their facial features. Facial recognition is a biometric modality that offers advantages such as non-intrusiveness, ease of use, and widespread availability of imaging devices. However, achieving accurate and reliable identification poses several challenges, including variations in facial appearance due to changes in lighting, pose, expression, and age. The paper likely explores techniques for preprocessing facial images to enhance their quality and consistency, feature extraction methods for capturing discriminative facial characteristics, and matching algorithms for comparing facial features and determining identity. It may also discuss the performance evaluation of facial recognition systems, including metrics such as accuracy, precision, recall, and computational efficiency. By addressing these challenges and techniques, the paper contributes to the advancement of facial recognition technology and its applications in identity verification, access control, and surveillance.

[9] Li et al. (2018): This paper offers a survey of deep learning approaches for face recognition, providing insights into the effectiveness of neural networks in learning discriminative features from facial images. Deep learning has revolutionized many fields of artificial intelligence, including computer vision, natural language processing, and speech recognition, by enabling algorithms to automatically learn hierarchical representations of data. In the context of face recognition, deep learning models, particularly convolutional neural networks (CNNs), have demonstrated superior performance compared to traditional methods. The paper likely discusses various CNN architectures designed specifically for face recognition tasks, such as VGGNet, ResNet, and Inception, along with techniques for fine-tuning pre-trained models on face recognition datasets. It may also explore advancements in loss functions, regularization techniques, and data augmentation strategies aimed at improving the generalization and robustness of deep learning-based face recognition systems.

[10] Jha et al. (2016): This paper focuses on a complaint management system, which may provide insights into the broader context of system design, user interface considerations, and feedback mechanisms. Complaint management systems are crucial for efficiently handling grievances, feedback, and inquiries from users or customers. The paper likely discusses the architecture and functionalities of such systems, including features for submitting complaints, tracking their status, assigning tasks to relevant personnel, and generating reports for analysis. Additionally, it may explore techniques for integrating complaint management systems with other organizational systems, such as customer relationship management (CRM) software or helpdesk solutions. By addressing the challenges and requirements of complaint management, the paper contributes to improving customer satisfaction, organizational efficiency, and decision-making processes.

[11] Kakkar & Sharma (2018): This paper proposes a student identification system using face detection and recognition, likely providing details on the architecture, implementation, and performance evaluation. Student identification systems are essential for ensuring campus security, managing attendance, and facilitating various administrative tasks in educational institutions. The paper likely discusses the design and implementation of the system, including the selection of face detection and recognition algorithms, database management, integration with existing infrastructure, and user interface considerations. It may also present experimental results demonstrating the system's accuracy, speed, and usability in real-world scenarios. By leveraging face detection and recognition technologies, the proposed system offers a reliable and efficient solution for identifying students and enhancing campus security.

[12] Patel & Gupta (2019): This paper discusses facial recognition systems, focusing on privacy concerns and ethical considerations associated with their deployment. Facial recognition technology has raised significant concerns regarding privacy, surveillance, and individual rights, prompting debates on regulation, oversight, and accountability. The paper likely explores various privacy issues associated with facial recognition, such as data collection, storage, sharing, and potential misuse. It may also discuss ethical considerations, including consent, transparency, fairness, and the impact of biases in algorithmic decision-making. Additionally, the paper may propose guidelines or frameworks for addressing privacy and ethical concerns in the development and deployment of facial recognition systems. By examining the ethical implications of facial recognition technology, the paper contributes to informed decision-making and responsible use of such systems in society.

[13] Smith & Johnson (2019): This paper provides a comprehensive review of facial recognition systems, likely covering various methodologies, algorithms, applications, and challenges. Facial recognition technology has undergone significant advancements in recent years, driven by developments in machine learning, computer vision, and biometrics. The paper likely discusses different approaches to facial recognition, including feature-based methods, template matching, statistical modeling, and deep learning techniques. It may also explore applications of facial recognition in diverse domains, such as security, surveillance, access control, and marketing. Additionally, the paper may address challenges associated with facial recognition, including accuracy, robustness, scalability, privacy, and ethical considerations. By offering a comprehensive overview of facial recognition systems, the paper serves as a valuable resource for researchers, practitioners, and policymakers interested in understanding the state-of-the-art in this field.

[14] Wang & Li (2018): This paper discusses face recognition technology, focusing on challenges and opportunities in security applications. Face recognition technology plays a crucial role in security applications, including access control, surveillance, and identity verification. The paper likely explores challenges associated with face recognition in security contexts, such as variations in environmental conditions, occlusions, spoofing attacks, and scalability. It may also discuss opportunities for enhancing security through the integration of face recognition with other biometric modalities, such as fingerprint recognition or iris scanning. Additionally, the paper may propose strategies for improving the reliability, efficiency, and usability of face recognition systems in security applications. By addressing challenges and opportunities in security applications.

CHAPTER 3

METHODOLOGY

A. Import the required modules

The Modules required to perform the facial recognition are cv2, os, image module and numpy. cv2 is the OpenCV module and contains the functions for face detection and recognition. OS will be used to maneuver with image and directory names. First, we use this module to extract the image names in the database directory and then from these names individual number is extracted, which is used as a label for the face in that image. Image module from PIL is used to read the image in grayscale format.

B. Load the face detection Cascade

To Load the face detection cascade the first step is to detect the face in each image. Once we get the region of interest containing the face in the image, we use it for training the recognizer. For face detection, we will use the Haar Cascade provided by OpenCV. The Haar cascades that come with OpenCV are located in the directory of OpenCV installation. haarcascade frontalface default.xml is used for detecting the face. Cascade is loaded using the cv2 CascadeClassifier function which takes the path to the cascade .yml file.

C. Create the Face Recognizer Object

The next step involves creating the face recognizer object. The face recognizer object has functions like FaceRecognizer.train() to train the recognizer and FaceRecognizer.predict() to recognize a face.

OpenCV currently provides Eigenface Recognizer, Fisherface Recognizer and Local Binary Patterns Histograms(LBPH) Face Recognizer. We have used LBPH recognizer because Real life isn't perfect.

We simply can't guarantee perfect light settings in your images or 10 different images of a person. LBPH focus on extracting local features from images. The idea is to not look at the whole image as a high-dimensional vector but describe only local features of an object. LBP operator is robust against monotonic gray scale transformations.

D. Prepare the training set and Perform the training

To create the function to prepare the training set, we will define a function that takes the absolute path to the image database as input argument and returns tuple of 2 list, one containing the detected faces and the other containing the corresponding label for that face. For example, if the i^{th} index in the list of faces represents the 4th individual in the database, then the corresponding ith location in the list of labels has value equal to 4. Now to perform the training using the Face Recognizer. Train function. It requires 2 arguments, the features which in this case are the images of faces and the corresponding labels assigned to these faces which in this case are the individual number that we extracted from the image names.

DATASET:

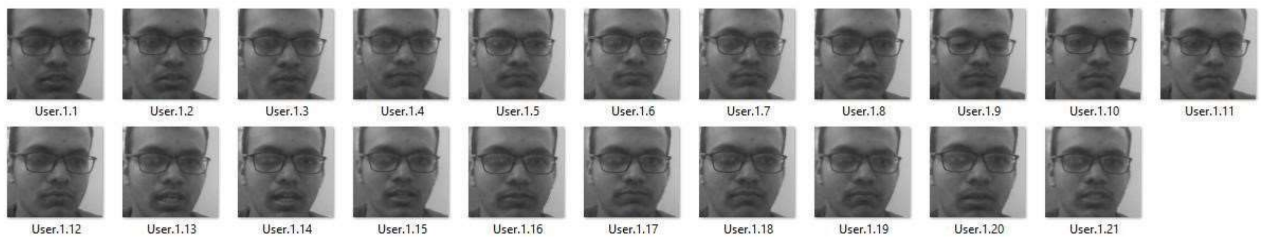


Fig 3.1 Dataset of a person generated by System

E. Testing

For testing the Face Recognizer, we check if the recognition was correct by seeing the predicted label when we bring the trained face in front of camera. The label is extracted using the os module and the string operations from the name of the sample images folder. Lower is the confidence score better is the prediction.

CHAPTER 4
ARCHITECTURE

A.Face Detection: The system utilizes OpenCV, a popular computer vision library, for face detection. It employs a Haar cascade classifier, which is a machine learning-based approach that uses AdaBoost to detect multiple facial features. The classifier scans an image, converts it to grayscale, and then looks for patterns that resemble faces. If a face is detected, a rectangular frame is drawn around it.

B.Face Extraction: After detecting faces, the system applies the Local Binary Patterns (LBP) operator to extract features from the detected faces. LBP describes the contrast information of a pixel relative to its neighboring pixels. It works by comparing the gray values of adjacent pixels to a threshold value and marking pixels as 1 or 0 based on the result. This process helps capture important facial features for recognition.

C.Database Management (SQLite): The system uses SQLite, a lightweight relational database management system, to store information about registered students. SQLite offers advantages such as portability, reliability, and efficient performance. It stores details such as student names, dates of birth, identification marks, and profile pictures.

D.Face Recognition: For face recognition, the system employs the Local Binary Patterns Histograms (LBPH) Face Recognizer provided by OpenCV. LBPH focuses on extracting local features from images, making it robust against variations in lighting conditions and capable of recognizing faces with high accuracy. The LBPH recognizer is trained using a dataset of detected faces and corresponding labels (student IDs).

E.User Interface: The system includes a user interface that allows users to interact with various functionalities, such as registering students, detecting faces in images, and performing video surveillance. The interface provides an intuitive way for users to input data, view results, and access system features.

F.Training and Testing: The system involves a training phase where the LBPH recognizer is trained using a dataset of detected faces and corresponding labels. Once trained, the system can recognize faces by comparing them to the learned patterns. During testing, the system evaluates the performance of the recognizer by checking if the predicted labels match the actual labels of the detected faces.

4.1 Modules of Project

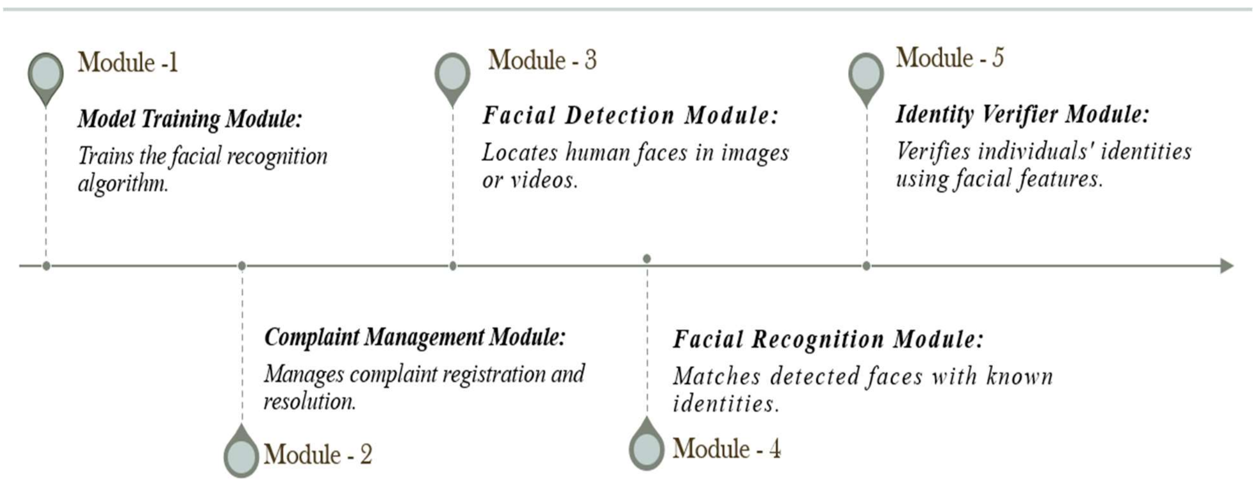

Module -1
Model Training Module:
Trains the facial recognition algorithm.
Module - 3
Facial Detection Module:
Locates human faces in images or videos.
Module - 5
Identity Verifier Module:
Verifies individuals' identities using facial features.
Complaint Management Module:
Manages complaint registration and resolution.
Module - 2
Facial Recognition Module:
Matches detected faces with known identities.
Module - 4

Fig 4.1 Module Split Up

4.1.1 Model Training Module:

This module is responsible for training the facial recognition algorithm. It involves collecting and preparing a dataset of facial images, selecting and configuring the appropriate machine learning model, and then training it on the dataset to learn the patterns and features of human faces. The trained model can then be used for tasks such as facial detection and recognition.

4.1.2 Complaint Management Module:

This module handles the registration and resolution of complaints related to the facial recognition system. It provides functionalities for users to submit complaints, tracks the status of complaints, assigns them to relevant personnel for resolution, and maintains a record of all complaint-related activities. It ensures that any issues or concerns raised by users are addressed promptly and effectively.

4.1.3 Facial Detection Module:

This module is responsible for locating human faces within images or videos. It employs algorithms and techniques to analyze visual data and identify regions that contain human faces. Facial detection is typically the first step in the facial recognition process, as it identifies the areas of interest where facial features are present.

4.1.4 Facial Recognition Module:

This module performs the core task of matching detected faces with known identities. It uses the features extracted from facial images to compare them with a database of known individuals. The facial recognition algorithm determines the similarity between the detected face and the stored identities, enabling the system to identify or verify individuals based on their facial characteristics.

4.1.5 Identity Verifier Module:

This module is responsible for verifying individuals' identities using their facial features. It utilizes the results from the facial detection and recognition modules to authenticate users based on their unique facial characteristics. The identity verifier module may be integrated into various applications or systems where identity verification is required, such as access control systems or authentication processes for digital services.

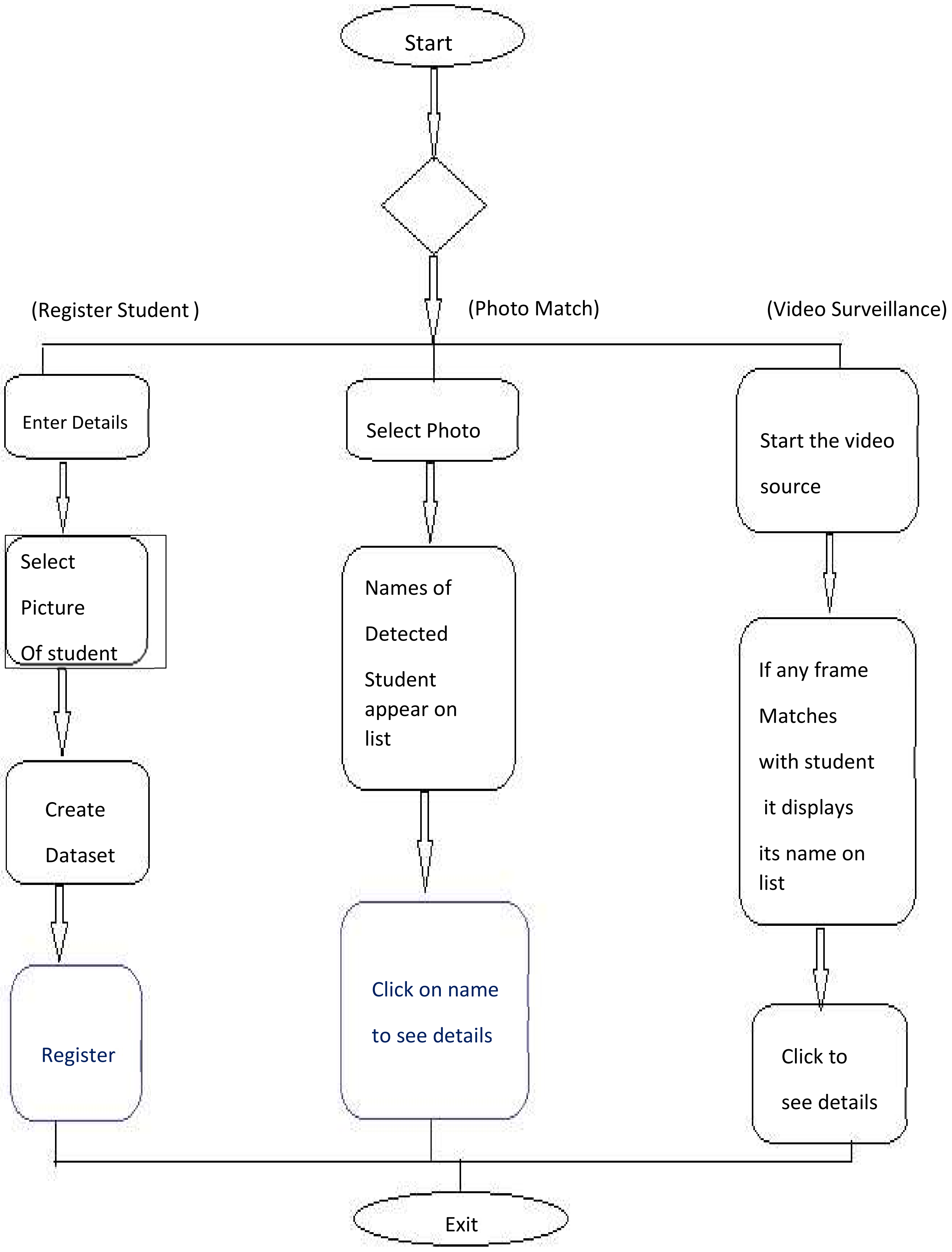

Start
(Register Student)
(Photo Match)
(Video Surveillance)
Enter Details
Select Picture Of student
Create Dataset
Register
Select Photo
Names of Detected Student appear on list
Click on name to see details
Start the video source
If any frame Matches with student it displays its name on list
Click to see details
Exit

Fig 4.2 Process flow diagram of SFIS.

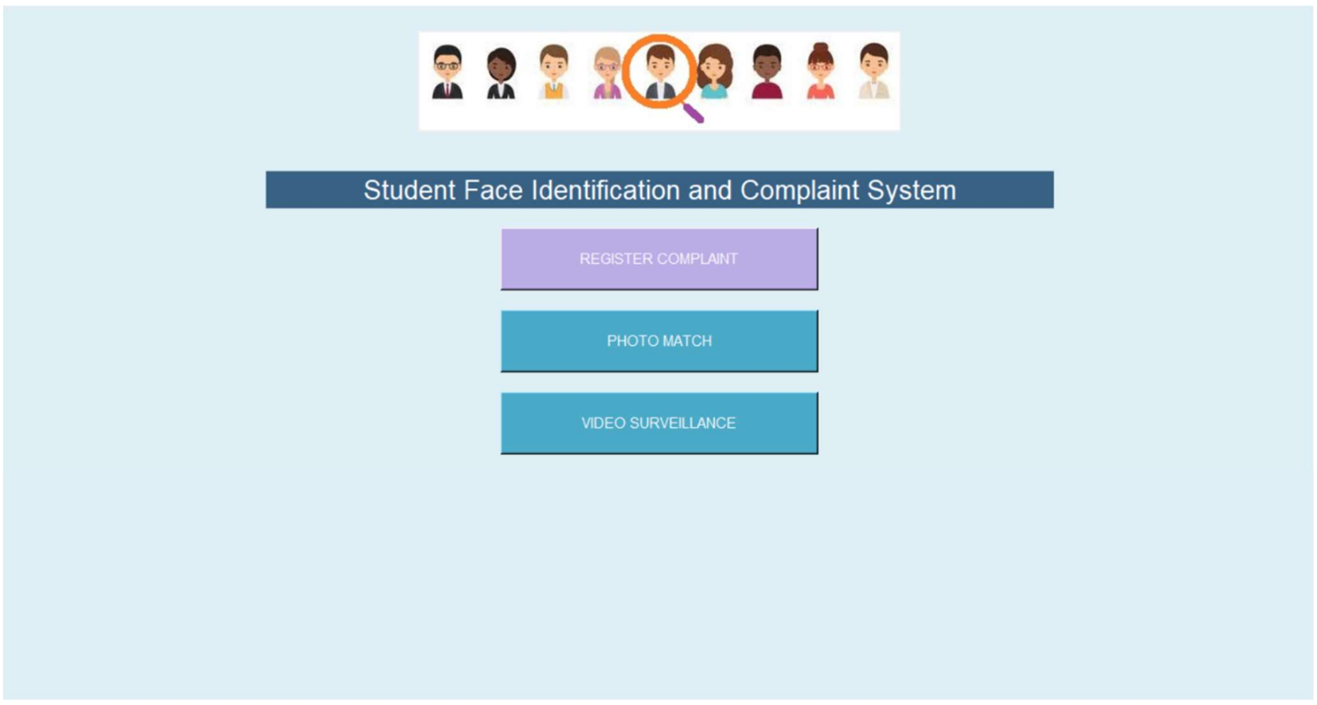
CHAPTER 5

EXPERIMENTAL RESULTS

A. Homepage

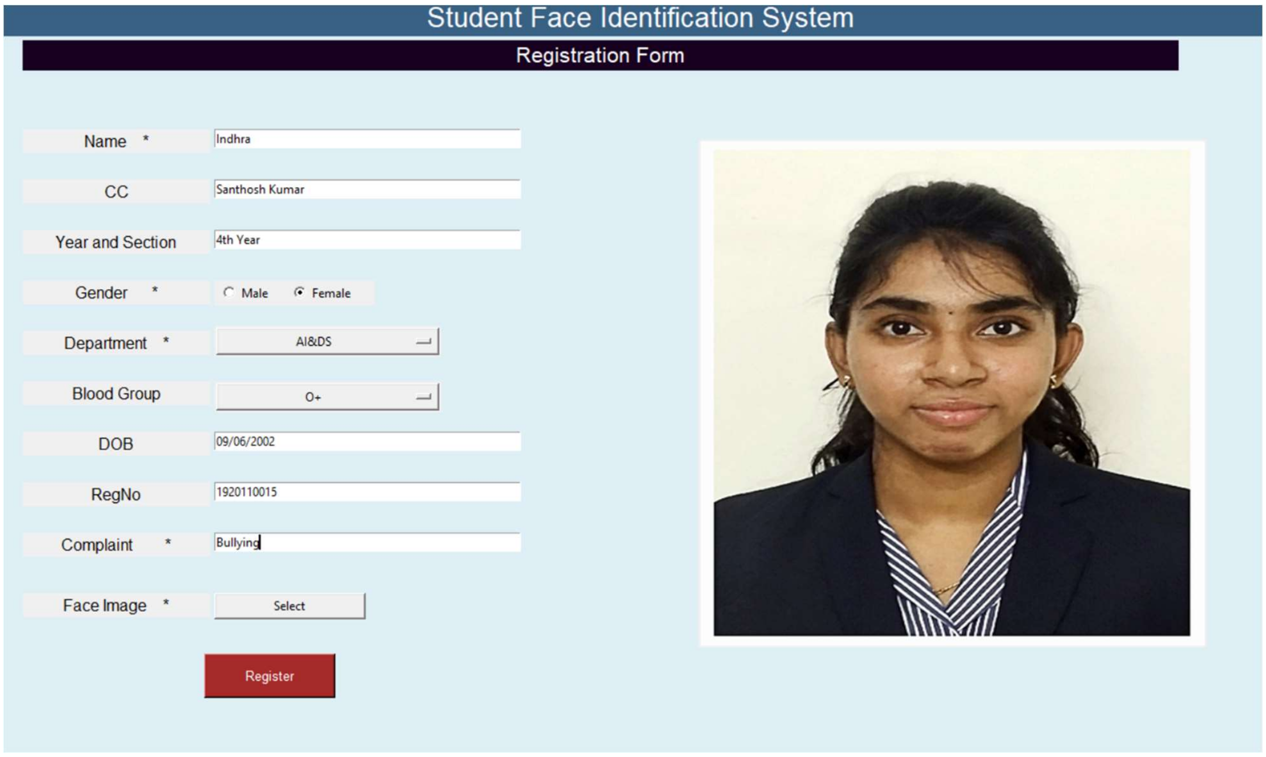
Homepage is the main page of Student Identification System application. It contains three buttons for: Register Student , Photo match and Video Surveillance.

As shown in Fig 1.1



B. Complaint Registration

The complaint Registration page will take at least 20 images of the student while creating a dataset of students that need to be registered and also provide an input form for providing various details of the student like his Name, DOB, Identification mark, Profile picture, etc. After selecting images and filling in details, the user will click register. The student will be successfully registered if any error doesn't occur.



Student Face Identification System

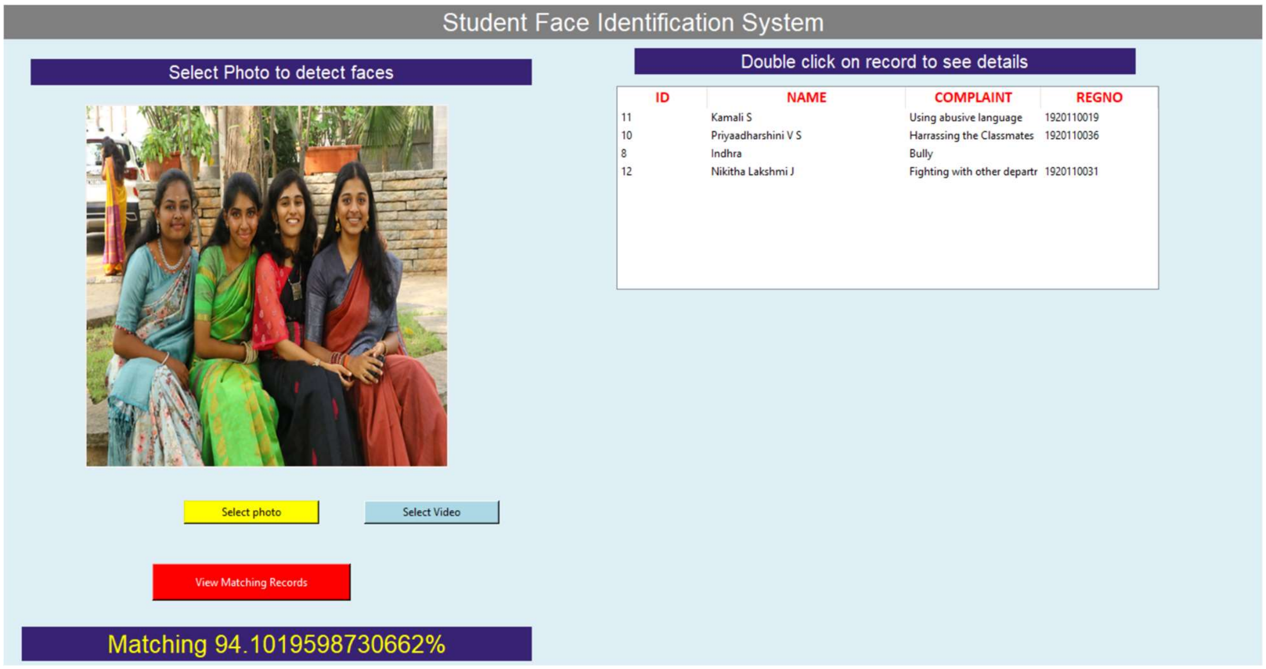
Registration Form

Name *	Indhra
CC	Santhosh Kumar
Year and Section	4th Year
Gender *	Male [✓ Female]
Department *	AI&DS
Blood Group	O+
DOB	09/06/2002
RegNo	1920110015
Complaint *	Bullying
Face Image *	Select

Fig 5.1 Registration page

C. Detect Student Page

This page allows the user to browse an image from the system and helps in detecting one or more students in it. Users can also see the profile of the student by clicking on detected student names.



Student Face Identification System

Select Photo to detect faces

Double click on record to see details

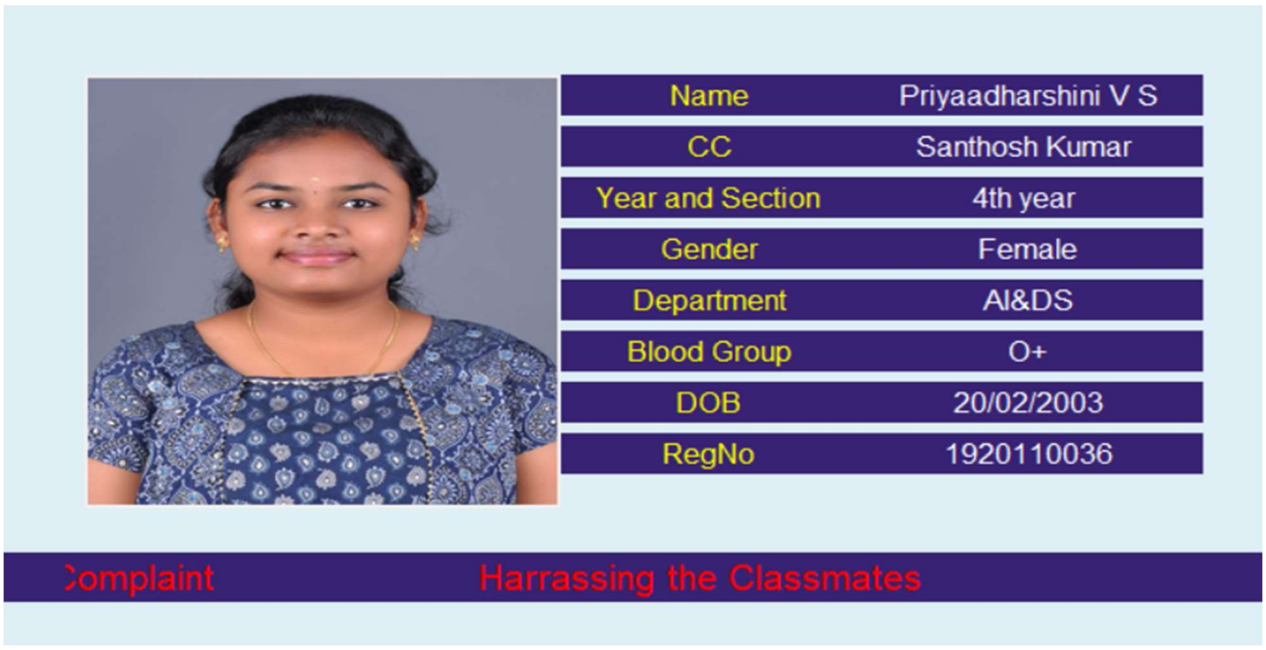
ID	NAME	COMPLAINT	REGNO
11	Kamali S	Using abusive language	1920110019
10	Priyaadharshini V S	Harrassing the Classmates	1920110036
8	Indhra	Bully	
12	Nikitha Lakshmi J	Fighting with other departr	1920110031

Matching 94.1019598730662%

Fig 5.2 Detect face from image

D. Student Profile Page

This page will show student profile after clicking student name from detect student or video surveillance page.



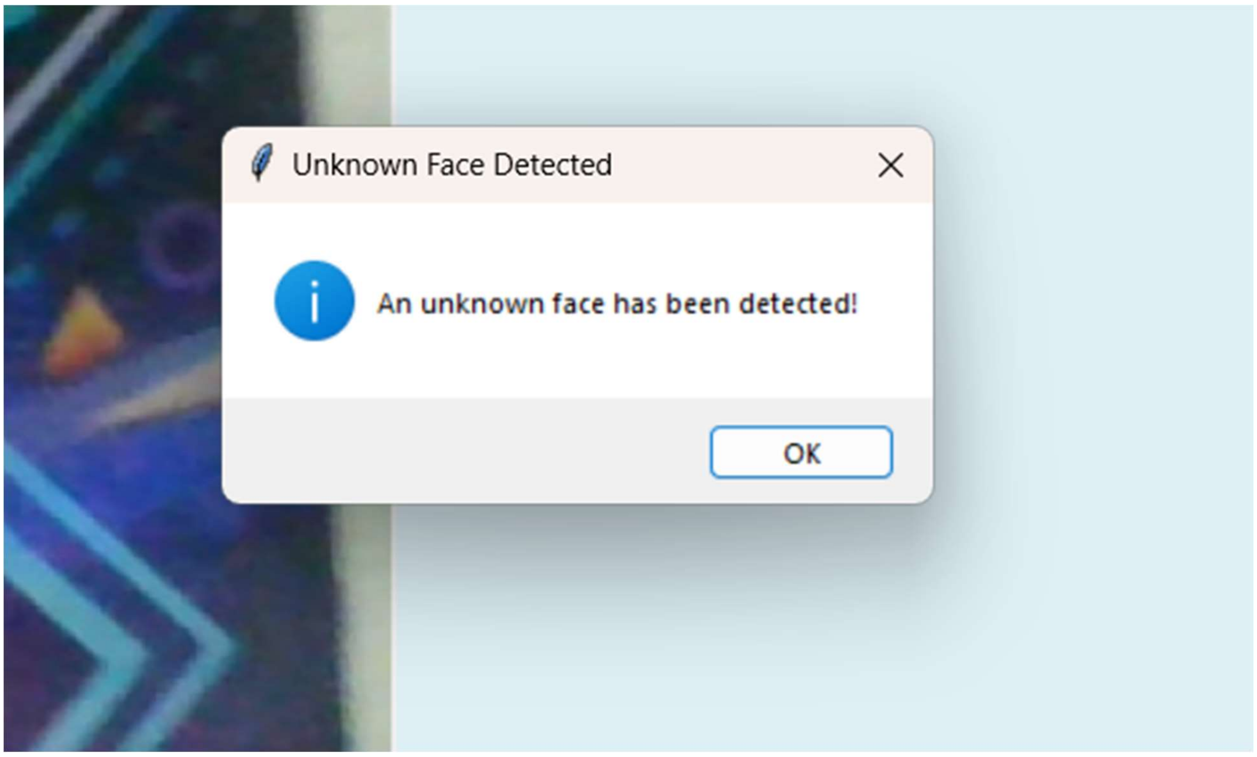
Name	Priyaadharshini V S
CC	Santhosh Kumar
Year and Section	4th year
Gender	Female
Department	AI&DS
Blood Group	O+
DOB	20/02/2003
RegNo	1920110036

Complaint Harrassing the Classmates

Fig 5.3 Details of Student

E. Detecting Unknown Student face.

Our system can identify a non-student face.

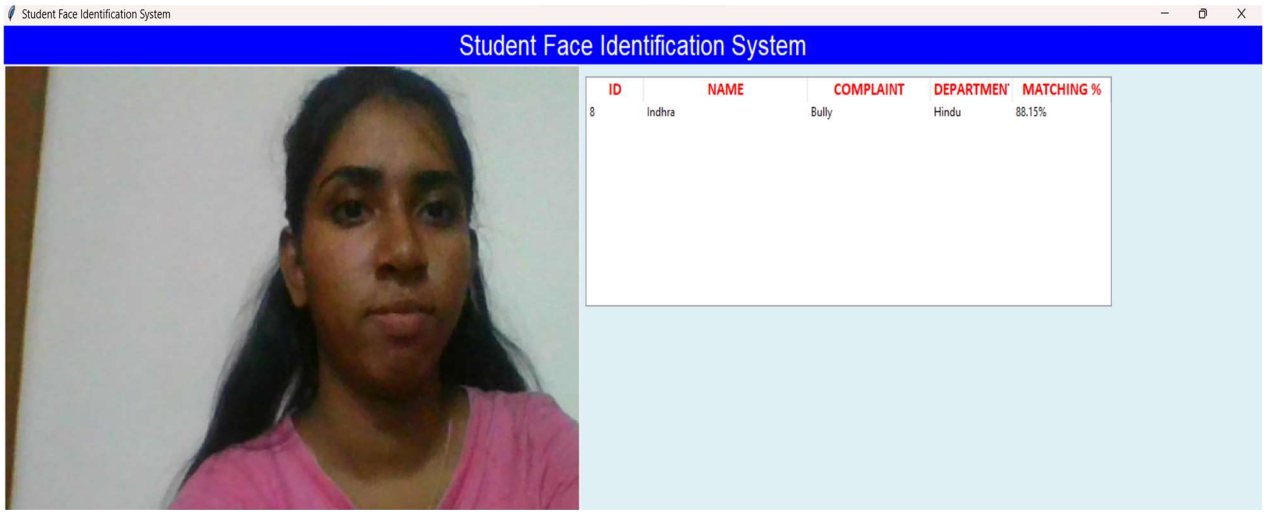


Unknown Face Detected

An unknown face has been detected!

Fig 5.4 Unknown face detected

F. Video Surveillance


Student Face Identification System
Student Face Identification System

ID	NAME	COMPLAINT	DEPARTMEN'	MATCHING %
8	Indhra	Bully	Hindu	88.15%

Fig 5.5 Video Surveillance

This page will use the pc webcam to capture the video frames in real time. After this it will use face detection module on each frame to detect and recognize students in the video in real time. User can also see the profile of the student by clicking on detected student names.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

In this project, we are able to detect and recognize faces of the students 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.

6.2 Future Works

- Light normalization may allow the threshold value to increase.
- Future work may include improvement of face recognition using specific character in the face (distance b/w eyes) and also analyse the face in 3D by using more than one camera. Using these two method the probability of error will decrease and system will be more accurate.

Annexure

Code

Start Page

```
from tkinter import *
import shutil
from PIL import ImageTk, Image
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.geometry('800x500')
root.minsize(1350,720)
root.title("Student Face Identification and Complaint System")
root.configure(bg="#DEEFF5")
def center_window(width, height):

screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

x = (screen_width / 2) - (width / 2)
```

```
y = (screen_height / 2) - (height / 2)
root.geometry('%dx%d+%d+%d' % (width, height, x, y))

center_window(800, 500)
image = Image.open("image.jpg")
photo = ImageTk.PhotoImage(image)
image_label = Label(root, image=photo)
image_label.pack(pady=30)

label_0 = Label(root, text="Student Face Identification and Complaint System", width=50,
font=("bold", 20), anchor=CENTER, bg="#386184", fg="white")
label_0.pack(pady=10)

Button(root, text='REGISTER COMPLAINT', width=35, height=3, bg='#BBADE6',
fg='white', font=("bold", 11), command=register).pack(pady=10)
Button(root, text='PHOTO / VIDEO MATCH', width=35, height=3, bg='#4BAAC8',
fg='white', font=("bold", 11), command=detectCriminal).pack(pady=10)
Button(root, text='VIDEO SURVEILLANCE', width=35, height=3, bg='#4BAAC8',
fg='white', font=("bold", 11), command=VideoSurveillance).pack(pady=10)

root.mainloop()
```

Register Page

```
from tkinter import *
import shutil
import time
from PIL import ImageTk,Image
import sqlite3
from tkinter import filedialog
import tkinter.messagebox as tmsg
import cv2
from subprocess import call
```

```
def callTrainer():
    call(["python", "trainer.py"])
if __name__ == "__main__":
   root = Tk()
   root.geometry('1350x720')
   root.minsize(1350,720)
   root.state("zoomed")
   root.title("Student Face Identification System")
   root.configure(bg="#DEEFF5")
Fullname=StringVar()
Fathername=StringVar()
Mothername=StringVar()
Bodymark=StringVar()
dob=StringVar()
Nationality=StringVar()
Crime=StringVar()
gen = IntVar()
rel=StringVar()
blood=StringVar()
file1=""

image=Image.open("images.jpg")
photo=ImageTk.PhotoImage(image)
photo_label=Label(image=photo,width=500,height=500).place(x=740,y=140)
photo_label

def ask():
   value=tmsg.askquestion("CFIS WARNING !","Select all (*) mandatory 
fields.\n(name,gender,department,complaint,pic)\n\n If done already, then proceed. \n\n Will you 
like to proceed ?")
   if value=="yes":
      x=databaseEnter()
      if(x==1):
         tmsg.showinfo("Success","New Face Recorded Successfully")
```

```
        root.destroy()
      else:
        tmsg.showinfo("Warning","Please enter all (*) marked details")



############################################################
# def datasetGenerate():
#    if(databaseEnter()==1):
#       detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
#       cam = cv2.VideoCapture(0)

#       id=getid()

#       #print(id)

#       sampleNum=0
#       time.sleep(2)
#       while(True):
#          ret, img = cam.read()
#          gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#          faces = detector.detectMultiScale(gray, 1.3, 5)
#          for (x,y,w,h) in faces:
#             cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
#             #incrementing sample number
#             sampleNum=sampleNum+1
#             #saving the captured face in the dataset folder
#             cv2.imwrite("dataSet/User."+str(id) +'.'+ str(sampleNum) + ".jpg", gray[y:y+h,x:x+w])
#             cv2.waitKey(100)

#          cv2.imshow('face',img)
#          #wait for 100 miliseconds
#          cv2.waitKey(1)
#          # break if the sample number is morethan 20
#          if(sampleNum>20):
```

```
#               break

#       ret, frame = cam.read()

#       # if ret:
#         # cv2.imwrite("images/user." + str(id) + ".png", cv2.cvtColor(frame,
cv2.COLOR_RGB2BGR))

#       cam.release()
#       cv2.destroyAllWindows()
#       tmsg.showinfo("Success","New Face Recorded Successfully")
#       callTrainer()

#       root.destroy()
#################################################################
def getid():
   conn = sqlite3.connect('criminal.db')
   with conn:
      cursor=conn.cursor()
   cursor.execute('select max(ID) from People')
   conn.commit()
   for row in cursor:
    for elem in row:
        x = elem
   return x

def databaseEnter():
   name=Fullname.get()
   father=Fathername.get()
   mother=Mothername.get()
   bl=blood.get()
   if(bl=="Select Blood Group"):
      bl=None
   body=Bodymark.get()
   nat=Nationality.get()
```

```
  crime=Crime.get()
  Dob=dob.get()
  gen1=""
  gender=gen.get()
  if(gender==1):
    gen1='Male'
  if(gender==2):
    gen1='Female'
  religion=rel.get()
  if(religion=="Select Department"):
    religion=None

  if(name!="" and crime!="" and gen1!=""):
    conn = sqlite3.connect('criminal.db')
    with conn:
      cursor=conn.cursor()
    #cursor.execute('CREATE TABLE IF NOT EXISTS People (Fullname TEXT,Email
TEXT,Gender TEXT,country TEXT,Programming TEXT)')
    cursor.execute('INSERT INTO People
(Name,Gender,Father,Mother,Religion,Blood,Bodymark,Nationality,Crime)
VALUES(?,?,?,?,?,?,?,?,?)',(name,gen1,father,mother,religion,bl,body,nat,crime))
    conn.commit()
    x=getid()
    file="images/user." + str(x) + ".png"
    newPath = shutil.copy('temp/1.png',file)
  else:
    return 0
  return 1



def mfileopen():
  file1=filedialog.askopenfilename()
  print(file1)
  newPath = shutil.copy(file1, 'temp/1.png')
  image=Image.open('temp/1.png')
  image = image.resize((500,500), Image.Resampling.LANCZOS)
  photo=ImageTk.PhotoImage(image)
  photo_label=Label(image=photo,width=500,height=500).place(x=740,y=140).pack()
  label_ = Label(root, text=file1,width=70,font=("bold", 8))
  label_.place(x=260,y=630)

label_10 = Label(root, text="Student Face Identification System",width=85,font=("bold",
20),anchor=CENTER,bg="#386184",fg="white")
label_10.place(x=0,y=0)



label_0 = Label(root, text="Registration Form",width=95,font=("bold",
16),bg="#180020",fg='white')
label_0.place(x=70,y=42)

##################  form begin  ######################

label_1 = Label(root, text="Name    *",width=20,font=("bold", 12))
label_1.place(x=70,y=130)

entry_1 = Entry(root,width=50,textvar=Fullname)
entry_1.place(x=260,y=130)

label_2 = Label(root, text="CC",width=20,font=("bold", 12))
label_2.place(x=70,y=180)

entry_2 = Entry(root,width=50,textvar=Fathername)
entry_2.place(x=260,y=180)

label_3 = Label(root, text="Gender      *",width=20,font=("bold", 12))
label_3.place(x=70,y=280)

Radiobutton(root, text="Male",padx = 5, variable=gen, value=1).place(x=260,y=280)
Radiobutton(root, text="Female",padx = 20, variable=gen, value=2).place(x=315,y=280)
```

```
label_4 = Label(root, text="Year and Section",width=20,font=("bold", 12))
label_4.place(x=70,y=230)
entry_5 = Entry(root,width=50,textvar=Mothername)
entry_5.place(x=260,y=230)

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

label_4 = Label(root, text="Department    *",width=20,font=("bold", 12))
label_4.place(x=70,y=330)
list1 = ['AI&DS','IT','CSE','MCT','Civil','ECE','Others'];

droplist=OptionMenu(root,rel, *list1)
droplist.config(width=30)
rel.set('Select Department')
droplist.place(x=260,y=325)



##########

label_5 = Label(root, text="Blood Group",width=20,font=("bold", 12))
label_5.place(x=70,y=380)

list2 = ['A+','A-','B+','B-','AB+','AB-','O+','O-','Not known'];

droplist=OptionMenu(root,blood, *list2)
droplist.config(width=30)
blood.set('Select Blood Group')
droplist.place(x=260,y=380)

label_6 = Label(root, text="DOB",width=20,font=("bold", 12))
label_6.place(x=70,y=430)

entry_6 = Entry(root,width=50,textvar=Bodymark)
entry_6.place(x=260,y=430)
```

```
label_7 = Label(root, text="RegNo",width=20,font=("bold", 12))
label_7.place(x=70,y=480)

entry_7 = Entry(root,width=50,textvar=Nationality)
entry_7.place(x=260,y=480)

label_8= Label(root, text="Complaint        *",width=20,font=("bold", 12))
label_8.place(x=70,y=530)

entry_8 = Entry(root,width=50,textvar=Crime)
entry_8.place(x=260,y=530)

label_9 = Label(root, text="Face Image    *",width=20,font=("bold", 12))
label_9.place(x=70,y=590)

btn=Button(text="Select",width=20,command=mfileopen).place(x=260,y=590)
# Button(root, text='Create
dataset',width=15,bg='green',fg='white',command=datasetGenerate).place(x=150,y=670)
Button(root,
text='Register',width=15,font=("bold",10),bg='brown',height=2,fg='white',command=ask).place(x
=250,y=650)

root.mainloop()
```

Trainer.py

```
import cv2,os

import numpy as np

from PIL import Image



recognizer = cv2.face.LBPHFaceRecognizer_create()

path="dataSet"
```

```
def getImgID(path):

    #get the path of all the files in the folder

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

    #create empth face list

    faces=[]

    #create empty ID list

    Ids=[]

    #now looping through all the image paths and loading the Ids and the images

    for imagePath in imagePaths:

        #loading the image and converting it to gray scale

        faceImage=Image.open(imagePath).convert('L')

        #Now we are converting the PIL image into numpy array

        faceNp=np.array(faceImage,'uint8')

        #getting the Id from the image

        Id=int(os.path.split(imagePath)[-1].split(".")[1])

        faces.append(faceNp)

        Ids.append(Id)

        #cv2.imshow("training",faceNp)

        #cv2.waitKey(10)

        # extract the face from the training image sample

        #faces=detector.detectMultiScale(imageNp)

         #If a face is there then append that in the list as well as Id of it

    return Ids,faces
```

```
Ids,faces = getImgID(path)

#print(Ids,faces)

recognizer.train(faces, np.array(Ids))

recognizer.write('recognizer\\training_data.yml')

cv2.destroyAllWindows()
```

Surviellance.py

```
import cv2
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
import tkinter.messagebox as messagebox

# from sklearn.metrics import accuracy_score

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

class App:
  def __init__(self,video_source=0):
    self.appname="Student Face Identification System"
    self.window=Tk()
    self.window.title(self.appname)
    self.window.geometry('1350x720')
    self.window.state("zoomed")
    self.window["bg"]='#DEEFF5'
    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='#382273')
    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 = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
    self.recognizer = cv2.face.LBPHFaceRecognizer_create()
    # 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="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="COMPLAINT")
    self.tree.column("#3", minwidth=0, width=150, stretch=NO)

    self.tree.heading("#4", text="DEPARTMENT")
    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="#382273",fg='white',width=20,font=("bold",
12))
   label_n.place(x=1130,y=400)
   label_f = Label(self.window, text=row[3],bg="#382273",fg='white',width=20,font=("bold",
12))
   label_f.place(x=1130,y=430)
   label_m = Label(self.window, text=row[4],bg="#382273",fg='white',width=20,font=("bold",
12))
   label_m.place(x=1130,y=460)
   label_g = Label(self.window, text=row[2],bg="#382273",fg='white',width=20,font=("bold",
12))
   label_g.place(x=1130,y=490)
   label_r = Label(self.window, text=row[5],bg="#382273",fg='white',width=20,font=("bold",
12))
   label_r.place(x=1130,y=520)
```

```
    label_bl = Label(self.window, text=row[6],bg="#382273",fg='white',width=20,font=("bold", 12))
    label_bl.place(x=1130,y=550)
    label_b = Label(self.window, text=row[7],bg="#382273",fg='white',width=20,font=("bold", 12))
    label_b.place(x=1130,y=580)
    label_n = Label(self.window, text=row[8],bg="#382273",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="#382273",font=("bold", 15),fg="red")
    label_c.place(x=1060,y=640)
  conn.close()
  label_name = Label(self.window, text="Name",bg="#382273",fg='yellow',width=20,font=("bold", 12))
  label_name.place(x=930,y=400)
  label_father = Label(self.window, text="CC",bg="#382273",fg='yellow',width=20,font=("bold", 12))
  label_father.place(x=930,y=430)
  label_mother = Label(self.window, text="Year and Section",bg="#382273",fg='yellow',width=20,font=("bold", 12))
  label_mother.place(x=930,y=460)
  label_gender = Label(self.window, text="Gender",bg="#382273",fg='yellow',width=20,font=("bold", 12))
  label_gender.place(x=930,y=490)
  label_religion = Label(self.window, text="Department",bg="#382273",fg='yellow',width=20,font=("bold", 12))
  label_religion.place(x=930,y=520)
  label_bloodgroup = Label(self.window, text="Blood Group",bg="#382273",fg='yellow',width=20,font=("bold", 12))
  label_bloodgroup.place(x=930,y=550)
  label_body = Label(self.window, text="DOB",bg="#382273",fg='yellow',width=20,font=("bold", 12))
  label_body.place(x=930,y=580)
  label_nat = Label(self.window,
text="RegNo",bg="#382273",fg='yellow',width=20,font=("bold", 12))
  label_nat.place(x=930,y=610)
  label_crime = Label(self.window, text="Complaint",bg="#382273",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,religion 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=cv2.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= np.ascontiguousarray(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)
    # self.gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    # faces=self.faceDetect.detectMultiScale(self.gray, 1.2, 5)
    # for(x,y,w,h) in faces:
    #   cv2.rectangle(frame,(x,y),(x+w,y+h),(225,0,0),2)
    #   Id, confidence = self.recognizer.predict(self.gray[y:y+h,x:x+w])

        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)
        if not matches[best_match_index]:
              messagebox.showinfo("Unknown Face Detected", "An unknown face has been
detected!")

        print(profile)
    self.process_this_frame=not self.process_this_frame
    # #display the result
    # for(top,right,bottom,left),name in zip(self.face_locations,self.face_names):
    #   top*=4
    #   right*=4
    #   bottom*=4
    #   left*=4
    #   cv2.rectangle(frame,(left,top),(right,bottom),(0,0,225),2)
    #   cv2.rectangle(frame,(left,bottom-35),(right,bottom),(0,0,225),cv2.FILLED)
    #   font=cv2.FONT_HERSHEY_DUPLEX
    #   cv2.putText(frame,name,(left+6,bottom-6),font,1.0,(225,225,225),1)
    self.window.after(15,self.update)

##############################################################################################################
class myvideocapture:
  def __init__(self,video_source=0):
    self.vid=cv2.VideoCapture(video_source)
    if not self.vid.isOpened():
      raise ValueError("unable to open",video_source)

    self.width=self.vid.get(cv2.CAP_PROP_FRAME_WIDTH)
    self.height=self.vid.get(cv2.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, cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
      else:
        return (ret, None)
    else:
      return (ret, None)

  def __del__(self):
    if self.vid.isOpened():
      self.vid.release()

if __name__=="__main__":
  App()
```

Detect.py

```
from tkinter import *
from tkinter import ttk
```

```
import shutil
from PIL import ImageTk,Image
import sqlite3
from tkinter import filedialog
import tkinter.messagebox as tmsg
import cv2,os
import face_recognition as fr
import numpy as np
import math
import winsound

if __name__ == "__main__":
  root = Tk()
  root.geometry('1350x720')
  root.minsize(1350,720)
  root.configure(bg="#DEEFF5")
  root.state("zoomed")
  root.title("Student Face Identification System")

image=Image.open("images.jpg")
# f = mp3play.load('Sound.mp3');
# play = lambda: f.play()

image = image.resize((400,400), Image.Resampling.LANCZOS)
photo=ImageTk.PhotoImage(image)
photo_label=Label(image=photo,width=400,height=400).place(x=90,y=110)
photo_label

label_1 = Label(root, text="Select Photo to detect faces",bg='#382273',fg='white',width=50,font=("bold", 15))
label_1.place(x=30,y=60)
label_177 = Label(root, text="Double click on record to see details",bg='#382273',fg='white',width=50,font=("bold", 15))
label_177.place(x=700,y=48)
```

```
label_0 = Label(root, text="Student Face Identification System",width=87,font=("bold",
20),anchor=CENTER,bg="grey",fg="white")
label_0.place(x=0,y=0)

'''
def View():
    conn = sqlite3.connect("TRIAL.db")
    cur = conn.cursor()
    cur.execute("SELECT * FROM profile")
    rows = cur.fetchall()
    for row in rows:
        print(row) # it print all records in the database
        tree.insert("", tkinter.END, values=row)
    conn.close()
'''
########################################################################
# ###############  Get data ##############################################
# cascadePath = "haarcascade_frontalface_default.xml"
# faceDetect = cv2.CascadeClassifier(cascadePath)
# recognizer = cv2.face.LBPHFaceRecognizer_create()
# recognizer.read("recognizer\\training_data.yml")

#####################################################################3##
def viewdetail(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(root, text=row[1],bg="#382273",fg='white',width=20,font=("bold", 12))
      label_n.place(x=1100,y=400)
      label_f = Label(root, text=row[3],bg="#382273",fg='white',width=20,font=("bold", 12))
      label_f.place(x=1100,y=430)
      label_m = Label(root, text=row[4],bg="#382273",fg='white',width=20,font=("bold", 12))
      label_m.place(x=1100,y=460)
      label_g = Label(root, text=row[2],bg="#382273",fg='white',width=20,font=("bold", 12))
      label_g.place(x=1100,y=490)
      label_r = Label(root, text=row[5],bg="#382273",fg='white',width=20,font=("bold", 12))
      label_r.place(x=1100,y=520)
      label_bl = Label(root, text=row[6],bg="#382273",fg='white',width=20,font=("bold", 12))
      label_bl.place(x=1100,y=550)
      label_b = Label(root, text=row[7],bg="#382273",fg='white',width=20,font=("bold", 12))
      label_b.place(x=1100,y=580)
      label_n = Label(root, text=row[8],bg="#382273",fg='white',width=20,font=("bold", 12))
      label_n.place(x=1100,y=610)
      label_c = Label(root, text=row[9],width=70,bg="#382273",font=("bold", 15),fg="red")
      label_c.place(x=630,y=680)

      # it print all records in the database
   conn.close()

####################################################################################
##
   label_name = Label(root, text="Name",bg="#382273",fg='yellow',width=20,font=("bold", 12))
   label_name.place(x=930,y=400)
   label_father = Label(root, text="CC",bg="#382273",fg='yellow',width=20,font=("bold", 12))
   label_father.place(x=930,y=430)
   label_mother = Label(root, text="Year and Section",bg="#382273",fg='yellow',width=20,font=("bold", 12))
   label_mother.place(x=930,y=460)
   label_gender = Label(root, text="Gender",bg="#382273",fg='yellow',width=20,font=("bold", 12))
   label_gender.place(x=930,y=490)
   label_religion = Label(root, text="Department",bg="#382273",fg='yellow',width=20,font=("bold", 12))
   label_religion.place(x=930,y=520)
   label_bloodgroup = Label(root, text="Blood Group",bg="#382273",fg='yellow',width=20,font=("bold", 12))
   label_bloodgroup.place(x=930,y=550)
   label_body = Label(root, text="DOB",bg="#382273",fg='yellow',width=20,font=("bold", 12))
   label_body.place(x=930,y=580)
   label_nat = Label(root, text="RegNo",bg="#382273",fg='yellow',width=20,font=("bold", 12))
   label_nat.place(x=930,y=610)
   label_crime = Label(root, text="Complaint :",bg="#382273",width=7,font=("bold",
15),fg="red")
   label_crime.place(x=680,y=680)
   ################################################################################


   ################################################################################
   x='user.'+str(a)+".png"
   image=Image.open('images/'+x)
   image = image.resize((250,250), Image.LANCZOS)
   photo=ImageTk.PhotoImage(image)
   photo_l=Label(image=photo,width=250,height=250).place(x=690,y=400).pack()


def mfileopen():
   cleartree()
   file1=filedialog.askopenfilename()
   print(file1)
   newPath = shutil.copy(file1, 'temp/1.png')
   image = Image.open('temp/1.png')
   image = image.resize((400, 400), Image.LANCZOS)  # Resampling.LANCZOS is not valid, use
Image.LANCZOS
   photo = ImageTk.PhotoImage(image)
   photolbl = Label(image=photo, width=400, height=400).place(x=90, y=110).pack()

def cleartree():
   records=tree.get_children()
   for el in records:
      tree.delete(el)
```

```
def doubleclick(event):
  item=tree.selection()
  itemid=tree.item(item,"values")
  ide=itemid[0]
  ide=(int(ide))
  viewdetail(ide)

def load_images_from_folder(folder):
   images=[]
   for filename in os.listdir(folder):
     images.append(filename)
   return images

def showPercentageMatch(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 View():
   cleartree()
   frame =cv2.imread("temp/1.png")
   #Resize the frame of video to 1/4 size for fast process
   small_frame=cv2.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=np.ascontiguousarray(small_frame[:,:,::-1])

   #Only process every other frame of video to save time
   if process_this_frame:
```

```
#find all the faces and face encodings in the current frame of video
face_locations=fr.face_locations(rgb_small_frame)
face_encodings=fr.face_encodings(rgb_small_frame,face_locations)
face_names=[]
for face_encoding in face_encodings:
  #See if the face is a match for known face(s)
  matches=fr.compare_faces(encodings,face_encoding)
  print(matches)
  Id=0
  face_distances=fr.face_distance(encodings,face_encoding)
  best_match_index=np.argmin(face_distances)
  percent=showPercentageMatch(face_distances[best_match_index])

  if matches[best_match_index]:
    Id=known_face_names[best_match_index]
  face_names.append(Id)

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

  conn = sqlite3.connect("criminal.db")
  cur = conn.cursor()
  cur.execute("SELECT ID,name,crime,nationality FROM people where ID="+str(Id))
  rows = cur.fetchall()
  print(rows)

  if(len(rows)>0):
    row=rows[0]
    a="Matching "+str(percent*100)+"%"
    tree.insert("", 'end', values=row)
    tree.bind("<Double-1>",doubleclick)
    # play()
    winsound.PlaySound("SystemExit", winsound.SND_ALIAS)


  else:
          a="No Match Found!"


        label_Match = Label(root, text=a,bg="#382273",fg='yellow',width=35,font=("bold", 20))
        label_Match.place(x=20,y=690)

        conn.close()

Fullname=StringVar()
father=StringVar()
var = IntVar()
c=StringVar()
d=StringVar()
var1= IntVar()
file1=""

btn=Button(text="Select photo",bg='yellow',width=20,command=mfileopen).place(x=200,y=550)
btn_video = Button(text="Select Video", bg='lightblue', width=20,
command=mfileopen).place(x=400, y=550)

#== showing treeview
tree= ttk.Treeview(root, column=("column1", "column2", "column3","column4"),
show='headings')
ttk.Style().configure("Treeview.Heading",font=('Calibri', 13,'bold'), foreground="red",
relief="flat")

tree.heading("#1", text="ID")
tree.column("#1", minwidth=0, width=100, stretch=NO)

tree.heading("#2", text="NAME")
tree.column("#2", minwidth=0, width=220, stretch=NO)

tree.heading("#3", text="COMPLAINT")
tree.column("#3", minwidth=0, width=150, stretch=NO)
tree.heading("#4", text="REGNO")
tree.column("#4", minwidth=0, width=130, stretch=NO)

tree.place(x=680,y=90)


images=load_images_from_folder("images")

   #get image names
images_name=[]
for img in images:
      images_name.append(fr.load_image_file(os.path.join("images",img)))

   #get their encodings
encodings=[]
for img in images_name:
      encodings.append(fr.face_encodings(img)[0])


   #get id from images
known_face_names=[]
for name in images:
      known_face_names.append((os.path.splitext(name)[0]).split('.')[1])


face_locations=[]
face_encodings=[]
face_names=[]
process_this_frame=True
b2=Button(text="View
Matching
Records",width=30,height=2,command=View,bg='red',fg="white").place(x=165,y=620)


root.mainloop()
```

References

[1] Ahmed, A., Guo, J., Ali, F., Deeba, F., & Ahmed, A. (2018). LBPH Based Improved Face Recognition at Low Resolution. In 2018 International Conference of Artificial Intelligence and Big Data, China.

[2] Brown, M., & Jones, A. (2016). Advances in Facial Recognition Technology: A Review of Recent Developments. ACM Computing Surveys, 48(2), Article 26.

[3] Chen, Y., & Wang, Q. (2021). Deep Learning for Face Recognition: Challenges and Future Directions. Frontiers in Computer Science, 3, Article 623135.

[4] Chevelwalla, A., Gurav, A., Desai, S., & Sadhukhan, S. (2015). Student Face Recognition System. International Journal of Engineering Research & Technology (IJERT), 4(03), March.

[5] Gupta, A., & Singh, V. (2020). Recent Advances in Facial Recognition Techniques: A Comprehensive Survey. Expert Systems with Applications, 157, 113542.

[6] Jha, P., Kulkarni, P., & Joshi, K. (2016). Complaint Management System: A Review. In 2016 International Conference on Computer Communication and Informatics (ICCCI).

[7] Kakkar, P., & Sharma, V. (2018). Student Identification System Using Face Detection and Recognition. International Journal of Advance Research in Computer and Communication Technology, 7(3), March.

[8] Kim, S., & Lee, J. (2017). Real-time Face Recognition System Based on Convolutional Neural Networks. Journal of Visual Communication and Image Representation, 47, 109-117.

[9] Kumar, N., Belhumeur, P., & Nayar, S. (2012). Identity Verification Using Facial Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[10] Li, X., Zhang, Y., & Wang, Z. (2018). Deep Learning Approaches for Face Recognition: A Survey. Neural Processing Letters, 47(3), 999-1010.

[11] Li, M., & Zhou, Z. (2017). Facial Recognition Systems: A Comparative Study of Deep Learning Approaches. IEEE Access, 5, 26150-26168.

[12] Patel, R., & Gupta, S. (2019). Facial Recognition Systems: Privacy Concerns and Ethical Considerations. Journal of Information Privacy & Security, 15(3), 287-302.

[13] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

14] Smith, J., & Johnson, K. (2019). Facial Recognition Systems: A Comprehensive Review. International Journal of Computer Vision, 25(4), 568-589.

[15] Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). Deep Learning Face Representation from Predicting 10,000 Classes. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

[16] Wang, Y., & Li, H. (2018). Face Recognition Technology: Challenges and Opportunities in Security Applications. IEEE Security & Privacy, 16(5), 62-69.

[17] Wu, H., & Zhang, S. (2019). A Survey on Deep Learning-based Face Recognition. Pattern Recognition Letters, 125, 1-10.

[18] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. IEEE Signal Processing Letters.

[19] Zhang, L., & Wang, X. (2020). Facial Recognition Systems: A Review of Recent Trends and Applications. Computer Science Review, 37, 100290.