**CHAPTER 1**

**INTRODUCTION**

In colleges, universities, organizations, schools, and offices, taking attendance is one of the most important tasks that must be done on a daily basis. The majority of the time, it is done manually, such as by calling by name or by roll number. The main goal of this project is to create a Face Recognition-based attendance system that will turn this manual process into an automated one. This project meets the requirements for bringing modernization to the way attendance is handled, as well as the criteria for time management. This device is installed in the classroom, where and student's information, such as name, roll number. class, sec, and photographs, is trained. The images are extracted using Open CV.

Before the start of the corresponding class, the student can approach the machine, which will begin taking pictures and comparing them to the qualified dataset. Logitech C270 web camera and NVIDIA Jetson Nano Developer kit were used in this project as the camera and processing board. The image is processed as follows: first, faces are identified using a Haarcascade classifier, then faces are recognized using the LBPH (Local Binary Pattern Histogram) Algorithm, histogram data is checked against an established dataset, and the device automatically labels attendance. An Excel sheet is developed, and it is updated every hour with the information from the respective class instructor.

In an era characterized by rapid technological advancements, the traditional methods of attendance tracking are being redefined by innovative solutions. One such groundbreaking technology is the Face Recognition Attendance System. This system leverages the power of facial recognition algorithms to accurately identify and record individuals, offering a modern, efficient, and secure alternative to conventional attendance tracking methods.

Gone are the days of manual roll calls or reliance on physical cards; the Face Recognition Attendance System brings forth a seamless and automated approach to attendance management. By utilizing facial biometrics, this system ensures not only the precision of data but also eliminates the potential for errors associated with traditional methods. This introduction explores the paradigm shift that face recognition attendance systems bring to various sectors, from educational institutions to corporate offices, ushering in an era of efficiency, accuracy, and enhanced security in attendance management.

**CHAPTER 2**

**SYSTEM STUDY**

**2.1 EXISTING SYSTEM**

You show Several existing systems offer face recognition for attendance tracking across various industries. Here are a few examples:

Keylo: This system utilizes facial recognition for attendance management. It captures employee faces through a designated camera and integrates with existing HR software for seamless attendance tracking.

Kairos: Known for its facial recognition capabilities, Kairos offers solutions for attendance tracking that use facial biometrics to identify individuals and record their attendance.

TimeForge: TimeForge integrates facial recognition technology into its attendance tracking system, allowing employees to clock in and out by simply facing a camera, streamlining the attendance

These systems often differ in terms of their specific features, integration capabilities, and user interfaces, catering to the needs of different organizations and industries. They typically involve hardware like cameras and software that perform facial recognition and attendance tracking functions.

**2.1.1 DRAWBACKS**

**False detection about traffic:** Use advanced algorithms like YOLO or SSD for object detection and tracking. Utilize multiple sensors or data sources for cross-validation and to reduce false positives.

**Less efficiency:** Optimize algorithms, reduce video stream resolutions, and use efficient data structures. Consider distributed computing or parallel processing for handling large volumes of traffic data more efficiently.

**Continuous Movement of Traffic:** Implement dynamic event detection algorithms. Adjust sensitivity based on traffic flow rates or use historical data for predicting traffic events.

**2.2 PROPOSED SYSTEM**

Our proposed Face Recognition Attendance System leverages advanced facial detection and recognition algorithms to automate the attendance tracking process with precision and efficiency. By integrating cutting-edge Python-based technologies, the system offers a seamless user experience while ensuring accurate identification of individuals in real-time.

Through a user-friendly interface, users can easily enroll their facial biometric data, allowing the system to recognize and record attendance seamlessly. Administrators benefit from comprehensive monitoring capabilities, receiving live updates on attendance statuses and accessing detailed reports for insights into attendance trends. With robust security measures in place, including data encryption and access controls, the system prioritizes the protection of sensitive information. Scalable and customizable, the system can adapt to diverse organizational needs, making it an ideal solution for enhancing attendance management across various environments.

* + 1. **FEATURES**

**Face Detection:** Utilize algorithms to detect and locate faces within images or video streams accurately.

**Face Recognition:** Employ facial recognition technology to identify individuals based on their unique facial features.

**Attendance Tracking:** Monitor and record attendance automatically, linking recognized faces to attendance records.

**Real-time Monitoring:** Provide instant updates on attendance status as faces are recognized.

**User Interface:** Offer a user-friendly interface for managing attendance data and system settings. **Top of Form**

**2.3 SYSTEM SPECIFICATION**

**2.3.1 HARDWARE CONFIGURATION**

Processor : Dual Core Processor

RAM : 4GB

Hard Disk : 500GB

**2.3.2 SOFTWARE SPECIFICATION**

Operating System : Windows 10

Programming Language : Python

**2.3.3 SOFTWARE FEATURES**

**Python**

Python is an object-oriented programming language created by Guido Rossum in 1989, It is ideally designed for rapid prototyping of complex applications, it has interfaces system calls and libraries and is extensible to C or C++, Many large companies use the Python language, including NASA, Google, YouTube, BitTorrent, etc. programming

Python is an easy to learn, powerful programming language, It has efficient high-level data structures and a simple but effective approach to object-oriented programming, Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python and may be freely distributed.

The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation. The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C), Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well. For a description of standard objects and modules, see library-index. Reference-index gives a more formal definition of the language.

To write extensions in C or C++, read extending-index and c-api-index. There are also several books covering Python in depth. This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in library-index.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

**History of python**

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wickenden & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989.Python 2.0 was released on 16 October 2000, with many major new features. Python 3.0, released on 3 December 2008, with many of its major features backported to Python 2.6.x and 2.7.x. Releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.

**Methods**

Methods on objects are functions attached to the object's class; the syntax instance.method(argument) is, for normal methods and functions, syntactic sugar for Class.method (instance, argument). Python methods have an explicit self parameter to access instance data, in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby).

**Typing**

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Python Programming Characteristics**

1. It provides rich data types and easier to read syntax than any other programming languages
2. It is a platform-independent scripted language with full access to operating system API's
3. Compared to other programming languages, it allows more run-time flexibility
4. It includes the basic text manipulation facilities of Perl and Awk
5. A module in Python may have one or more classes and free functions
6. Libraries in Pythons are cross-platform compatible with Linux, Macintosh, and Windows
7. For building large applications, Python can be compiled to byte-code
8. Python supports functional and structured programming as well as OOP
9. It supports interactive mode that allows interacting Testing and debugging of snippets of code
10. In Python, since there is no compilation step, editing, debugging, and testing are fast.

**Application of Python Programming Language**

1. Program video games
2. Build Artificial Intelligence algorithms
3. Program various scientific programs such as statistical models

**CHAPTER 3**

**SYSTEM DESIGN AND DEVELOPMENT**

**3.1 FILE DESIGN**

The file design of the face recognition attendance system encompasses two fundamental components: configuration files and data storage formats. Configuration files serve as the backbone for customizing system behavior and parameters. They encapsulate settings such as recognition thresholds, camera configurations, and database connection details. These files adhere to a structured format, typically in JSON or YAML, allowing users to easily modify settings according to their requirements. On the other hand, data storage design governs the organization and management of attendance records. Attendance data is stored efficiently, often in relational databases or flat files, structured to facilitate seamless retrieval and analysis. Each record may include attributes such as timestamps, user IDs, and recognition confidence scores. Security considerations are paramount, with encryption mechanisms safeguarding sensitive data and access controls regulating file interactions. Moreover, adherence to best practices ensures robustness, scalability, and interoperability across different file systems and environments. By meticulously designing configuration files and data storage formats, the face recognition attendance system achieves flexibility, reliability, and security, thereby facilitating efficient management of attendance records while upholding data integrity and privacy.

**3.2 INPUT DESIGN**

Input design for the face recognition attendance system revolves around intuitive interfaces that enable users to interact seamlessly with the system. The primary input mechanism involves capturing facial images for recognition and attendance recording. This process typically occurs through the integration of cameras or image sensors, which serve as the input devices for acquiring facial data. The user interface should provide clear instructions for users to position themselves appropriately within the camera frame and trigger the image capture process. Feedback mechanisms, such as visual cues or auditory prompts, can enhance user experience by confirming successful image capture or indicating errors. Additionally, the system may incorporate supplementary input methods, such as text-based forms or touchscreens, for tasks like user registration, configuration adjustments, or manual attendance entries. These interfaces should prioritize simplicity and usability, guiding users through input tasks with minimal cognitive load.

* 1. **OUTPUT DESIGN**

Output design in the face recognition attendance system focuses on delivering clear, informative, and actionable feedback to users. The primary output mechanism entails presenting recognition results and attendance records in a comprehensible format. Upon successful recognition, the system should promptly display relevant information, such as the recognized individual's name, associated metadata, and attendance status. Visual indicators, such as color-coded notifications or status icons, can intuitively convey the outcome of recognition events, facilitating quick interpretation by users. Additionally, the system may generate comprehensive attendance reports summarizing attendance data over specified time periods, which users can access for analysis and decision-making. These reports should feature organized layouts, customizable views, and export functionalities to accommodate diverse user preferences and analytical requirements. Real-time feedback mechanisms, such as auditory cues or on-screen messages, can further enhance user engagement and situational awareness during recognition operations. Moreover, the system should prioritize accessibility considerations, employing adaptive design principles to cater to users with varying sensory or cognitive abilities. By embracing user-centered design principles, the face recognition attendance system delivers outputs that empower users with actionable insights, streamline administrative workflows, and foster informed decision-making.

**3.4 DATABASE DESIGN**

Database design for the face recognition attendance system integrates seamlessly with Python and machine learning methodologies, leveraging a relational database management system (RDBMS) to store and manage attendance-related data. The system architecture encompasses Python libraries and frameworks for face detection, recognition, and attendance tracking, integrating machine learning algorithms to enhance recognition accuracy and performance. The database schema consists of tables representing entities such as users, attendance sessions, and recognition results, with relationships defined using primary and foreign keys to ensure data integrity and facilitate efficient data retrieval operations. The integration of machine learning models for facial feature extraction and recognition augments the system's capabilities, enabling real-time attendance tracking and analytics based on captured facial images By combining Python's machine learning capabilities with a robust relational database backend, the face recognition attendance system delivers a scalable, accurate, and user-friendly solution for attendance management in diverse environments.

* 1. **SYSTEM DEVELOPMENT**

**3.5.1 DESCRIPTION OF MODULES**

1. Face Detection Module
2. Face Recognition Module
3. Face Training Module
4. Matching Algorithm Module
5. Report Generation Module

**Face Detection Module**

The Face Detection Module stands as the cornerstone of our face recognition attendance system, enabling swift and accurate identification of human faces within images or live video feeds. Powered by state-of-the-art algorithms and machine learning techniques, this module seamlessly navigates through varying lighting conditions, angles, and facial orientations to pinpoint facial regions with remarkable precision. Leveraging methodologies such as Haar cascades, Histogram of Oriented Gradients (HOG), or deep learning-based approaches, our system efficiently detects faces and delineates facial boundaries, ensuring robust coverage across diverse scenarios. Real-time processing capabilities enable rapid analysis of video streams or image data, facilitating seamless integration with downstream recognition and attendance tracking components. With its robust error handling mechanisms and adaptive processing capabilities, the Face Detection Module serves as the linchpin of our system, laying the foundation for precise and reliable attendance management in dynamic environments.

**Face Training Module**

The Face Training Module refines our face recognition system through supervised learning. It optimizes algorithms using labeled datasets, enhancing the system's ability to detect subtle facial features accurately. The Face Training Module is a component of the system responsible for training the face recognition model. It involves capturing and processing images of individuals' faces to create a database of known faces. Employing machine learning models like Support Vector Machines or Convolutional Neural Networks, it ensures robust recognition performance. Through iterative training and parameter adjustments, the module equips our system to adapt to changing environments, ensuring consistent accuracy in attendance tracking.

**Face Recognition Module**

The Face Recognition Module serves as the cornerstone of our attendance system, seamlessly integrating cutting-edge algorithms and techniques to identify individuals from images or live video streams. Leveraging advanced face detection methodologies, the module swiftly locates facial regions within input data, navigating through diverse lighting conditions and facial orientations with precision. Following detection, intricate feature extraction mechanisms meticulously distill distinctive facial attributes, laying the groundwork for accurate identification. Through rigorous model training on labeled datasets, our system learns to discern subtle nuances and variations, equipping it with the capability to distinguish between individuals with exceptional accuracy.

Incorporating sophisticated recognition algorithms, the module conducts intricate comparisons between extracted features and stored templates, culminating in the assignment of confidence scores that reflect the strength of recognition. Real-time processing capabilities ensure swift and responsive handling of recognition tasks, facilitating seamless integration with user interfaces and database systems. Security measures, including data encryption and privacy safeguards, underscore our commitment to protecting sensitive biometric information. With its robust error handling mechanisms and intuitive integration capabilities, the Face Recognition Module stands as a cornerstone in our quest for efficient and reliable attendance management.

**Matching Algorithm Module**

The Matching Algorithm Module serves as the backbone of our face recognition system, facilitating the precise comparison of facial features for identification purposes. Leveraging advanced algorithms such as Euclidean distance or cosine similarity, this module evaluates the likeness between facial templates and detected features with remarkable accuracy. By calculating similarity scores and thresholds, it determines the confidence level of matches, enabling reliable identification of individuals in real-time scenarios. Through meticulous tuning and optimization, the Matching Algorithm Module ensures robust performance across diverse lighting conditions, facial expressions, and orientations, making it a cornerstone in our pursuit of efficient and accurate attendance management.

**Report Generation Module**

The Report Generation Module automates the creation of comprehensive attendance reports, offering insightful summaries and analyses of attendance data. Utilizing data stored in the system's database, this module generates visually appealing reports tailored to specific time frames, user groups, or organizational needs. The Report Generation Module processes attendance data, generating comprehensive reports in various formats. It retrieves and aggregates attendance records from the database, calculating metrics like total attendance and percentages. Users can customize reports based on date ranges or specific individuals. The module presents reports in a user-friendly format for easy analysis and decision-making. Additionally, it allows for the exporting of reports for further analysis or sharing purposes. Through customizable templates and intuitive interfaces, users can effortlessly generate reports encompassing attendance trends, individual records, and statistical insights. The module supports various output formats such as PDF, CSV, or Excel, ensuring compatibility with different platforms and user preferences. By providing actionable insights and facilitating data-driven decision-making, the Report Generation Module enhances the efficiency and transparency of attendance management processes.

**CHAPTER 4**

**TESTING AND IMPLEMENTATION**

**TESTING**

Software testing is a critical element of software quality assurance that represents the ultimate review of specifications, design and coding. The user tests the developed system and changes are made according to their needs. The testing phase involves the testing of developed system using various kinds of data. It involves user training, system testing and successful running of the developed system.

The changes are made according to their needs. The testing phase involves the testing of the developed system using various kinds of data. While testing, errors are noted and corrections are made system testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences.

**TESTING PLAN**

Testing is the process of executing a program with the intent of finding any errors. A good test of course has the high probability of finding a yet undiscovered error. A successful testing is the one that uncovers a yet undiscovered error.

A test is vital to the success of the system; system test makes a logical assumption that if all parts of the system are correct, then goal will be successfully achieved. The candidate system is subjected to a verity of tests online like responsiveness, its value, stress and security. A series of tests are performed before the system is user acceptance testing.

**TESTING METHODS**

The different types of testing are:-

1. Unit Testing
2. Integration Testing
3. Validation Testing
4. Output Testing
5. User Acceptance Testing

**UNIT TESTING**

Unit testing focuses verification efforts on the smallest unit of software design, the module. This is also known as "Module Testing" The modules are tested separately this testing is caried out during programming stage itself. In this step each module is found to be working satisfaction as regard to the expected output from the module.

**INTEGRATION TESTING**

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units. Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.

**VALIDATION TESTING**

Validation testing is the process of ensuring that the tested and developed software satisfies the client/user’s needs. The business requirement logic or scenarios have to be tested in detail. All the critical functionalities of an application must be tested here. As a tester, it is always important to know how to verify the business logic or scenarios that are given to you. One such method that helps in detailed evaluation of functionalities is the Validation Process.

**WHITEBOX TESTING**

White box Testing is done with the project which drive test cases that do the following

1. Guarantee that all the independent paths with in modules have been exercise at least once.
2. Exercise all logical decision on the true and false side.
3. Execute all loops at the boundaries and within their operation bounds.
4. Exercise internal data structures to ensure the validity

It is aimed at ensuring that the system works accurately and efficiently before live operation command.

**BLACKBOX TESTING**

Black box System methods focus on the functional requirement of the software. Using the black box testing method the following errors are identified and rectified in the package.

1. Incorrect or Missing functions
2. Interface Errors
3. Errors in data Structures or external database access

**TEST RESULT**

The above mentioned all testing were conducted successfully and no errors were encountered.

**IMPLEMENTATION**

**WORKFLOW**

The implementation workflow for our face recognition attendance system involves initial planning and requirement gathering, followed by system design, module development, integration, testing, deployment, and maintenance. We'll define project objectives, gather requirements, design system architecture and database schema, develop modules including face detection and recognition, integrate components, conduct testing for functionality and performance, deploy the system, and ensure ongoing maintenance and improvement. Comprehensive documentation and knowledge transfer will support effective implementation and operation.

**Implementation Plan**

The implementation plan for our face recognition attendance system is structured to ensure efficient development and deployment while maintaining high standards of functionality and usability. We'll begin by meticulously planning project objectives and gathering comprehensive requirements from stakeholders to establish a clear direction for the system. With this foundation, our focus will shift to system design, where we'll architect the system's architecture, database schema, and user interfaces to align with user needs and technical requirements. The development phase will involve the creation of crucial modules such as face detection and recognition, along with database integration to handle user profiles and attendance records. Integration and testing will follow closely, ensuring seamless interaction between components and rigorous validation of system functionality and performance.

Once the system has been thoroughly tested, we'll proceed with deployment, leveraging best practices to ensure a smooth transition into the production environment. Continuous maintenance and improvement will be prioritized post-deployment, with a focus on monitoring system performance, addressing user feedback, and implementing necessary updates.

In this phase, we'll define project objectives, establish timelines, and engage stakeholders to gather detailed requirements.

As we transition into module development, we'll prioritize the creation of key components such as the face detection and recognition modules, integrating them seamlessly with our database for efficient data management. Rigorous testing will then ensue, encompassing unit tests for individual modules and comprehensive integration testing to ensure smooth interoperability across the system.

Once our system passes testing benchmarks, we'll embark on the deployment phase, carefully orchestrating the rollout to ensure minimal disruption to operations. Post-deployment, our focus will shift to maintenance and improvement, where we'll establish robust monitoring mechanisms, address user feedback, and implement enhancements to optimize system performance and user experience. Throughout this journey, documentation will be meticulously crafted to serve as a knowledge repository, empowering stakeholders and end-users with the insights needed to leverage the system effectively and foster ongoing success.

**CHAPTER 5**

**CONCLUSION**

The development of our face recognition attendance system represents a significant milestone in enhancing attendance management processes. Through meticulous planning, collaborative effort, and innovative technology, we have created a robust solution that not only accurately tracks attendance but also prioritizes user convenience and data security. Despite encountering challenges along the way, our team's dedication and resilience have propelled us to success. Looking forward, we recognize the system's potential to revolutionize attendance tracking across various sectors, improving efficiency and transparency.

A face recognition attendance system offers a modern and efficient solution to traditional attendance tracking methods. It leverages advanced technology to accurately identify and record individuals, eliminating the need for manual processes such as taking roll calls or using physical cards. This not only saves time and reduces administrative burdens but also enhances the overall security and reliability of attendance data.

While face recognition attendance systems bring numerous benefits, it is crucial to address potential challenges such as privacy concerns, ethical considerations, and the need for robust security measures to protect the collected data. Striking a balance between technological advancements and ethical considerations will be key for the widespread acceptance and successful implementation of face recognition attendance systems in various educational, corporate, and organizational settings.

**BIBLIOGRAPHY**

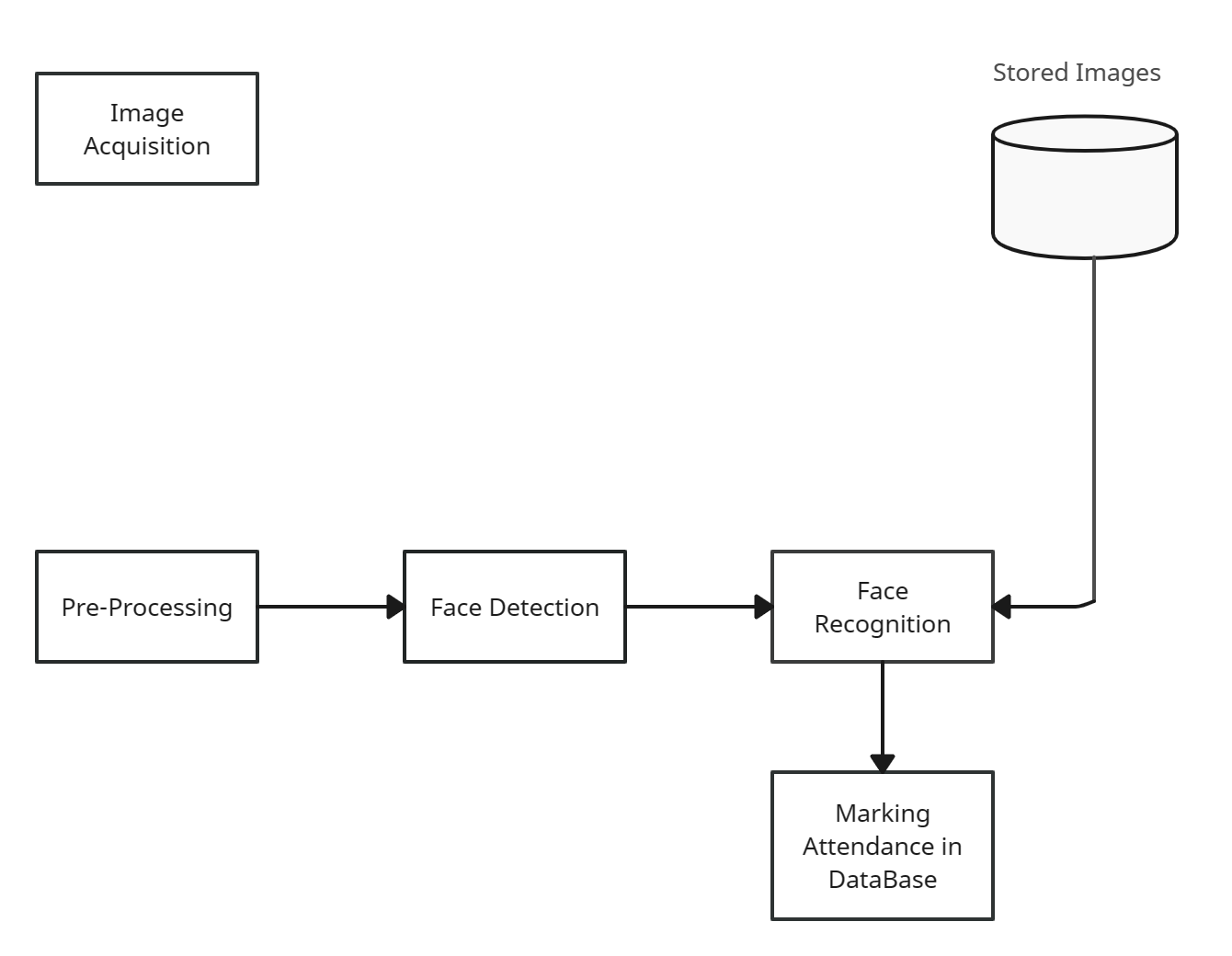
**BOOKS REFERRED**

* Roger S Pressman**, 'Software Engineering'**, 2000 Edition, Dreamtech Publications
* Paul Barry, **‘Head-First Python’**,1991 Edition, O'Reilly Media, Inc.
* Eric Matthes,**‘Python Crash Course’**, 2000 edition,‎No Starch Press.
* Kaner, C., Falk, J., & Nguyen, **‘Testing Computer Software’**, 1999 Edition Wileypublication
* Glenford J. Myers, Corey Sandler, and Tom Badgett,**"The Art Of Software Testing"** 1979 Edition, John Wiley & SonsPublications.

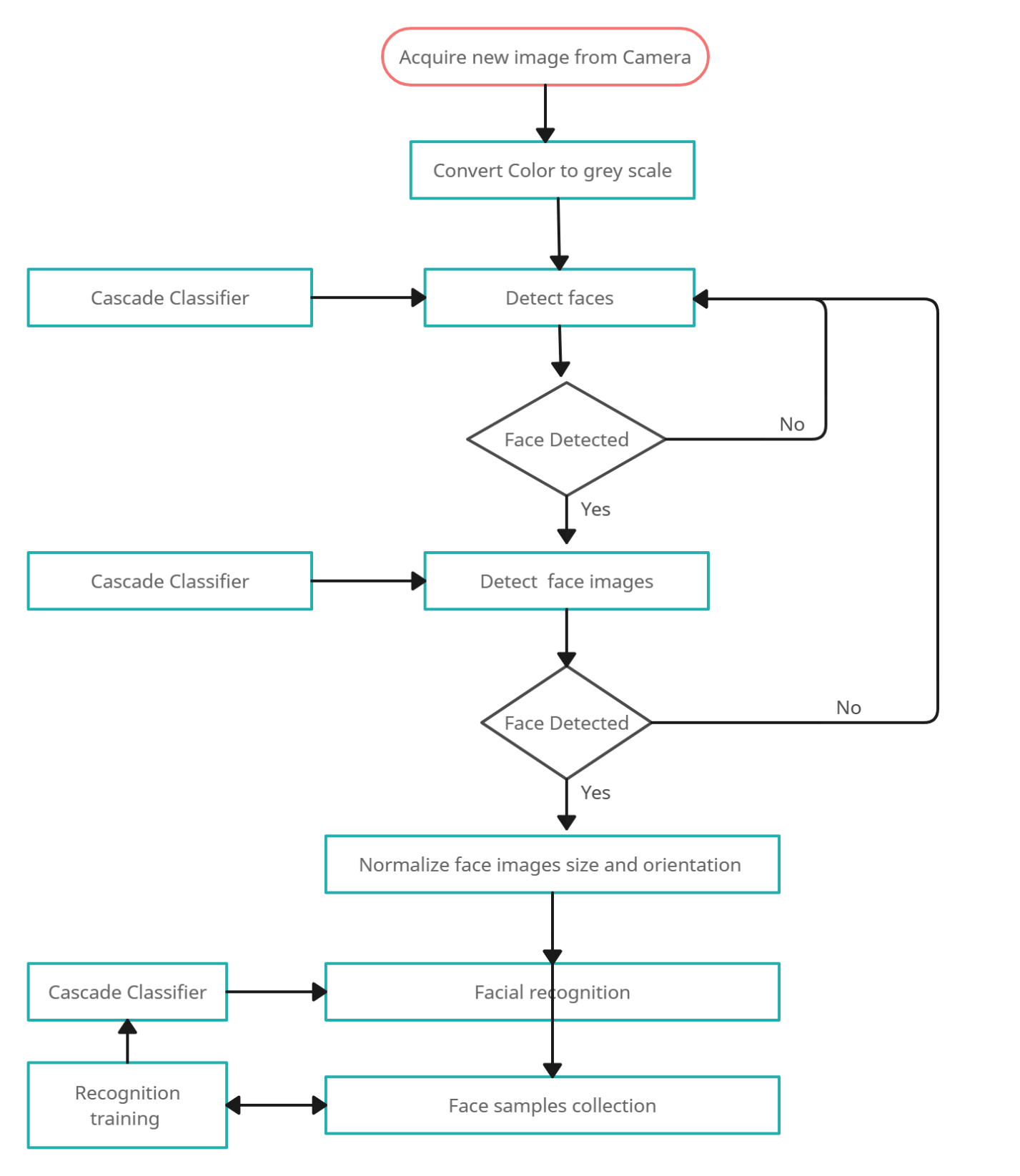
**APPENDICES**

**A. DATA FLOW DIAGRAM**





**STUDENT**



**B. SAMPLE CODE**

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox as mess

import tkinter.simpledialog as tsd

import cv2,os

import csv

import numpy as np

from PIL import Image

import pandas as pd

import datetime

import time

def assure\_path\_exists(path):

dir = os.path.dirname(path)

if not os.path.exists(dir):

os.makedirs(dir)

def tick():

time\_string = time.strftime('%H:%M:%S')

clock.config(text=time\_string)

clock.after(200,tick)

def contact():

def check\_haarcascadefile():

cascade\_file = "haarcascade\_frontalface\_alt2.xml"

exists = os.path.isfile(cascade\_file)

if exists:

pass

else:

mess.\_show(title='Some file missing', message='Please contact us for help')

window.destroy()

def save\_pass():

assure\_path\_exists("TrainingImageLabel/")

exists1 = os.path.isfile("TrainingImageLabel\psd.txt")

if exists1:

tf = open("TrainingImageLabel\psd.txt", "r")

key = tf.read()

else:

master.destroy()

new\_pas = tsd.askstring('Old Password not found', 'Please enter a new password below', show='\*')

if new\_pas == None:

mess.\_show(title='No Password Entered', message='Password not set!! Please try again')

else:

tf = open("TrainingImageLabel\psd.txt", "w")

tf.write(new\_pas)

mess.\_show(title='Password Registered', message='New password was registered successfully!!')

return

op = (old.get())

newp= (new.get())

nnewp = (nnew.get())

if (op == key):

if(newp == nnewp):

txf = open("TrainingImageLabel\psd.txt", "w")

txf.write(newp)

else:

mess.\_show(title='Error', message='Confirm new password again!!!')

return

else:

mess.\_show(title='Wrong Password', message='Please enter correct old password.')

return

mess.\_show(title='Password Changed', message='Password changed successfully!!')

master.destroy()

def clear():

txt.delete(0, 'end')

res = "1)Take Images >>> 2)Save Profile"

message1.configure(text=res)

def clear2():

txt2.delete(0, 'end')

res = "1)Take Images >>> 2)Save Profile"

message1.configure(text=res)

window = tk.Tk()

window.geometry("1280x720")

window.resizable(True,False)

window.title("Attendance System")

window.configure(background='#2d420a')

frame1 = tk.Frame(window, bg="#c79cff")

frame1.place(relx=0.11, rely=0.17, relwidth=0.39, relheight=0.80)

frame2 = tk.Frame(window, bg="#c79cff")

frame2.place(relx=0.51, rely=0.17, relwidth=0.38, relheight=0.80)

message3 = tk.Label(window, text="Face Recognition Based Attendance Monitoring System" ,fg="white",bg="#2d420a" ,width=55 ,height=1,font=('comic', 29, ' bold '))

message3.place(x=10, y=10)

frame3 = tk.Frame(window, bg="#c4c6ce")

frame3.place(relx=0.52, rely=0.09, relwidth=0.09, relheight=0.07)

frame4 = tk.Frame(window, bg="#c4c6ce")

frame4.place(relx=0.36, rely=0.09, relwidth=0.16, relheight=0.07)

datef = tk.Label(frame4, text = day+"-"+mont[month]+"-"+year+" | ", fg="#ff61e5",bg="#2d420a" ,width=55 ,height=1,font=('comic', 22, ' bold '))

datef.pack(fill='both',expand=1)

clock = tk.Label(frame3,fg="#ff61e5",bg="#2d420a" ,width=55 ,height=1,font=('comic', 22, ' bold '))

clock.pack(fill='both',expand=1)

tick()

head2 = tk.Label(frame2, text=" For New Registrations ", fg="black",bg="#00fcca" ,font=('comic', 17, ' bold ') )

head2.grid(row=0,column=0)

head1 = tk.Label(frame1, text=" For Already Registered ", fg="black",bg="#00fcca" ,font=('comic', 17, ' bold ') )

head1.place(x=0,y=0)

lbl = tk.Label(frame2, text="Enter ID",width=20 ,height=1 ,fg="black" ,bg="#c79cff" ,font=('comic', 17, ' bold ') )

lbl.place(x=80, y=55)

txt = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold '))

txt.place(x=30, y=88)

lbl2 = tk.Label(frame2, text="Enter Name",width=20 ,fg="black" ,bg="#c79cff" ,font=('comic', 17, ' bold '))

lbl2.place(x=80, y=140)

txt2 = tk.Entry(frame2,width=32 ,fg="black",font=('comic', 15, ' bold ') )

txt2.place(x=30, y=173)

message1 = tk.Label(frame2, text="1)Take Images >>> 2)Save Profile" ,bg="#c79cff" ,fg="black" ,width=39 ,height=1, activebackground = "#3ffc00" ,font=('comic', 15, ' bold '))

message1.place(x=7, y=230)

message = tk.Label(frame2, text="" ,bg="#c79cff" ,fg="black" ,width=39,height=1, activebackground = "#3ffc00" ,font=('comic', 16, ' bold '))

message.place(x=7, y=450)

lbl3 = tk.Label(frame1, text="Attendance",width=20 ,fg="black" ,bg="#c79cff" ,height=1 ,font=('comic', 17, ' bold '))

lbl3.place(x=100, y=115)

res=0

exists = os.path.isfile("StudentDetails\StudentDetails.csv")

if exists:

with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:

reader1 = csv.reader(csvFile1)

for l in reader1:

res = res + 1

res = (res // 2) - 1

csvFile1.close()

else:

res = 0

message.configure(text='Total Registrations till now : '+str(res))

tv= ttk.Treeview(frame1,height =13,columns = ('name','date','time'))

tv.column('#0',width=82)

tv.column('name',width=130)

tv.column('date',width=133)

tv.column('time',width=133)

tv.grid(row=2,column=0,padx=(0,0),pady=(150,0),columnspan=4)

tv.heading('#0',text ='ID')

tv.heading('name',text ='NAME')

tv.heading('date',text ='DATE')

tv.heading('time',text ='TIME')

clearButton = tk.Button(frame2, text="Clear", command=clear ,fg="black" ,bg="#ff7221" ,width=11 ,activebackground = "white" ,font=('comic', 11, ' bold '))

clearButton.place(x=335, y=86)

clearButton2 = tk.Button(frame2, text="Clear", command=clear2 ,fg="black" ,bg="#ff7221" ,width=11 , activebackground = "white" ,font=('comic', 11, ' bold '))

clearButton2.place(x=335, y=172)

takeImg = tk.Button(frame2, text="Take Images", command=TakeImages ,fg="white" ,bg="#6d00fc" ,width=34 ,height=1, activebackground = "white" ,font=('comic', 15, ' bold '))

takeImg.place(x=30, y=300)

trainImg = tk.Button(frame2, text="Save Profile", command=psw ,fg="white" ,bg="#6d00fc" ,width=34 ,height=1, activebackground = "white" ,font=('comic', 15, ' bold '))

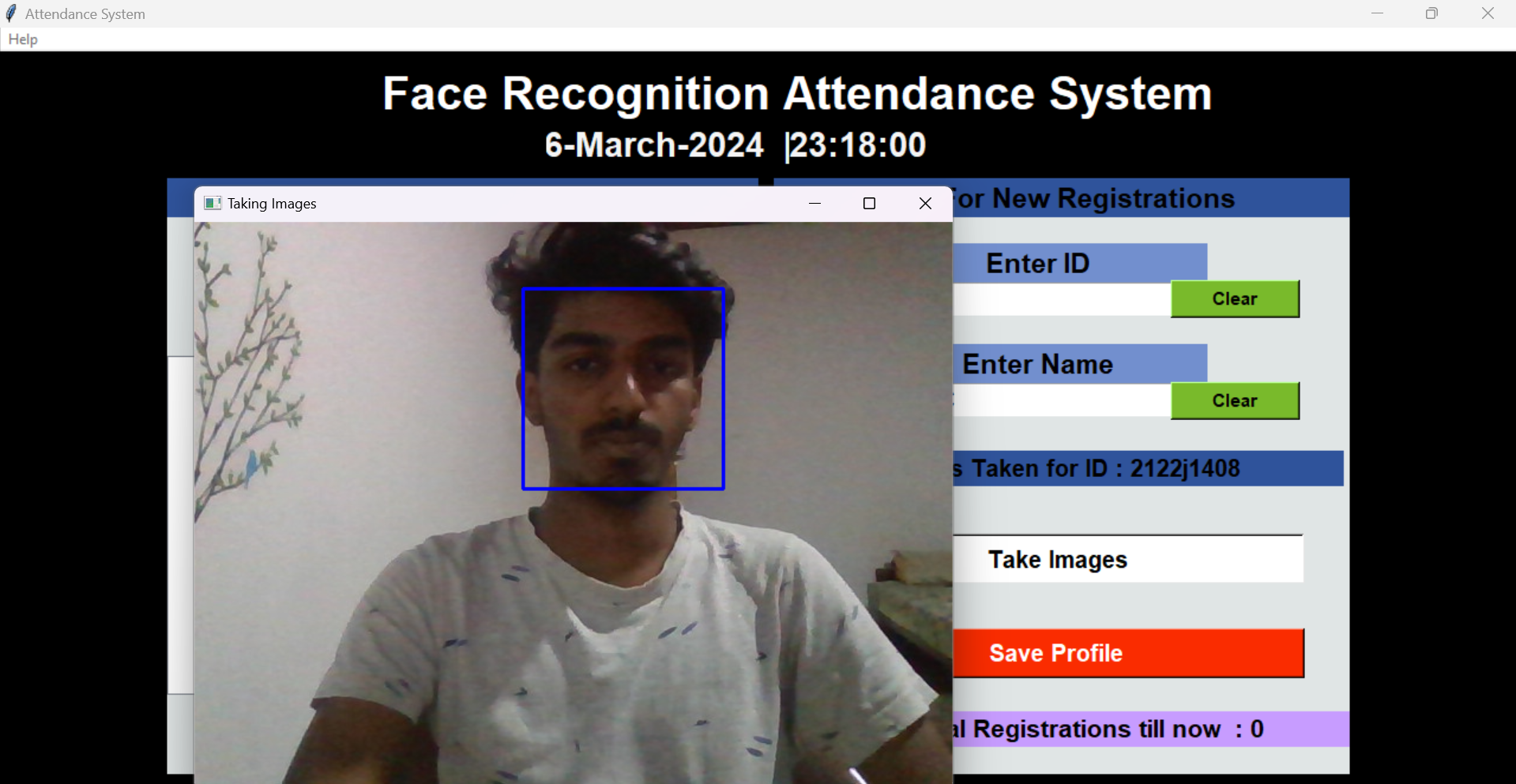
trainImg.place(x=30, y=380)

quitWindow.place(x=30, y=450)

window.configure(menu=menubar)

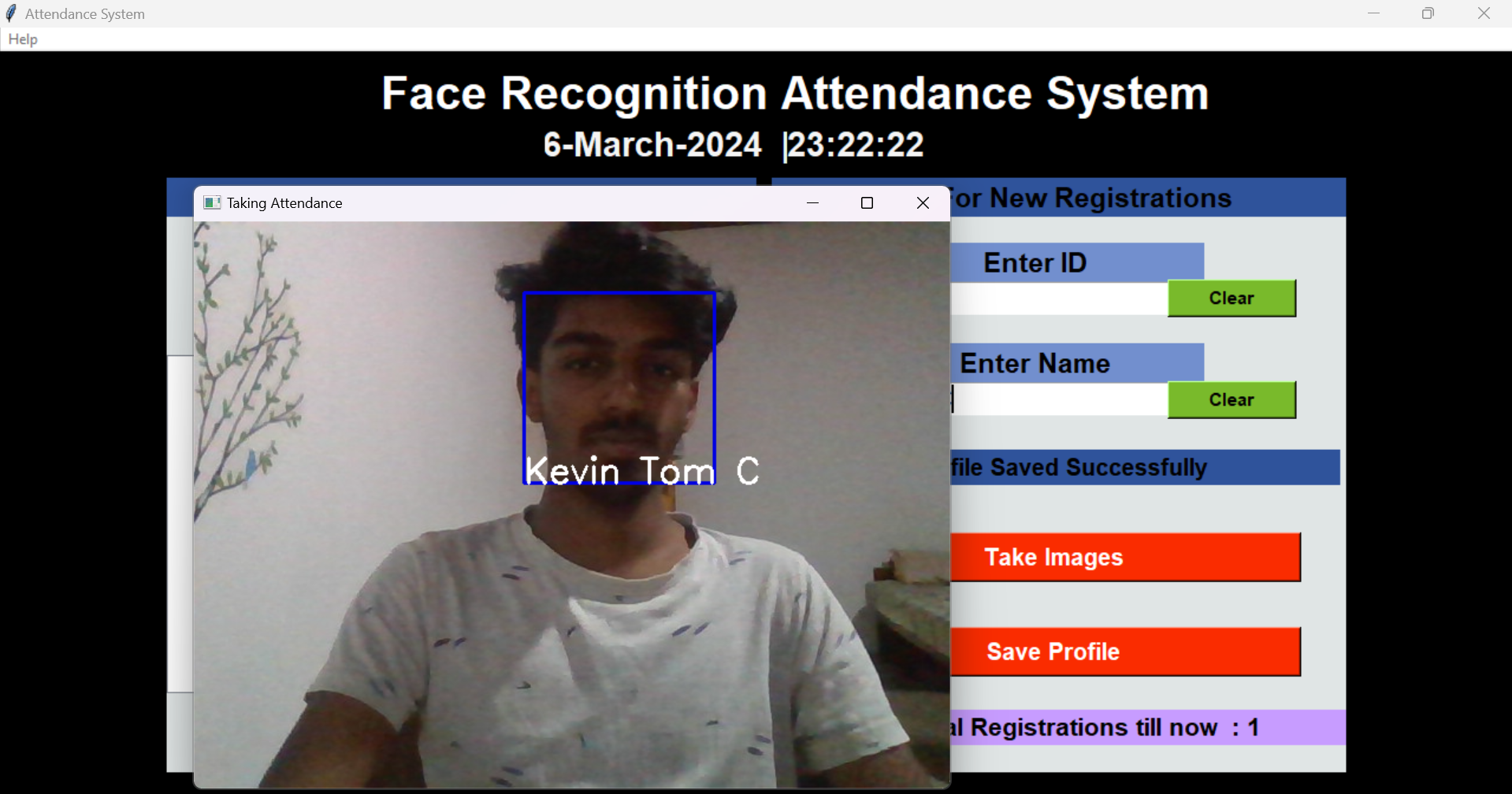
window.mainloop()

**C. SAMPLE INPUT**



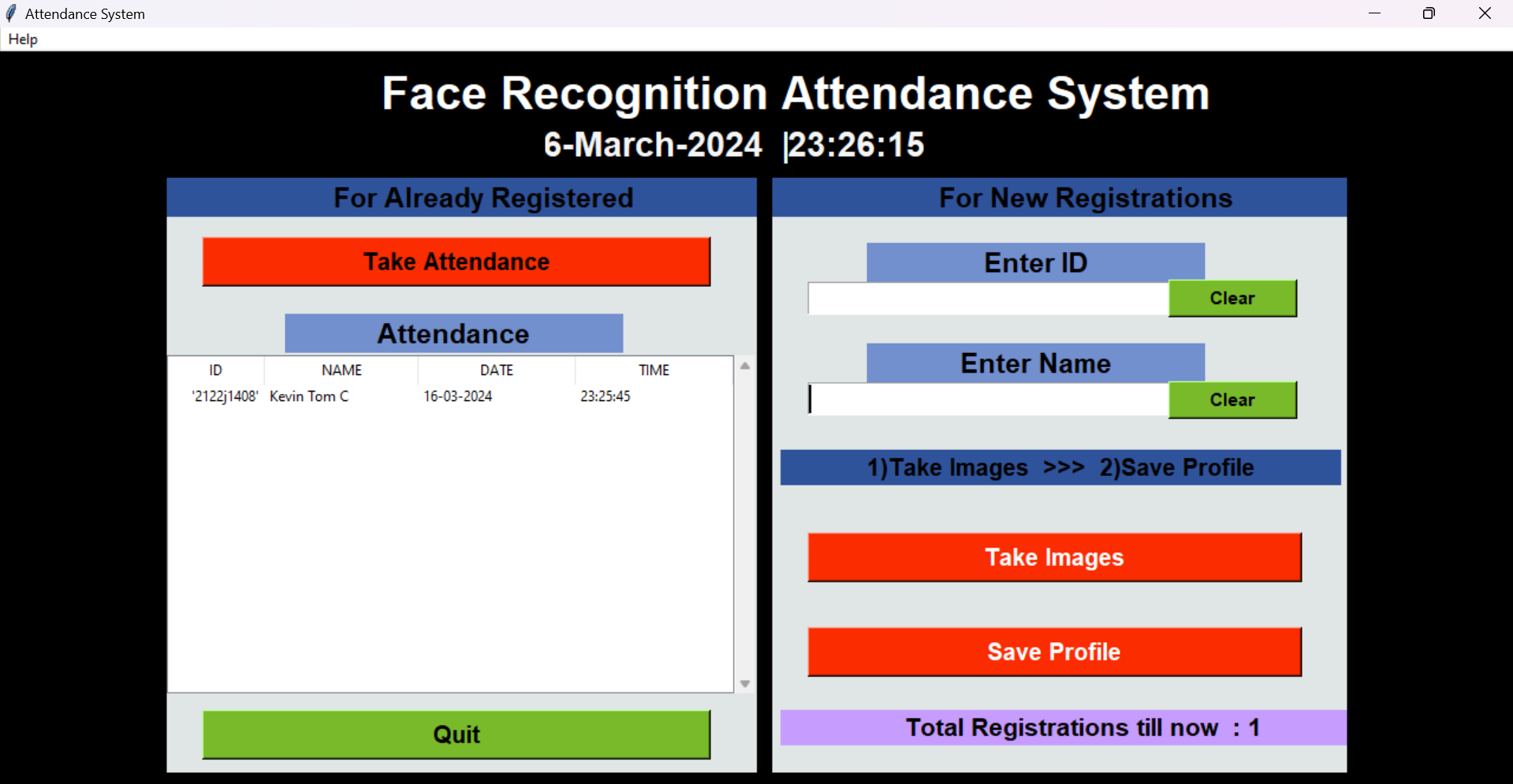
**Fig C.1 Face Training**

**D. SAMPLE OUTPUT**



**Fig D.1 Detect And Finding**

**OUTPUT 2:**



**Fig D.2Report Generation**