PIXEL FORGED DEVICE DRIVER

A PROJECT REPORT

Submitted by

ARAVIND P K [211420104021]

AMBAREESHAN N [211420104015]

ARIN R S [211420104023]

in partial fulfilment for the award for the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **“PIXEL FORGED DEVICE DRIVER”** is the bonafide work of “**ARAVIND P K(211420104021), AMBAREESHAN N(211420104015), ARIN R S (211420104023)”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.L.JABASHEELA,M.E.,Ph.D.,	**Dr. MOHANA PRAKASH TA**
HEAD OF THE DEPARTMENT	**ASSOCIATE PROFESSOR**
DEPARTMENT OF CSE,	DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,	PANIMALAR
ENGINEERING COLLEGE,NASARATHPETTAI,	NASARATHPETTAI,
POONAMALLEE,	POONAMALLEE,
CHENNAI-600 123.	CHENNAI-600 123.

Certified that the above candidate(s) was examined in the End Semester

ProjectViva-Voce Examination held on..........................

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

We **Aravind P K (211420104021), Ambareeshan N (211420104015), Arin R S(211420104023)** hereby declare that this project report titled "**Pixel forged device driver**", under the guidance of **Mr.Sasikumar A N..,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ARAVIND P K

AMBAREESHAN N

ARIN R S

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D**. for his kind words andenthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI,Dr.C.SAKTHIKUMAR,M.E.,Ph.D,** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us withthe necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D**. whofacilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA , M.E.,Ph.D.**, forthe support extended throughout the project.

We would like to thanks to Project Coordinator **DR.G.SENTHIL KUMAR, M.C.A., M.Phil., M.E., M.B.A., Ph.D,** and Project Guide **Dr.A.N.SASIKUMAR M.E, Ph.D**., and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

ARAVIND P K

AMBAREESHAN N

ARIN R S

ABSTRACT

This innovative project harnesses the power of Artificial Intelligence (AI) and Machine Learning (ML) to create a robust system for real-time personnel tracking and identification. By leveraging advanced computer vision algorithms, the system can accurately detect and recognize individuals within a camera feed, enabling seamless monitoring of their entry and exit times. Through sophisticated facial recognition techniques, known individuals are swiftly identified, with their names automatically linked to the corresponding timestamps. The system's adaptability is one of its key strengths. In scenarios where an unknown individual is detected, the system intelligently flags the occurrence as "not matching." This prompts user intervention, allowing personnel to promptly input the unidentified individual's identity. By continuously updating its database with newly provided information, the system enhances its recognition capabilities over time, ensuring improved accuracy and reliability in future detections. Moreover, the system's user interface is designed for intuitive interaction, facilitating streamlined management of personnel movement. Recognized individuals' names are prominently displayed alongside their entry and exit timestamps, empowering administrators to efficiently monitor and track personnel activity in diverse environments such as offices, schools, and public spaces. Beyond its immediate applications in security and access control, this AI and ML-powered system promises broader implications for organizational efficiency and resource optimization.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURE

CHAPTER 1
INTRODUCTION

CHAPTER1

INTRODUCTION

In an era marked by rapid technological advancements, the integration of Artificial Intelligence (AI) and Machine Learning (ML) has revolutionized various aspects of our daily lives. One significant application of these technologies is in the realm of surveillance systems, where the need for efficient and accurate monitoring is paramount. In this context, our project endeavors to leverage AI and ML capabilities to develop a sophisticated surveillance system that focuses on the detection and recognition of individuals using camera feeds.

1.1Background

Traditional surveillance systems often rely on manual monitoring, which is labor-intensive, prone to errors, and lacks real-time responsiveness. With the advent of AI and ML, there has been a paradigm shift towards automated surveillance systems capable of intelligent analysis and decision-making. Our project aims to capitalize on this trend by employing facial detection and recognition techniques to automate the process of personnel tracking within a monitored area.

1.2 Objectives

The primary objective of our project is to design and implement a robust surveillance system that can detect and recognize individuals captured by a camera feed. Specifically, our goals include: Developing an AI-based facial detection and recognition algorithm capable of accurately identifying individuals in real-time. Implementing a database management system to record the entry and exit times of detected individuals. Creating an intuitive user interface for system interaction and management. Evaluating the performance and effectiveness of the developed system through rigorous testing and analysis.

1.3 Scope

The scope of our project encompasses the development of a prototype surveillance system that can be deployed in various settings, including offices, schools, and public spaces. The system will be designed to handle multiple camera feeds simultaneously and support the detection and recognition of multiple individuals within each feed. Additionally, the system will provide functionalities for updating the database with the identities of newly detected individuals and retrieving historical data for analysis.

1.4 Overview Of The System:

The proposed surveillance system will comprise hardware components such as cameras for capturing video feeds, along with software components including AI algorithms for facial detection and recognition, database management systems for data storage, and a user interface for system interaction. Upon detection of individuals, the system will record their entry and exit times, associate them with their identities if known, and prompt for identity input if unknown. This abstract summarizes the key aspects of our project and sets the stage for a detailed exploration of each component in subsequent sections.

CHAPTER 2
LITREATURE REVIEW

CHAPTER2

LITRATURE REVIEW

2.1 LITURATURE REVIEW

In recent years, the integration of Artificial Intelligence (AI) and Machine Learning (ML) technologies has revolutionized various industries, including surveillance and security. One of the prominent applications of AI and ML in this domain is the development of personnel tracking systems, which automate the process of monitoring individuals' movement within a designated area. This literature review aims to provide an overview of existing research, methodologies, challenges, and future directions in the field of AI and ML-based personnel tracking systems. State-of-the-Art Techniques: Facial Recognition: Facial recognition algorithms play a pivotal role in personnel tracking systems, enabling the identification of known individuals from a camera feed. Deep learning techniques, particularly convolutional neural networks (CNNs), have demonstrated remarkable performance in facial recognition tasks (Schroff et al., 2015). Object detection algorithms, such as YOLO (You Only Look Once) and Faster R-CNN (Region-based Convolutional Neural Networks), are commonly employed for detecting and localizing individuals within a video stream (Redmon et al., 2016; Ren et al., 2017). Tracking Algorithms: Once individuals are detected, tracking algorithms, including Kalman filters and deep SORT (Simple Online and Realtime Tracking), are utilized to maintain continuity in tracking across frames (Bewley et al., 2016). Applications and Case Studies: Office Environments Personnel tracking systems find extensive use in office environments for attendance monitoring, access control, and workspace optimization. Research by Kim et al. (2018) demonstrates the effectiveness of AI-based personnel tracking in enhancing workplace security and resource allocation. Educational Institutions: In schools and universities, AI and ML-powered tracking systems are employed for student attendance management, campus security, and behavior analysis. Studies by Jiang et al. (2019) highlight the benefits of such systems in improving campus safety and operational efficiency. Challenges and Limitations:

Privacy Concerns The deployment of AI-based surveillance systems raises significant privacy concerns regarding the collection and storage of individuals' biometric data. Striking a balance between security and privacy remains a critical challenge (Kosinski et al., 2018). Performance in Complex Environments: Personnel tracking systems may face challenges in complex environments with varying lighting conditions, occlusions, and crowded scenes. Addressing these challenges requires robust algorithmic solutions and extensive dataset augmentation (Wang et al., 2020). Future Directions: Multimodal Fusion: Future research directions involve integrating multiple modalities, such as facial recognition, gait analysis, and RFID (Radio Frequency Identification), to enhance the robustness and accuracy of personnel tracking systems (Zhang et al., 2019). Edge Computing: Leveraging edge computing techniques for real-time processing and analysis of surveillance data can alleviate bandwidth constraints and enhance the scalability of personnel tracking systems (Sathya et al., 2021). Conclusion: AI and ML-based personnel tracking systems hold immense potential for enhancing security, efficiency, and resource management in various settings. While significant progress has been made in algorithm development and real-world implementations, ongoing research efforts are needed to address challenges related to privacy, performance, and scalability. By leveraging interdisciplinary approaches and emerging technologies, future advancements in this field are poised to redefine the landscape of surveillance and personnel management. References Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple Online and Realtime Tracking with a Deep Association Metric. In 2016 IEEE International Conference on Image Processing (ICIP) (pp. 3464–3468). IEEE. Jiang, B., Zhang, K., & Cai, Z. (2019). Face Recognition Technology Application in Campus Safety Management. 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). IEEE. Kim, J., Kim, J., Kim, H., & Oh, S. (2018). OfficeMate Smart Attendance Management System Based on Face Recognition. 2018 IEEE International Conference on Consumer Electronics (ICCE). IEEE. Kosinski, M., Wang, Y., Lakkaraju, H., & Leskovec, J. (2018).

CHAPTER 3
SYSTEM ANALYSIS

CHAPTER 3

SYSTEM ANALYSIS

3.1 System Overview

Your system consists of several interconnected components working together to detect and recognize faces: Input: Video feed or images containing faces. Preprocessing: Image preprocessing techniques to enhance face detection and recognition. Face Detection: Algorithms to locate and extract faces from input images or video frames. Feature Extraction: Techniques to extract distinctive features from detected faces. Face Recognition: ML models trained to recognize known faces based on extracted features. Database Management: Storage and retrieval of known faces and their corresponding identities. Output: Displaying results indicating whether the detected faces match known identities or are unknown.

3.2 Functional Requirements

Face Detection

The system must accurately detect faces in input images or video frames.

Face Recognition

It should be able to recognize known faces by comparing extracted features with those in the database.

Database Management

Ability to store, update, and retrieve known faces and their identities.

Real-time Processing

The system should perform face detection and recognition in real-time for efficient monitoring. User Interface: Provide a user-friendly interface to display results and manage the system.

3.3Non-functional Requirements

Accuracy

The system should have high accuracy in both face detection and recognition tasks.

Speed

Real-time or near-real-time processing to handle a continuous stream of input.

Scalability

Ability to scale with increased data or users. Robustness: Tolerance to variations in lighting conditions, facial expressions, and poses.

Security

Ensuring the privacy and security of the stored face data.

3.4 System Architecture

Input Module

Responsible for receiving and preprocessing input images or video frames.

Detection Module

Performs face detection using algorithms like Haar cascades or deep learning-based approaches.

Recognition Module

Utilizes ML models (e.g., CNNs, Siamese networks) for face recognition.

Database Module

Manages storage and retrieval of known faces and identities.

User Interface

Provides a graphical interface for interaction and displaying results.

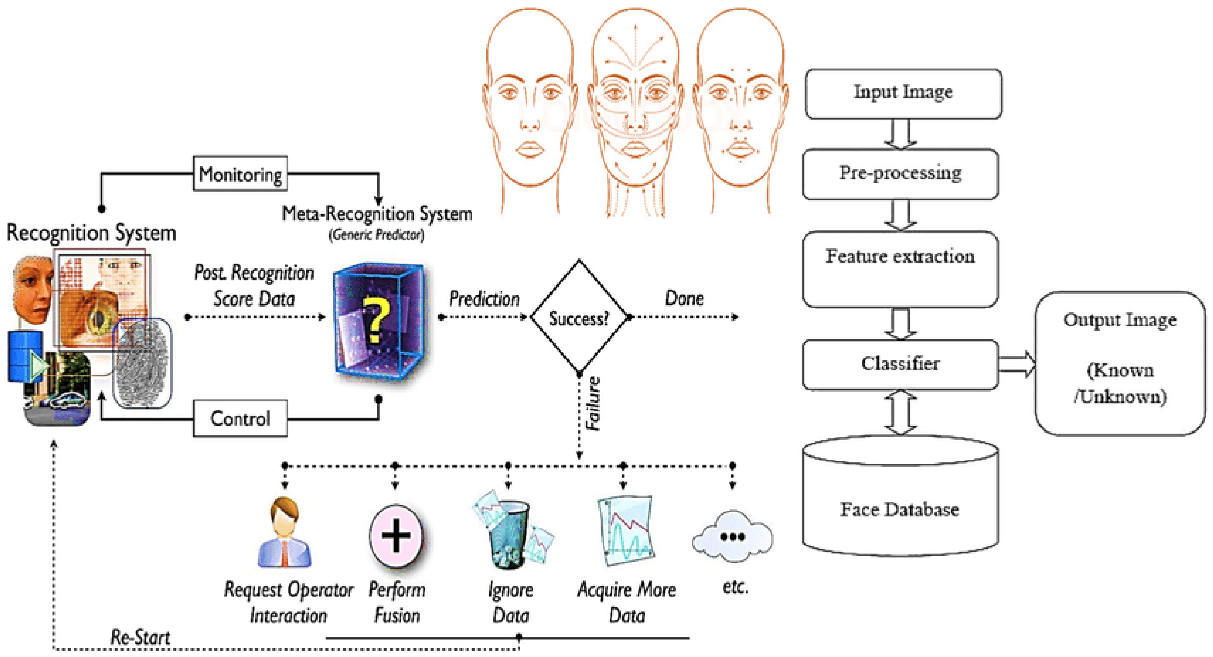


Figure 3.1 System Architecture Diagram

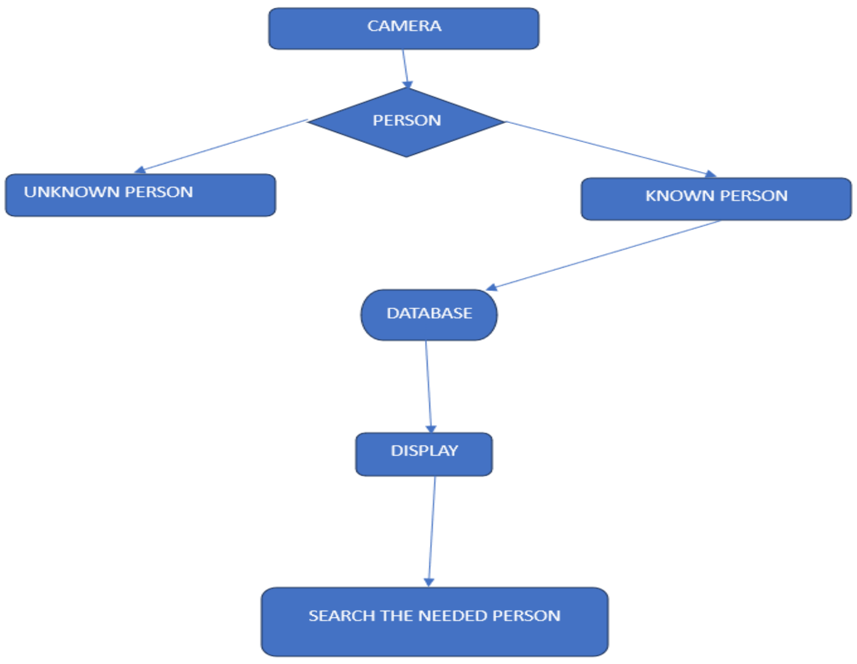


Figure 3.2 D ata Flow Diagram

CHAPTER 4
SYSTEM DESIGN

CHAPTER 4

SYSTEM DESIGN

4.1 Data For The System

In designing a robust system for face detection and recognition utilizing artificial intelligence (AI) and machine learning (ML), a comprehensive system architecture is essential. The architecture encompasses several interconnected modules, each serving a distinct function within the system. At the core of the architecture lies the Input Module, responsible for capturing input from various sources such as cameras or video files, and preprocessing it to enhance subsequent processing stages. Following this, the Face Detection Module comes into play, employing sophisticated algorithms like Haar cascades or deep learning-based approaches to identify and extract regions of interest (ROIs) containing faces from the input frames. Subsequently, the Feature Extraction Module extracts discriminative features from the detected faces, transforming them into a feature vector representation. These feature vectors are then compared against known faces stored in the Database Management Module, facilitated by the Face Recognition Module. This module employs ML models or deep learning architectures to perform recognition tasks, determining the identities of detected faces. Concurrently, the Database Management Module manages the storage and retrieval of known faces and their corresponding identities, ensuring efficient querying and updating operations. The User Interface Module provides a graphical interface for user interaction and result visualization, displaying live or recorded video feeds with detected faces highlighted, along with the recognized identities associated with them. Throughout the system, an Integration Layer facilitates seamless communication and coordination among different modules, ensuring smooth data flow and process synchronization. This systematic architecture lays the foundation for a reliable and efficient face detection and recognition system, capable of real-time performance and scalable deployment in various applications.

4.2 UML Diagrams

4.2.1 Package Diagram

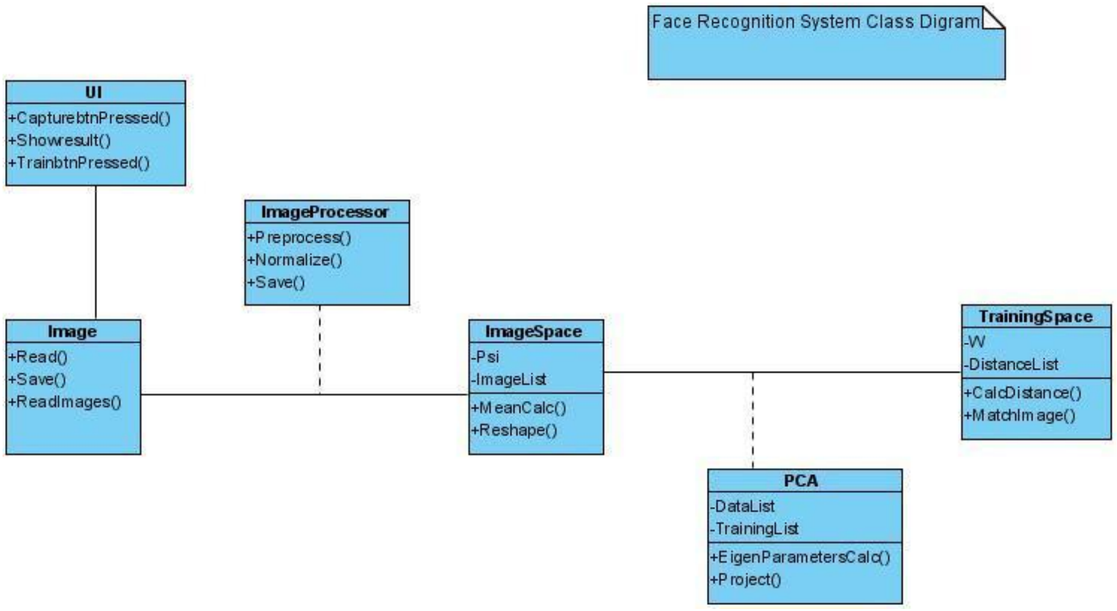


Figure 4.1 Package Diagram

4.2.2 Activity Diagram

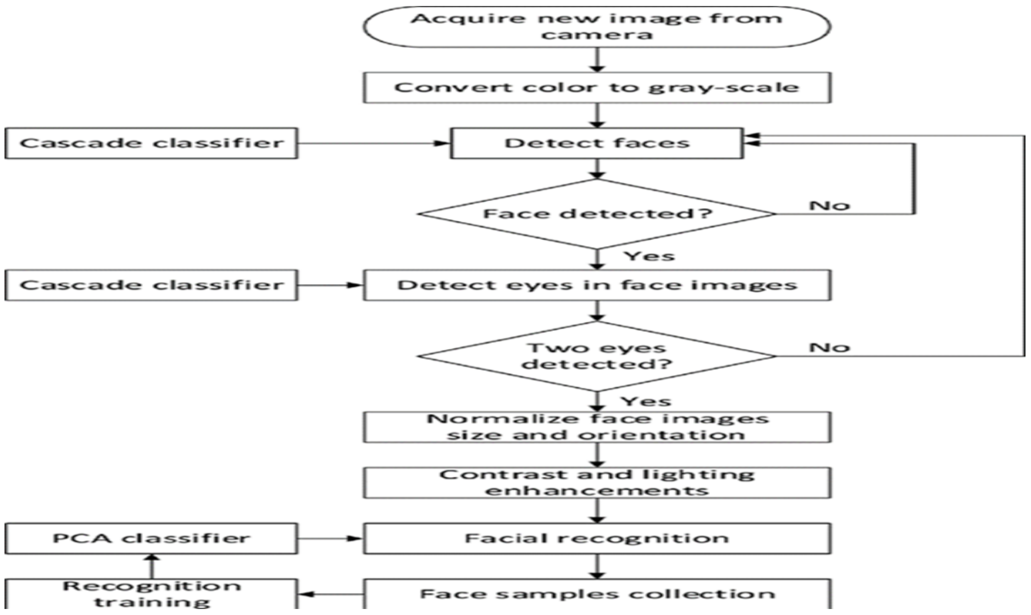


Figure 4.2 Activity Diagram

4.2.3 Entity Relationship Diagram

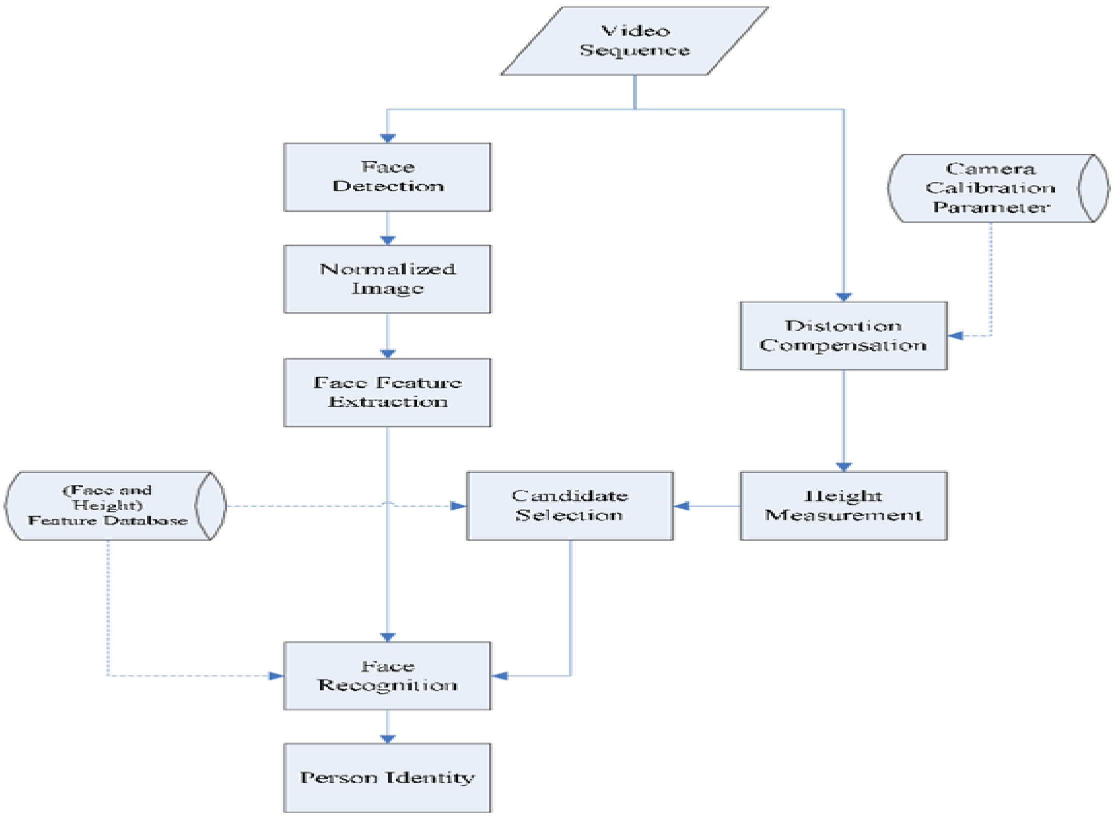


Figure 4.3 ER diagram

CHAPTER 5
SYSTEM IMPLEMENTATION

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 ALGORITHM

5.1.1 Create Dataset

```
import cv2

import os


# Ask user for the person's name

person_name = input("Enter the name for the dataset (person's name): ")


# Load the cascade

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')


# To capture video from webcam

cap = cv2.VideoCapture(0)


# Create a directory to store images

os.makedirs('person_name', exist_ok=True)

img_counter = 0


while True:

    # Read the frame
    _, img = cap.read()

    # Convert to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detect faces
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    # Draw rectangle around the facesaaabarn
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # Display
    cv2.imshow('img', img)

    # Wait for the 'c' key to capture the face
    k = cv2.waitKey(30) & 0xff
    if k == ord('c'):
        # If 'c' is pressed, capture and save the face
        face_img = gray[y:y+h, x:x+w]
        face_filename                    =                    os.path.join('person_name',
f'{person_name}_{img_counter}.png')
        cv2.imwrite(face_filename, face_img)
        print(f'Image saved: {face_filename}')
        img_counter += 1
    elif k == 27: # Press 'ESC' to exit
        break

# Release the VideoCapture object
cap.release()
cv2.destroyAllWindows()
```

5.1.2 Training Model

```
import os
import cv2
import pickle
import numpy as np
from cv2 import face
import pickle  # Import pickle
from datetime import datetime

# Path to the folder containing images
image_folder = r"A:\cam\Faces"
```

```
# Create a face recognizer
recognizer = face.LBPHFaceRecognizer_create()

# Load Haar cascade for face detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")

# Function to get the images and label data
def get_training_data(image_folder):
    face_samples = []
    labels = []

    for person_id in os.listdir(image_folder):
        person_path = os.path.join(image_folder, person_id)
        if not os.path.isdir(person_path):
            continue  # Skip if it's not a directory

        for image_name in os.listdir(person_path):
            if not (image_name.endswith(".jpg") or image_name.endswith(".png")):
                continue  # Skip non-image files

            img_path = os.path.join(person_path, image_name)
            image = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
            if image is None:
                continue  # Skip if image not loaded

            faces = face_cascade.detectMultiScale(image, scaleFactor=1.1, minNeighbors=5)
            for (x, y, w, h) in faces:
                face_samples.append(image[y:y+h, x:x+w])
                labels.append(person_id)

    return face_samples, labels

# Train the recognizer
def train_recognizer(image_folder):
    faces, string_labels = get_training_data(image_folder)

    # Convert string labels to numeric labels
    label_mapping = {label: idx for idx, label in enumerate(set(string_labels))}
    numeric_labels = [label_mapping[label] for label in string_labels]

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

    # Save the trained model and the label mapping
    recognizer.write('face_recognizer.yml')
    with open('label_mapping.pkl', 'wb') as f:
        pickle.dump(label_mapping, f)

# Load the trained model and label mapping
def load_recognizer():
    recognizer.read('face_recognizer.yml')
    with open('label_mapping.pkl', 'rb') as f:
        label_mapping = pickle.load(f)
    return recognizer, label_mapping

from datetime import datetime

# Recognition loop
def recognize_faces(image_folder):
    recognizer, label_mapping = load_recognizer()
    reversed_mapping = {v: k for k, v in label_mapping.items()}
    log_file_path = os.path.join(image_folder, 'detection_log.txt')

    cap = cv2.VideoCapture(0)

    # Dictionary to store entry times of each recognized person
    entry_times = {}
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray)

        for (x, y, w, h) in faces:
            id, confidence = recognizer.predict(gray[y:y+h, x:x+w])
            label = reversed_mapping.get(id, "Unknown")
            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            if confidence > 50:  # Adjust this threshold based on testing
                # If the person is recognized, note the entry time if it's a new person
                if id not in entry_times:
                    entry_times[id] = datetime.now()   # Store the entry time as a datetime
object
                    print(f"{label} entered at {timestamp}")
                    with open(log_file_path, 'a') as log_file:
                        log_entry = f"{timestamp} - {label} entered\n"
                        log_file.write(log_entry)
            else:
                # If the person is not recognized, note the exit time if it's a known person
                if id in entry_times:
                    entry_time = entry_times.pop(id)
                    exit_time = datetime.now()  # Store the exit time as a datetime object
                    time_difference = exit_time - entry_time
                    print(f"{label} exited at {timestamp}. Total time: {time_difference}")
                    with open(log_file_path, 'a') as log_file:
                        log_entry = f"{timestamp} - {label} exited. Total time: {time_difference}\n"
                        log_file.write(log_entry)

            with open(log_file_path, 'a') as log_file:
                log_entry = f"{timestamp} - ID: {id}, Person Name: {label}\n"
                log_file.write(log_entry)

            if confidence > 50:  # Adjust this threshold based on testing
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
                cv2.putText(frame, label, (x+5, y-5), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
            else:
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
                cv2.putText(frame, "Not Matching", (x+5, y-5), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
        cv2.imshow('frame', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):

            break


    cap.release()

    cv2.destroyAllWindows()

# Train the recognizer

train_recognizer(image_folder)

# Run the recognition looprecognize_faces(image_folder)
```

CHAPTER 6
SYSTEM TESTING

CHAPTER 6

SYSTEM TESTING

6.1 Model Test

TEST TYPE	TEST CASE DESCRIPTION	TEST SCENARIO	TEST RESULT
Unit test	Checking database creation with images	Captured image with name	passed
Integration test	The person appearing in the frame with their names	Visibility of name	passed
System test	The person entry and exit time is noted in the system	Date and time is noted	passed

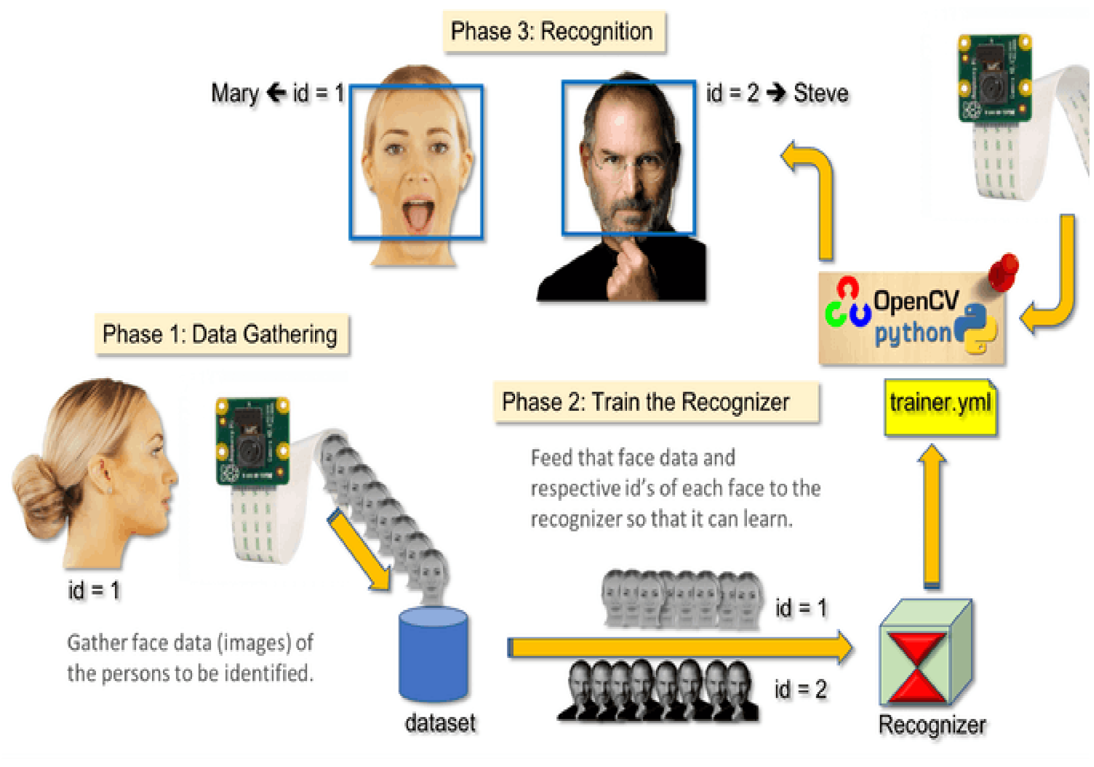

Phase 3: Recognition
Mary ← id = 1
id = 2 → Steve
Phase 1: Data Gathering
OpenCV
python
Phase 2: Train the Recognizer
trainer.yml
Feed that face data and respective id's of each face to the recognizer so that it can learn.
id = 1
Gather face data (images) of the persons to be identified.
id = 1
id = 2
dataset
Recognizer

Figure 6.1 Architecture Diagram

CHAPTER 7
CONCLUSION

CHAPTER 7

CONCLUSION

7.1 Result

Face Detection Performance Provide metrics such as precision, recall, and F1 score to evaluate the performance of the face detection module. Present results on how accurately the system detects faces in various scenarios (e.g., different lighting conditions, angles, and distances). Include visualizations such as bounding boxes around detected faces to illustrate the effectiveness of the detection algorithm.

Face Recognition Accuracy Report the accuracy of the face recognition module in identifying known faces. Use metrics like accuracy, precision, recall, and F1 score to quantify the performance. Include results on recognition accuracy for different datasets and conditions (e.g., variations in facial expressions, occlusions). Provide examples of correct and incorrect recognition cases to showcase the system's capabilities and limitations.

Real-time Processing Evaluate the system's performance in real-time processing scenarios. Report the frame rate achieved by the system while processing live video feeds. Discuss any latency issues encountered and how they were addressed or mitigated. Database Management Describe the efficiency and scalability of the database management module. Report on the time taken for database queries and updates. Discuss any challenges encountered in managing the database, such as data storage requirements and retrieval speed. User Interface and Experience Evaluate the usability and user experience of the system's interface. Gather feedback from users regarding the intuitiveness and effectiveness of the interface.

Discuss any improvements made based on user feedback to enhance the interface. Case Studies or Use Cases Present real-world examples or case studies where the system was deployed or tested. Describe the context of each case study, including the application scenario and specific requirements. Discuss the results obtained and any insights gained from deploying the system in practical settings. .Comparative Analysis Compare the performance of your system with existing face detection and recognition solutions.

Highlight the advantages and limitations of your system compared to others in terms of accuracy, speed, scalability, and user experience. Future Directions Discuss potential areas for improvement and future research directions. Propose enhancements to the system architecture or algorithms to further improve performance. Identify new application domains or use cases where the system could be deployed.

7.2 Conclusion

Summarize the key findings and results obtained from the project. Reflect on the achievements and limitations of the system. Discuss the implications of the results and their significance in the context of face detection and recognition technology.

APPENDICES

A. Dataset Description

Provide detailed information about the datasets used for training and testing your face detection and recognition system. Include details such as the number of images/videos, resolution, diversity of subjects, and any preprocessing steps applied.

B. Source Code

Training model.py

```
import os

import cv2

import pickle

import numpy as np

from cv2 import face

import pickle  # Import pickle

from datetime import datetime
```

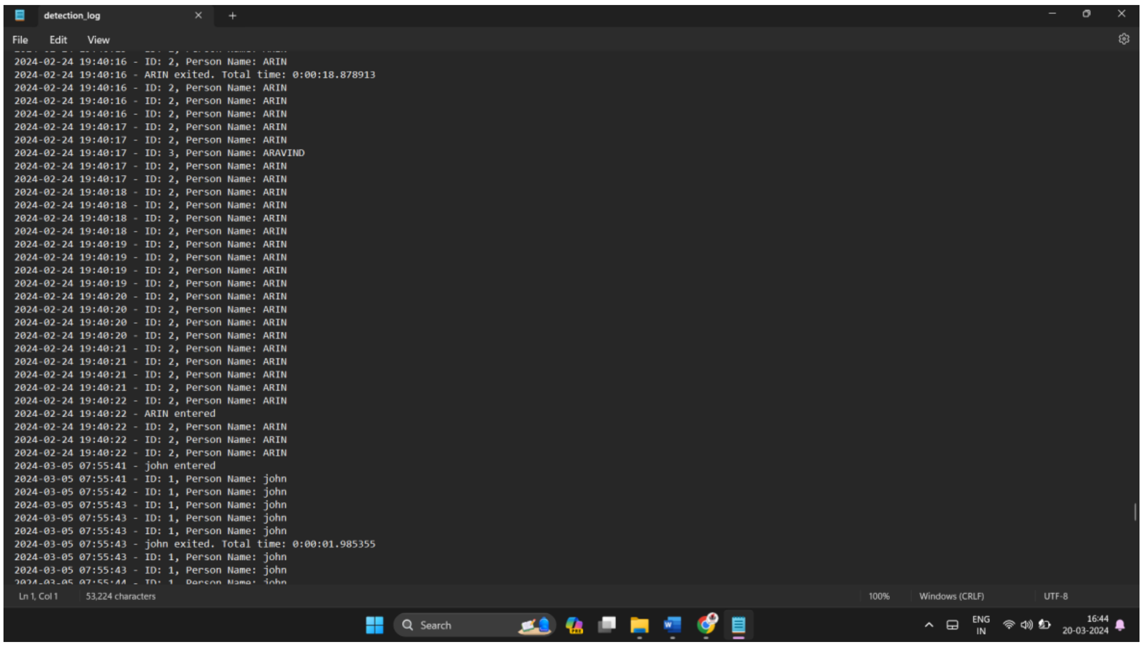
```
# Path to the folder containing images
image_folder = r"A:\cam\Faces"

# Create a face recognizer
recognizer = face.LBPHFaceRecognizer_create()

# Load Haar cascade for face detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

# Function to get the images and label data
def get_training_data(image_folder):
    face_samples = []
    labels = []

    for person_id in os.listdir(image_folder):
        person_path = os.path.join(image_folder, person_id)
        if not os.path.isdir(person_path):
            continue  # Skip if it's not a directory

        for image_name in os.listdir(person_path):
            if not (image_name.endswith(".jpg") or image_name.endswith(".png")):
```

```
            continue  # Skip non-image files

        img_path = os.path.join(person_path, image_name)
        image = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        if image is None:
            continue  # Skip if image not loaded

        faces = face_cascade.detectMultiScale(image, scaleFactor=1.1,
minNeighbors=5)
        for (x, y, w, h) in faces:
            face_samples.append(image[y:y+h, x:x+w])
            labels.append(person_id)

    return face_samples, labels

# Train the recognizer
def train_recognizer(image_folder):
    faces, string_labels = get_training_data(image_folder)

    # Convert string labels to numeric labels
    label_mapping = {label: idx for idx, label in enumerate(set(string_labels))}
    numeric_labels = [label_mapping[label] for label in string_labels]
    recognizer.train(faces, np.array(numeric_labels))


    # Save the trained model and the label mapping
    recognizer.write('face_recognizer.yml')
    with open('label_mapping.pkl', 'wb') as f:
        pickle.dump(label_mapping, f)


# Load the trained model and label mapping
def load_recognizer():
    recognizer.read('face_recognizer.yml')
    with open('label_mapping.pkl', 'rb') as f:
        label_mapping = pickle.load(f)
    return recognizer, label_mapping


from datetime import datetime


# Recognition loop
def recognize_faces(image_folder):
    recognizer, label_mapping = load_recognizer()
    reversed_mapping = {v: k for k, v in label_mapping.items()}
    log_file_path = os.path.join(image_folder, 'detection_log.txt')


    cap = cv2.VideoCapture(0)
```

```
# Dictionary to store entry times of each recognized person
entry_times = {}

while True:
    ret, frame = cap.read()
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray)

    for (x, y, w, h) in faces:
        id, confidence = recognizer.predict(gray[y:y+h, x:x+w])
        label = reversed_mapping.get(id, "Unknown")
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        if confidence > 50:  # Adjust this threshold based on testing
            # If the person is recognized, note the entry time if it's a new person
            if id not in entry_times:
                entry_times[id] = datetime.now()  # Store the entry time as a datetime
object
                print(f"{label} entered at {timestamp}")
                with open(log_file_path, 'a') as log_file:
                    log_entry = f"{timestamp} - {label} entered\n"
                    log_file.write(log_entry)
        else:
            # If the person is not recognized, note the exit time if it's a known person
            if id in entry_times:
                entry_time = entry_times.pop(id)
                exit_time = datetime.now()  # Store the exit time as a datetime object
                time_difference = exit_time - entry_time
                print(f"{label} exited at {timestamp}. Total time: {time_difference}")
                with open(log_file_path, 'a') as log_file:
                    log_entry = f"{timestamp} - {label} exited. Total time:
{time_difference}\n"
                    log_file.write(log_entry)

        with open(log_file_path, 'a') as log_file:
            log_entry = f"{timestamp} - ID: {id}, Person Name: {label}\n"
            log_file.write(log_entry)

        if confidence > 50:  # Adjust this threshold based on testing
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
            cv2.putText(frame, label, (x+5, y-5), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 255, 255), 2)
```

```
            else:
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
                cv2.putText(frame, "Not Matching", (x+5, y-5), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
        cv2.imshow('frame', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
# Train the recognizer
train_recognizer(image_folder)
# Run the recognition loop
recognize_faces(image_folder)
```

C. Screenshots



```
2024-02-24 19:40:16 - ID: 2, Person Name: ARIN
2024-02-24 19:40:16 - ARIN exited. Total time: 0:00:18.878913
2024-02-24 19:40:16 - ID: 2, Person Name: ARIN
2024-02-24 19:40:16 - ID: 2, Person Name: ARIN
2024-02-24 19:40:16 - ID: 2, Person Name: ARIN
2024-02-24 19:40:17 - ID: 2, Person Name: ARIN
2024-02-24 19:40:17 - ID: 2, Person Name: ARIN
2024-02-24 19:40:17 - ID: 3, Person Name: ARAVIND
2024-02-24 19:40:17 - ID: 2, Person Name: ARIN
2024-02-24 19:40:17 - ID: 2, Person Name: ARIN
2024-02-24 19:40:18 - ID: 2, Person Name: ARIN
2024-02-24 19:40:18 - ID: 2, Person Name: ARIN
2024-02-24 19:40:18 - ID: 2, Person Name: ARIN
2024-02-24 19:40:18 - ID: 2, Person Name: ARIN
2024-02-24 19:40:19 - ID: 2, Person Name: ARIN
2024-02-24 19:40:19 - ID: 2, Person Name: ARIN
2024-02-24 19:40:19 - ID: 2, Person Name: ARIN
2024-02-24 19:40:19 - ID: 2, Person Name: ARIN
2024-02-24 19:40:20 - ID: 2, Person Name: ARIN
2024-02-24 19:40:20 - ID: 2, Person Name: ARIN
2024-02-24 19:40:20 - ID: 2, Person Name: ARIN
2024-02-24 19:40:20 - ID: 2, Person Name: ARIN
2024-02-24 19:40:21 - ID: 2, Person Name: ARIN
2024-02-24 19:40:21 - ID: 2, Person Name: ARIN
2024-02-24 19:40:21 - ID: 2, Person Name: ARIN
2024-02-24 19:40:21 - ID: 2, Person Name: ARIN
2024-02-24 19:40:22 - ID: 2, Person Name: ARIN
2024-02-24 19:40:22 - ARIN entered
2024-02-24 19:40:22 - ID: 2, Person Name: ARIN
2024-02-24 19:40:22 - ID: 2, Person Name: ARIN
2024-02-24 19:40:22 - ID: 2, Person Name: ARIN
2024-03-05 07:55:41 - john entered
2024-03-05 07:55:41 - ID: 1, Person Name: john
2024-03-05 07:55:42 - ID: 1, Person Name: john
2024-03-05 07:55:43 - ID: 1, Person Name: john
2024-03-05 07:55:43 - ID: 1, Person Name: john
2024-03-05 07:55:43 - ID: 1, Person Name: john
2024-03-05 07:55:43 - john exited. Total time: 0:00:01.985355
2024-03-05 07:55:43 - ID: 1, Person Name: john
2024-03-05 07:55:43 - ID: 1, Person Name: john
2024-03-05 07:55:44 - ID: 1, Person Name: john
```

Figure A.1 Detection Log

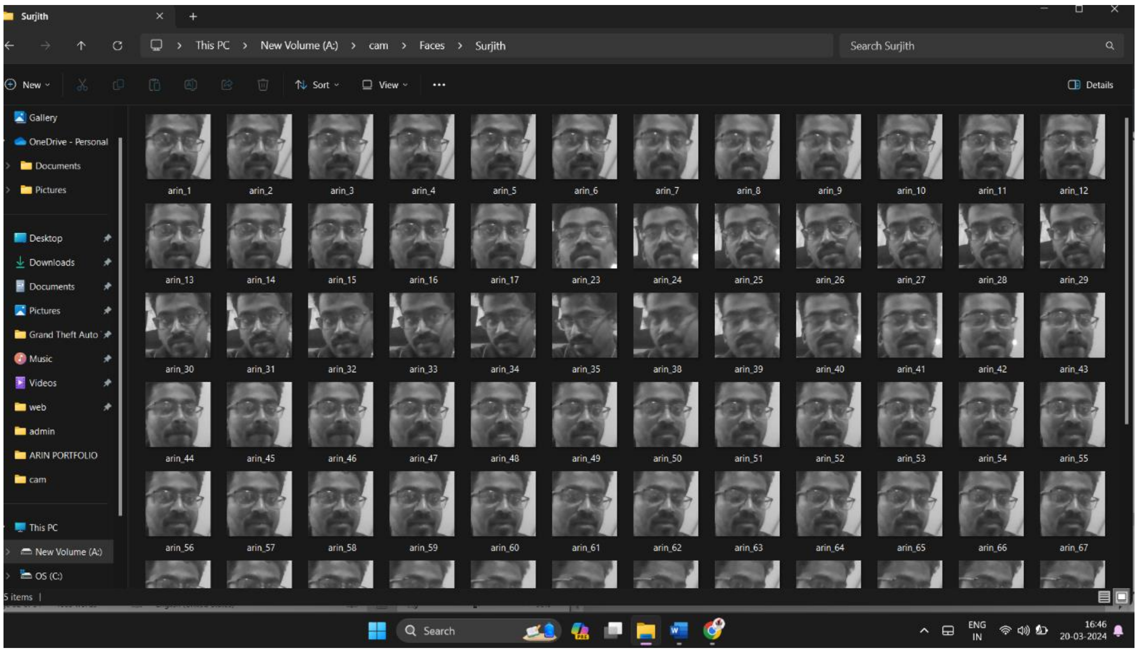
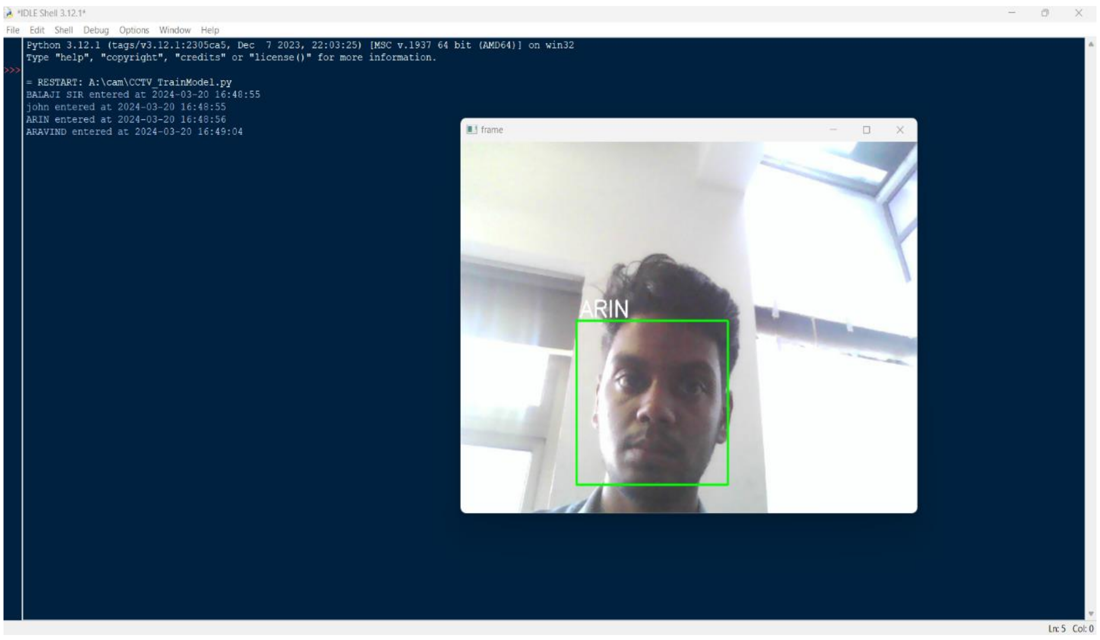


Figure A.2 Captured data



```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: A:\cam\CCTV_TrainModel.py
BALAJI SIR entered at 2024-03-20 16:48:55
john entered at 2024-03-20 16:48:55
ARIN entered at 2024-03-20 16:48:56
ARAVIND entered at 2024-03-20 16:49:04
```

Figure A.3 Person Identification Report

REFERENCES

H. References

1)Kim, J., Kim, J., Kim, H., & Oh, S. (2018). OfficeMate: Smart Attendance Management System Based on Face,

https://www.researchgate.net/publication/341876647_Face_Recognition_based_Attendance_Management_System

2) OfficeMate Smart Attendance Management System Based on Face Recognition. 2018 IEEE International Conference on Consumer Electronics (ICCE). https://www.researchgate.net/publication/334820129_Smart_Attendance_Management_System_Using_Face_Recognition

3) Research on Multimodal Fusion Technology Based on Deep Learning in Intelligent Personnel Recognition. IEEE Access, 7, 91947–91959. https://www.researchgate.net/publication/221345149_Multimodal_Deep_Learning

4) 2018 IEEE International Conference on Consumer Electronics (ICCE). IEEE. Kosinski, M., Wang, Y., Lakkaraju, H., & Leskovec, J. (2018). https://www.aconf.org/conf_108746.2018_IEEE_International_Conference_on_Consumer_Electronics.html

5) Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple Online and Realtime Tracking with a Deep Association Metric. https://arxiv.org/abs/1602.00763

6) . Jiang, B., Zhang, K., & Cai, Z. (2019). Face Recognition Technology Application in Campus Safety Management https://www.researchgate.net/publication/330546911_Application_of_Face_Recognition_Technological_Access_Control_System_in_College_Apartment_Management

7) 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) https://www.proceedings.com/48905.html