Smart Attendance System Using Face Recognition

Version 2.3.0

Major Project (ICI651)

BCA (CTIS)

BACHELOR OF COMPUTER APPLICATION
(Cloud Technology and Information Security)

PROJECT GUIDE:	SUBMITTED BY:	
Mr. Aatish Kumar Baitha	**Gulvesh**	**(TCA2256029)**
Mr. Umesh Chandra Garjola	**Anmol**	**(TCA2256010)**
	Asheesh kumar	**(TCA2256015)**

April, 2025

FACULTY OF ENGINEERING & COMPUTING SCIENCES

TEERTHANKER MAHAVEER UNIVERSITY, MORADABAD

DECLARATION

We hereby declare that this Project Report titled Smart Library Management System submitted by us and approved by our project guide, Faculty of Engineering & Computing Sciences. Teerthanker Mahaveer University, Moradabad, is a bonafide work undertaken by us and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.

Project ID :

Student Name:	**Gulvesh**	Signature
Student Name:	**Anmol**	Signature
Student Name:	**Asheesh kumar**	Signature
		Signature
Project Guide :	**Mr. Aatish Kumar Baitha**	Signature
Project Guide :	**Mr. Umesh Chandra Garjola**	Signature

Table of Contents

Appendix

1. Project Title

Smart Attendance System Using face Recognition

2. Problem Statement

Traditional attendance systems are often prone to proxy attendance, manual errors, and time-consuming processes. There is a strong need for a smart, contactless, and automated system that marks attendance accurately using facial recognition technology.

To overcome these challenges, the topic of a Smart Attendance System using face recognition using Python was chosen. The key motivation behind selecting this project is the increasing demand for digitization and automation in educational environments. The project aims to provide an efficient, reliable, and user-friendly solution that can assist librarians in managing library operations more effectively while offering students an easier way to access and use library resources.

The Smart Attendance System is an innovative solution designed to automate the attendance making process by recognizing students' faces through a live webcam feed. The system uses Python, OpenCV, and face_recognition libraries for face detection and matching, and stores the attendance data securely in MySQL database.

It eliminate manual efforts, prevents proxy attendance, enhanace security, and provides a contactless experience, which is highly relevant in the post-pandemic word.

The project aims to be accurate, time-efficient, secure, and user-friendly by implementing real-time face recognition technologies.

This project belongs to the domains of Artificial Intelligence (AI), Machine Learning (ML), and Computer Vision.

It primarily utilizes face recognition algorithms to automate attendance management.

The project combines techniques like image processing, pattern recognition, and real-time data handling to create a smart, contactless attendance system.

3. Project Description

The Smart Attendance System is an innovative solution designed to automate the attendance making process by recognizing students' faces through a live webcam feed. The system uses Python, OpenCV, and face_recognition libraries for face detection and matching, and stores the attendance data securely in MySQL database.

It eliminate manual efforts, prevents proxy attendance, enhanace security, and provides a contactless experience, which is highly relevant in the post-pandemic word.

The project aims to be accurate, time-efficient, secure, and user-friendly by implementing real-time face recognition technologies.Scope of the Work

3.1.1.1 In-Scope:

- Capture real-time images through a webcam.
- Detect and recognize faces using trained data.
- Mark attendance automatically without manual intervention.
- Store attendance records with timestamp into a structured database.
- Provide a simple GUI for users to operate easily.

3.1.1.2 Out of Scope:

- Facial recognition in extremely low-light or backlit environments.
- Handling recognition if the face is partially covered (like masks, scarves).
- Advanced biometric features such as fingerprint or iris scanning.
- Cloud database storage (e.g., Firebase or AWS – reserved for future scope).
- Mobile application version for attendance (planned for future work).
- Large-scale deployment for multiple campuses simultaneously.

3.2 Project Modules

- **Face Registration Module: Store face images of all students/employees for training the system.**
- **Face Detection Module**: Capture and detect faces using webcam and OpenCV.
- **Face Recognition Module**: Match detected face against stored face encodings.
- **Face Encoding & Recognition**: Encode facial features and recognize them using face_recognition library.
- **Attendance Marking Module**: Automatically mark and save attendance with date and time.
- **Database Management Module**: Store and manage attendance records in MySQL.
- **Database Integration**: Store attendance data in MySQL database.
- **GUI Interface**: Use Tkinter to build and interactive graphical user interface.

3.2.1.1 Main Menu & Navigation

- Acts as the central hub for users after login.
- Based on user role (Admin or Student), directs to the respective panel.
- Provides access to major features through buttons or menus.

3.2.1.2 Admin Panel (Attendance.py)

- To open the main pages on the project
- First you click the New Registration and then track the images
- And the Take attendance in the subject bise
- Select the any subject
- then click the show attendance in the View button

3.2.1.3 Smart Attendance Management Module

- **Attendance (Attendance.py)**: Adds student attendance in the excel.
- **Project Snap (Project Snap)**: Store in the one by one picture in the structure.
- **StudentDetails (StudentDetails.cvs)**: Store in the student attendance in the cvs file.

- **UI_image(UI_Image)**: Store UI image in the file.
- **Attendance (attendance**.py): To run the program to the smart attendance
- **automaticAttendance (automaticAttendance.**py): To store automatic attendance in the project
- **show_attendance (show_attendance.**py): To check the student attendance in details.
- takeImage (takeImage.py): To take the student image in the file.
- trainImage (trainImage.py): To train the image in automatic image.

3.2.1.4 Database Connection Module (Connect_DB.py)

- Establishes connection with the MySQL database.
- Executes SQL queries for data insertion, retrieval, and updates.

3.3 Context Diagram (High Level)

At Level 0, the context diagram defines the interaction between the user and the system, showing all inputs and outputs.

3.3.1.1 External Entities:

- **Admin**
- **Student**

3.3.1.2 Inputs from Admin:

- Take new registration and train the image
- take the attendance in the project
- view all student attendance
- exit the project in the module

3.3.1.3 Outputs to Admin:

- New Registration conform
- To take the attendance live webcam
- View attendance to show the subject bise attendance

3.3.1.4 Inputs from Student:

- Take new registration and train the image
- student give the face attendance
- view all student attendance

- exit the project in the modul

3.3.1.5 Outputs to Student:

- Day to day attendance show the student
- attendance Availability Status
- Student Panel Overview

4. Implementation Methodology

The project follows a systematic development methodology focused on quick prototyping, modular testing, and real-time integration.

The major goal is to build an efficient and accurate attendance system using face recognition technologies and database management.

4.1.1 Process Flow

The process begins with **user authentication**. Based on the role (Admin or Student), the respective dashboard is loaded. Webcam to live detection the face in camera.

All operations interact with the **central MySQLdatabase**, and user-friendly feedback is given via GUI windows.

4.1.2 Supporting Diagrams

To better understand the structure and flow of the software system, the following diagrams/models can be created:

4.1.2.1 Data Flow Diagram (DFD - Level 0 and 1)

- **Level 0** shows the interaction between users (Admin, Student) and the system.
- **Level 1** Detected the face to take the attendance in

4.1.2.2 Use Case Diagram

- Actors: Admin, Student
- Use Cases:
  - Admin: Add the Student attendance in the file
  - Student: Views the attendance in the subject wise the organization part.

4.1.2.3 Flowchart

- Student to face detected the camera to take the student attendance in subject wise. The show the attendance to student in the camera go from the attendance.

5. Technologies to be used

This section highlights the software and hardware platforms, along with tools used during the development of the **Smart Attendance System Using Face Recognition**. The project leverages lightweight technologies for fast, local execution and simplified development.

.

5.1 Software Platform

a) Front-end

- **Technology Used:** Python `Tkinter` Library, face recognition
- **Purpose:** Used to create the Graphical User Interface (GUI) for Admin and Student Form. Provides buttons, labels, text fields, pop-up messages, and window navigation.

.

b) Back-end

- **Technology Used:** Python + MySQL
- **Purpose: :** Used to store the attendance records (name, date, time).

Provides structure data management and easy retrieval

5.2 Hardware Platform

Component	Specification
RAM	Minimum 4 GB (Recommended 8 GB)
Hard Disk	Minimum 100 MB free space
Processor	Intel Core i3 or higher
Operating System	Windows 10/11 or Linux (Ubuntu preferred)
Code Editor	Visual Studio Code (VS Code)
Python Version	Python 3.9 or above
Database	MySQL (built-in with Python)
Browser	Not required (project is desktop-based)

5.3 Tools

The following tools were used during various phases of the software development life cycle:

Tool Name	Vendor Name	Version	Purpose of Use
Visual Studio Code	Microsoft	1.88	Source code writing, file management, debugging
DB Browser for MySQL	DB Browser Team	3.12.2	Visual management and testing of MySQL database
Python	Python Software	3.11	Core programming language used

6. Advantages of this Project

- The The **Smart Attendance System Using Face Recognition** System provides several clear advantages over traditional and manual attendance taking methods :

6.1.1 Contactless and Hygienic

- Eliminates the need for physical contact (like fingerprint scanners or signing registers).
- Very important in a post-COVID world for maintain hygiene and safety.

6.1.2 Accurate and Reliable

- Reduces human errors like incorrect entries or missed attendance.
- Uses biometric facial features which are unique to every individual, ensuring high accuracy.
-

6.1.3 Time-Efficient

- Automates the entire attendance process.
- Marks attendance in real-time within 2–3 seconds of face detection, saving significant classroom or meeting time.

6.1.4 Prevents Proxy Attendance

- Face recognition technology ensures only the genuine person is marked present.
- Stops friends or colleagues from marking proxy attendance

6.1.5 Automate Data storage

- Attendance data (Name, Date, Time) is stored directly into a MySQL database.
- Easy to generate reports, analyze data, and maintain records digitally.

6.1.6 User-Friendly Interface

- Built with Tkinter GUI, making the system easy to use even for non-technical users.
- Start/Stop Camera buttons, Status Messages, and System Logs improve usability.

6.1.7 Scalable and Flexible

- Can be scaled up for use in multiple classrooms, offices, or even organizations.
- System is flexible to add new faces anytime without much reconfiguration.

6.1.8 Cost-Effective

- Uses open-source technologies (Python, OpenCV, MySQL) — No need for expensive biometric devices.
- Runs easily on basic hardware like laptops with webcams

7. Assumptions

The following assumptions were made during the development of the Smart Attendance System Using Face Recognition:

1. **User Access:**
   - It is assumed that only registered users (Admin and Students) will access the system.
   - Admin credentials will not be shared or misused.
2. **Hardware Requirements:**
   - The system will run on a machine with basic configurations (e.g., 4GB RAM, Windows 10+).
   - A stable environment for GUI operations is available.
3. **Database Integrity:**
   - The MySQL database will not be manually altered or corrupted by users.
   - Data entered into the system is assumed to be accurate and complete.
4. **Library Environment:**
   - It is assumed that the library has internet access only if cloud features or email reminders are implemented in the future.
   - Book inventory is properly tagged and updated before system deployment.
5. **Face Use:**
   - Face are functional and properly configured.
   - Every person has a unique and scannable face code generated within the system.
6. **User Knowledge:**
   - Users (especially Admins) are assumed to have basic computer knowledge and can use GUI-based systems.

8. Future Scope and further enhancement of the Project

The **Smart Library Attendance System Using Face Recognition** Although the current system efficiently automates attendance using face recognition, there are several ways to enhance its functionality and make it even robust and scalable.

:

8.1.1 Cloud-Based Data based Integration

- Future versions can connect to cloud databases like Firebase, AWS RDS, or Google Cloud SQL.
- This will enable centralized attendance storage accessible across multiple campuses or remote locations.

8.1.2 Face Mask Detection

- Implement a model that can recognize faces even when the student is wearing a mask.
- Post-pandemic, face mask detection is a crucial real-world enhancement.

8.1.3 Multiple Camera Support

- Enable multi-camera setup to cover large classrooms, auditoriums, or open office spaces.
- Attendance can be captured simultaneously from different locations.

8.1.4 Real-Time Notifications System

- Send instant notifications (SMS/Email) to students/parents if attendance is missed or marked.
- Improves transparency and communication between institutions and students.

8.1.5 Attendance Analytics Dashboard

- Build a dashboard using Flask /Django showing:
- Daily, weekly, monthly attendance trends
- Student-wise reports
- Defaulters list
- Useful for management decisions and performance tracking.

8.1.6 Artificial Intelligent-Based Attendance Prediction

- AI models can predict expected attendance trends based on previous data.
- Helps institutions forecast and plan resource allocation.

8.1.7 Security Enhancement

- Add multi-factor authentication (like OTPs) for administrative logins.
- Improve data encryption while storing student records in databases.

9. Project Repository Location

The following table lists the locations of all key artifacts related to the **Smart Attendance System Using Face Recognition** project. These files are stored in the CCSIT lab infrastructure for access, verification, and record-keeping.*-Lab*

S#	Project Artifacts (softcopy)	Location (Mention Lab-ID, Server ID, Folder Name etc.)	Verified by Project Guide	Verified by Lab In-Charge
1.	Project Synopsis Report (Final Version)	LAB-CCSIT-03 / Server-02 / Library_Management_2025 /Synopsis_Final	Name and Signature	Name and Signature
2.	Project Progress updates	LAB-CCSIT-03 / Server-02 / Library_Management_2025 /Progress_Updates	Name and Signature	Name and Signature
3.	Project Requirement specifications	LAB-CCSIT-03 / Server-02 / Library_Management_2025 /Requirements	Name and Signature	Name and Signature
4.	Project Report (Final Version)	LAB-CCSIT-03 / Server-02 / Library_Management_2025 /Report_Final	Name and Signature	Name and Signature
5.	Test Repository	LAB-CCSIT-03 / Server-02 / Library_Management_2025 /Testing	Name and Signature	Name and Signature
6.	Project Source Code (final version) with executable	LAB-CCSIT-03 / Server-02 / Library_Management_2025 /Source_Code	Name and Signature	Name and Signature
7.	Any other document	LAB-CCSIT-03 / Server-02 / Library_Management_2025 /Supporting_Docs	Name and Signature	Name and Signature

10. Definitions, Acronyms, and Abbreviations

The following table provides definitions and explanations of the technical terms, abbreviations, and acronyms used throughout the project documentation. This is intended to ensure clear understanding and interpretation of the content in the Software Requirement Specification (SRS) and related documents.

Abbreviation	Description
SAMS	**Smart Attendance Management System** – A software designed to manage Student attendance operations.
GUI	**Graphical User Interface** – A user interface that includes graphical elements like buttons, icons, and windows.
DBMS	**Database Management System** – Software for storing, retrieving, and managing data in databases.
SQL	**Structured Query Language** – A language used for interacting with databases.
SRS	**Software Requirements Specification** – A document that describes the functionalities and requirements of the software system.
UI	**User Interface** – The visual part of a software application that users interact with.
CRUD	**Create, Read, Update, Delete** – Basic operations performed on database records.
DFD	**Data Flow Diagram** – A visual representation of the flow of data in a system.
ER Diagram	**Entity Relationship Diagram** – A diagram that shows the relationships between entities in a database.
Face Code	**FaceResponse Code** – A machine-readable face used for webcam live scan student image in the camera.
Admin	**Administrator** – A user with full control and management access within the system.

11. Conclusion

The Smart Attendance System using Face Recognition successfully automates the traditional attendance marking process by using advanced machine learning and computer vision techniques. By using Python, OpenCV and face recognition, the system accurately detects and recognizes student faces through a live webcam feed and marks attendance in a MySQL database without any manual intervention.

The project achieves:

High accuracy in face matching,

Fast processing within seconds,

Secure, contactless attendance,

Prevention of proxy attendance,

And easy record maintenance via an interactive GUI

This system not only enhances operational efficiency in educational institutions and workplaces but also promotes a hygienic and reliable attendance solution, especially necessary in a post-pandemic world. With future enhancements like cloud integration, mobile app support, and AI-driven analytics, this system holds strong potential for wide-scale deployment across different industries. Thus, this project serves as a practical, efficient, the Smart Attendance System using Face Recognition successfully automates the traditional attendance marking process by using advanced machine learning and computer vision techniques and scalable solution for modern attendance management.

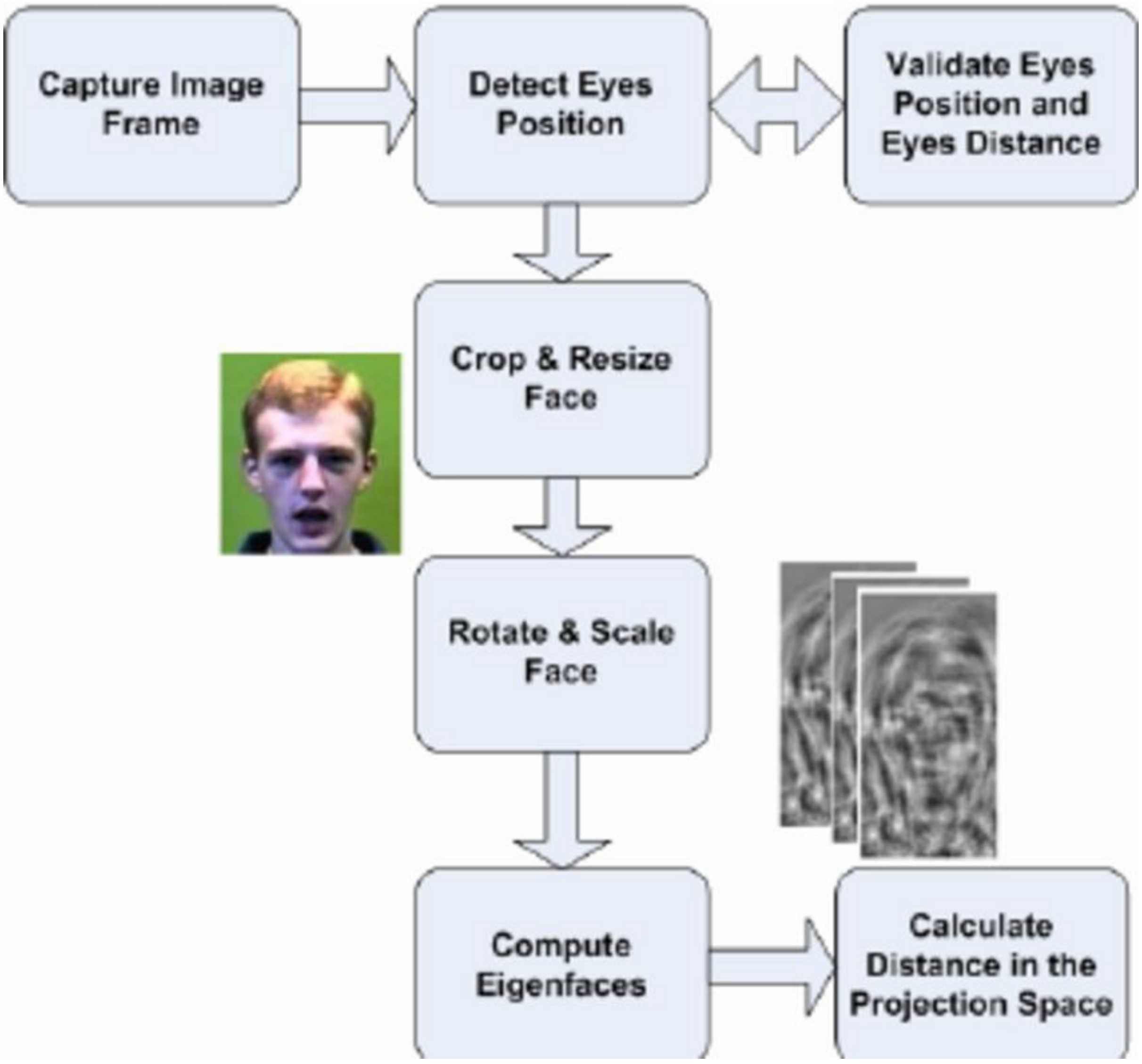
12. References

This section provides a complete list of all documents and resources referenced throughout the Software Requirements Specification (SRS) and project documentation. These references serve as a basis for understanding system design, development, and testing standards.

S#	Reference Details	Owner	URLs
1.	Python 3.10 Documentation	Python Software Foundation	https://docs.python.org/3/
2.	MySQL Documentation	MySQL.org	https://www.Mysql.org/docs.html
3.	Geeksforgeeks	Geeksforgeeks.org	https://www.geeksforgeeks.org/python-programming-language-tutorial/

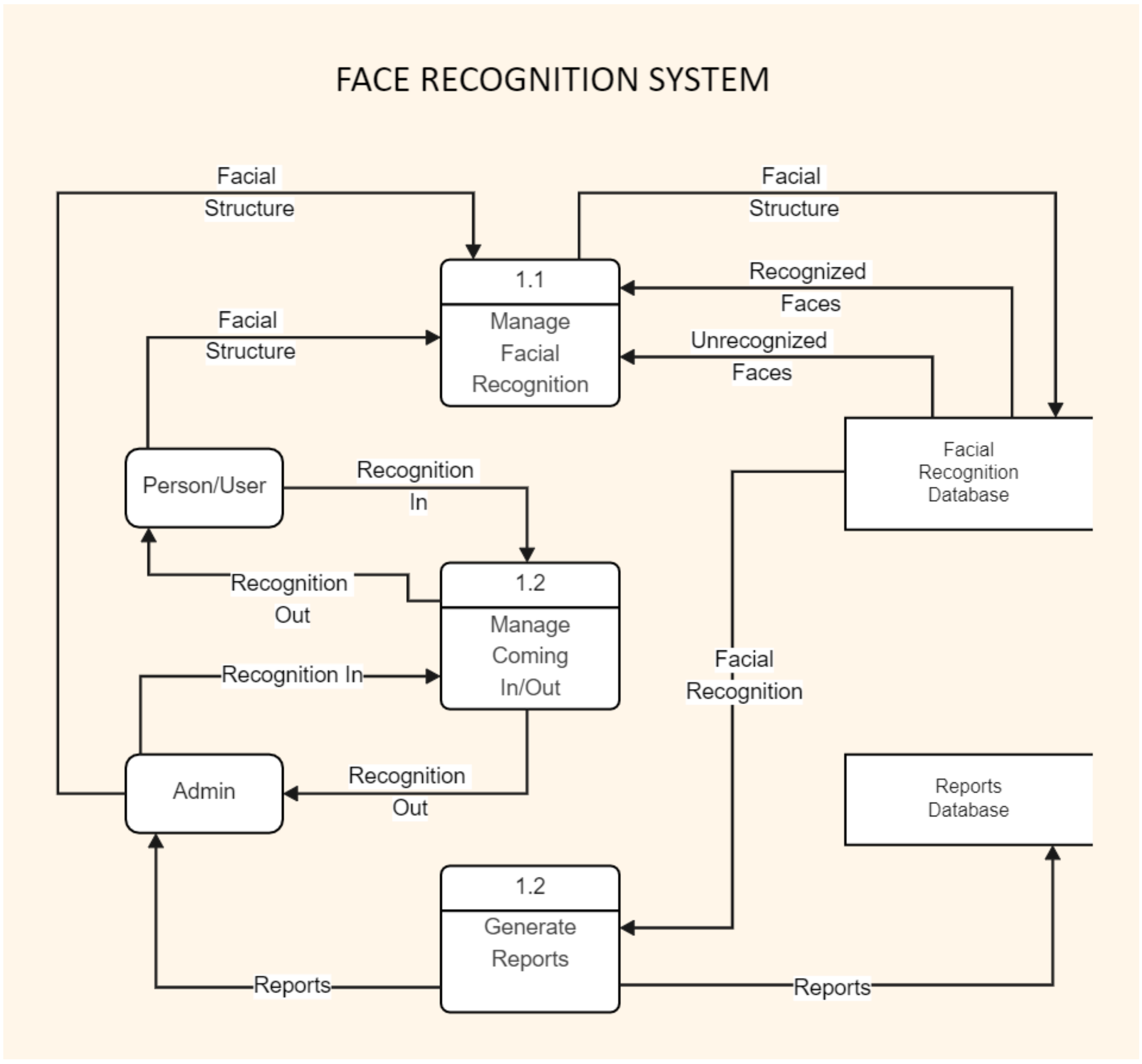
1. **OpenCV Official Documentation:** Image processing and face detection.
2. **Face_recognition Github Repository:** face recognition library for python.
3. **Python Official Documentation:** Python programming language.
4. **Tkinter Documentation:** Graphical User Interface (GUI) Development.
5. **MySQL Official Documentation:** Database management and SQL queries.
6. **Real Python Totorials:** Learning resources for python and related technologies.

Annexure A
Data Flow Diagram (DFD)



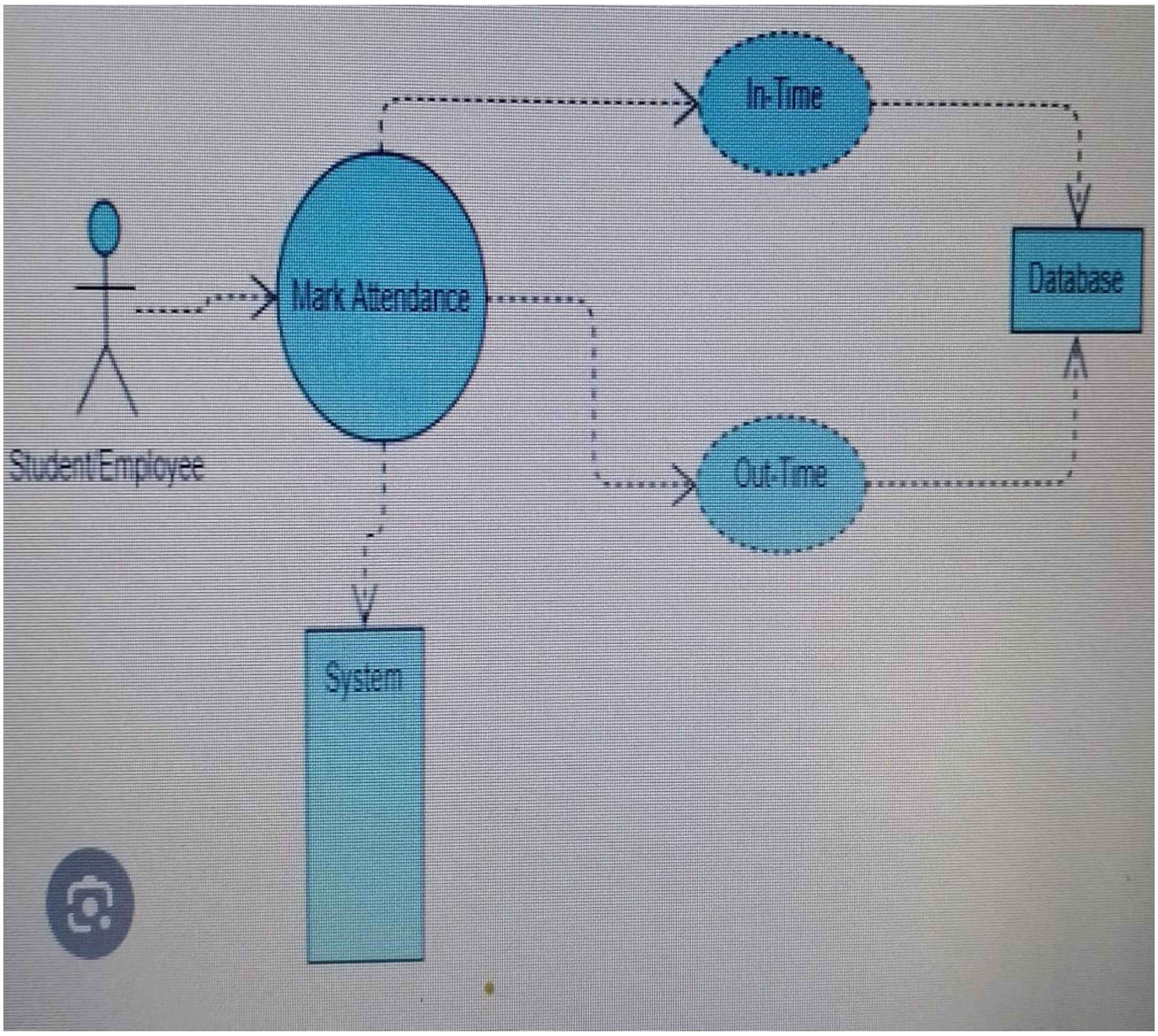
Annexure B
Entity-Relationship Diagram (ERD)

(Mandatory)



Annexure C

Use-Case Diagram (UCD)



Annexure D
Data Dictionary (DD)

Table: Attendance

Field Name	Data Type	Description	Constraints
ID	VARCHAR(100)	Unique identifier for each book	Primary Key, Not Null
Title	VARCHAR(100)	Title of the book	Not Null
Author	VARCHAR(100)	Name of the book's author	Optional
Availability	BOOLEAN	Status of book (Available or Issued)	Default TRUE

Table: Students

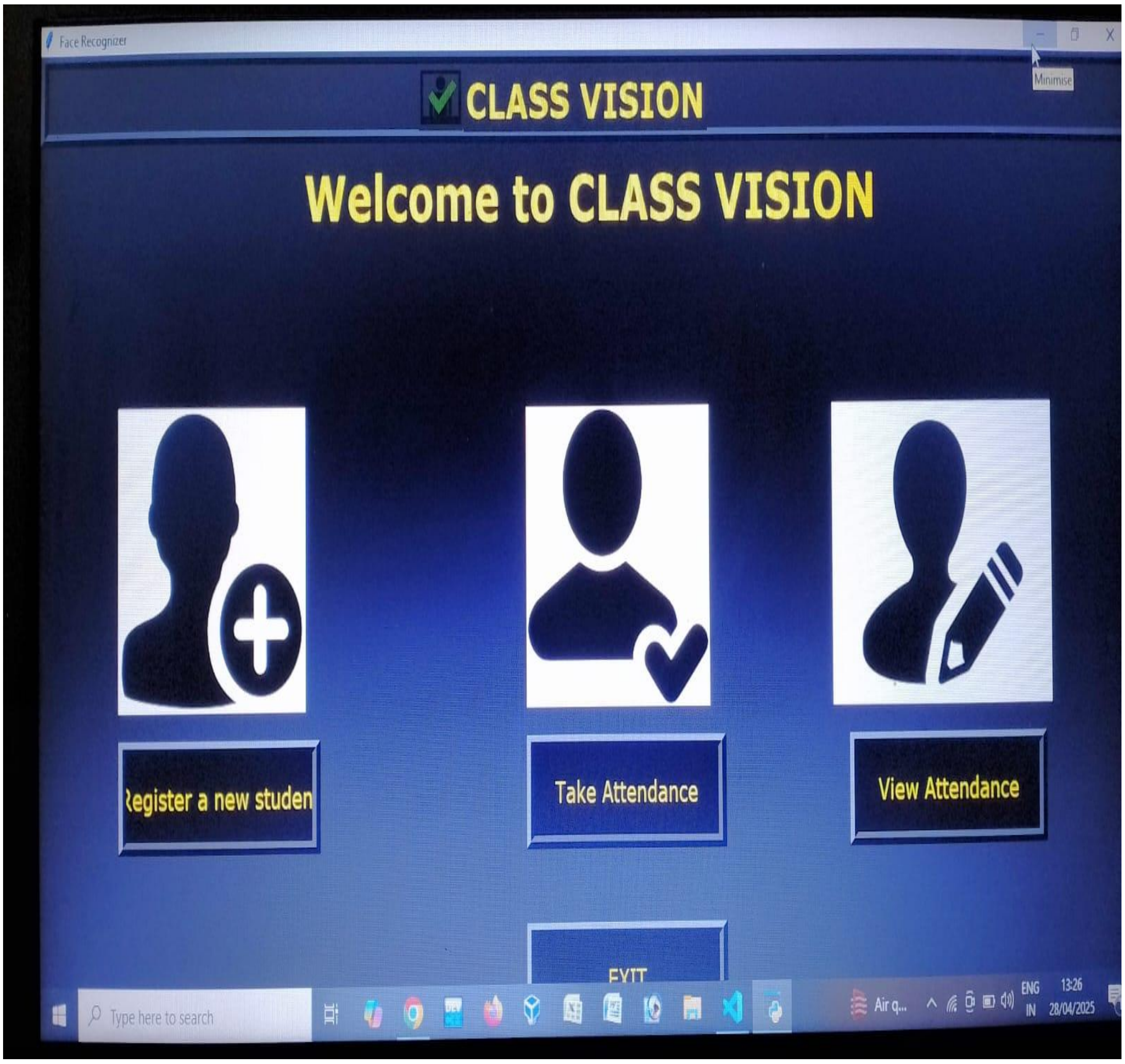
Field Name	Data Type	Description	Constraints
ID	VARCHAR(20)	Unique student ID	Primary Key, Not Null
Name	VARCHAR(100)	Full name of the student	Not Null
RollNo	VARCHAR(20)	Unique Roll Number	Unique, Not Null
Contact	VARCHAR(15)	Student's contact number	Optional

Table: Transactions

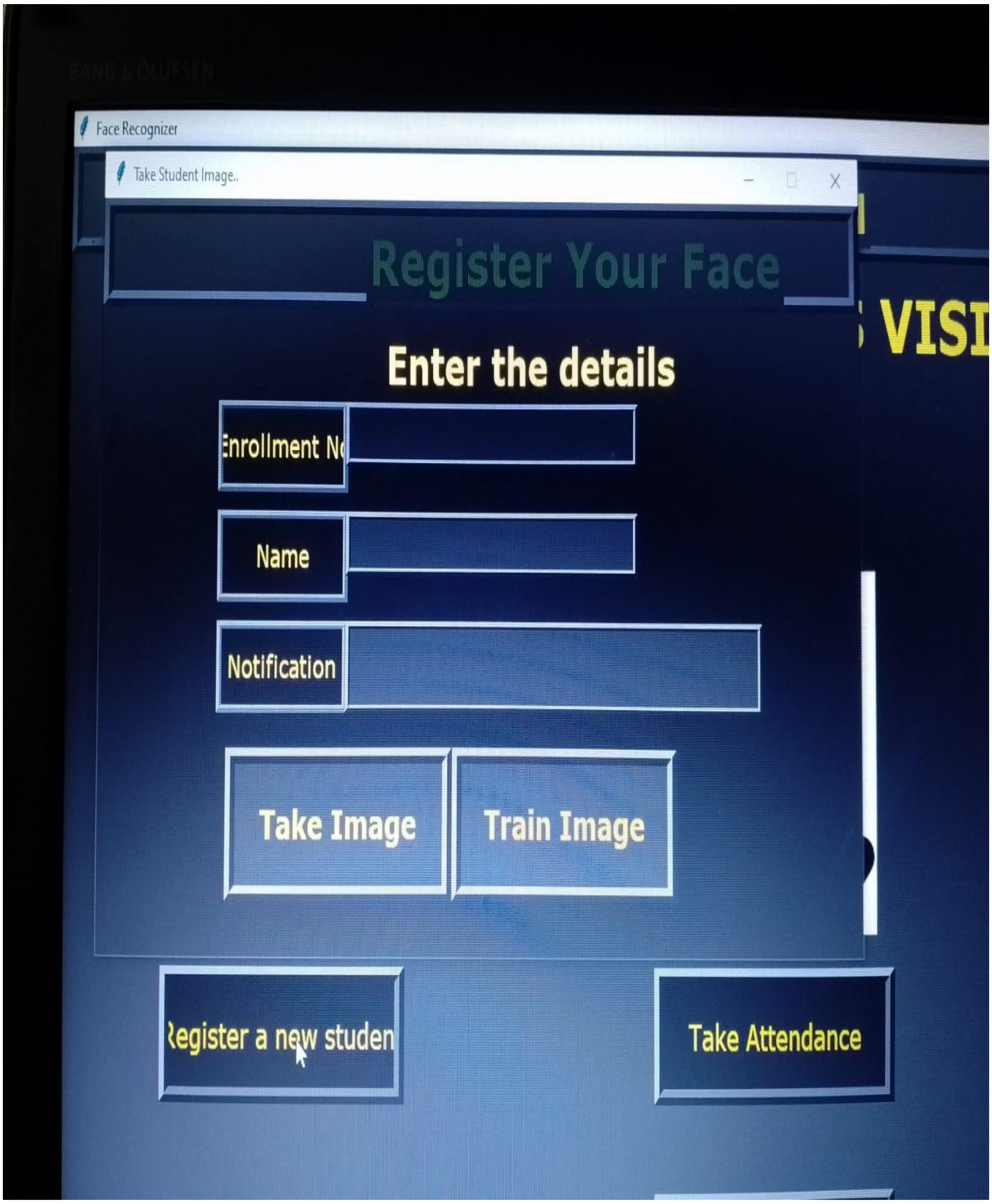
Field Name	Data Type	Description	Constraints
StudentID	INT	Unique identifier for each transaction	Primary Key, Auto Increment
FaceID	VARCHAR(20)	Book being issued/returned	Foreign Key → Books(ID)
StudentID	VARCHAR(20)	Student involved in the transaction	Foreign Key → Students(ID)
AttendanceDate	DATE	Date when the book was issued	Optional (based on logic)
ReturnDate	DATE	Date when the book was returned	Optional (based on logic)

Annexure E
Screen Shots

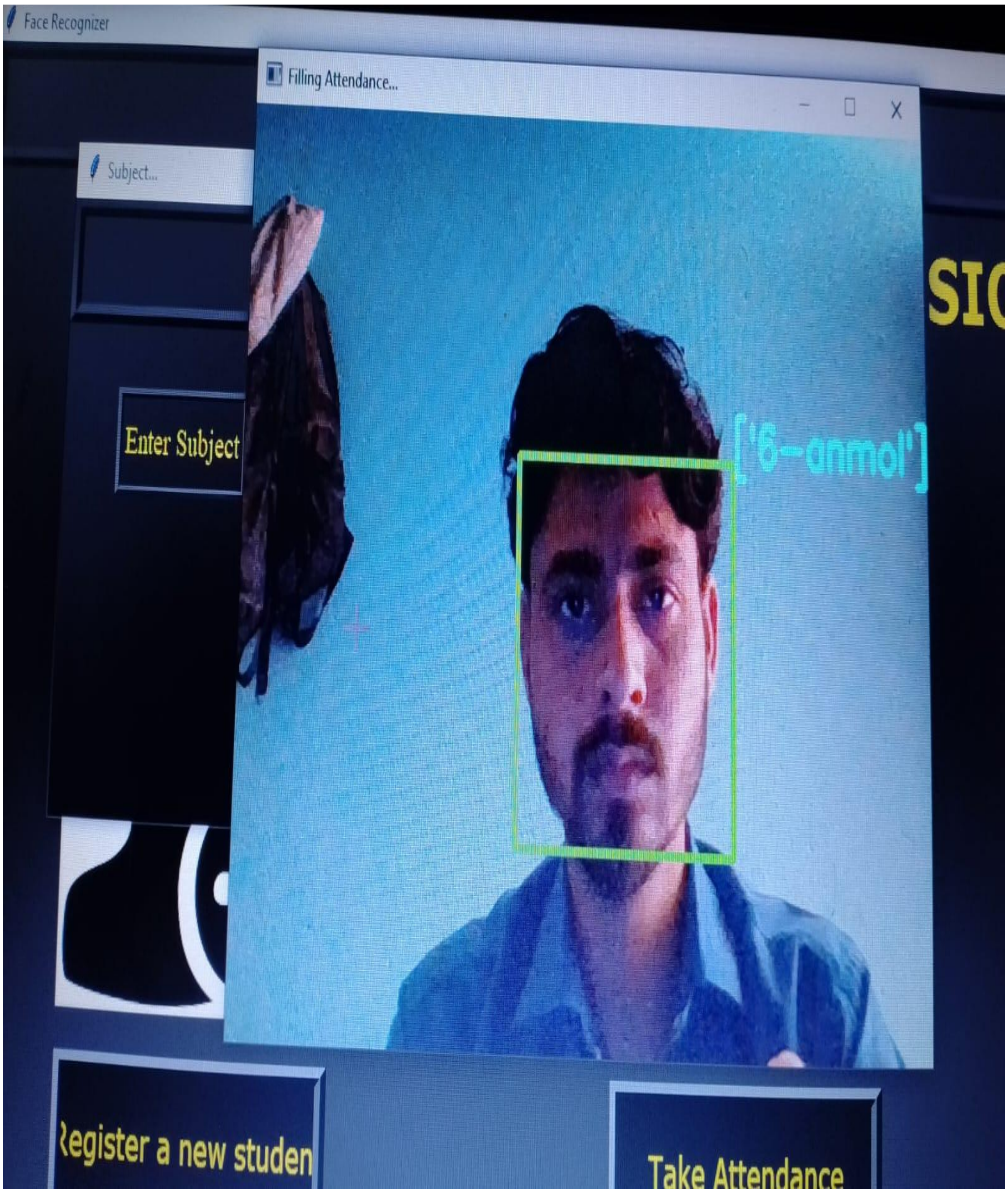
Home Page:



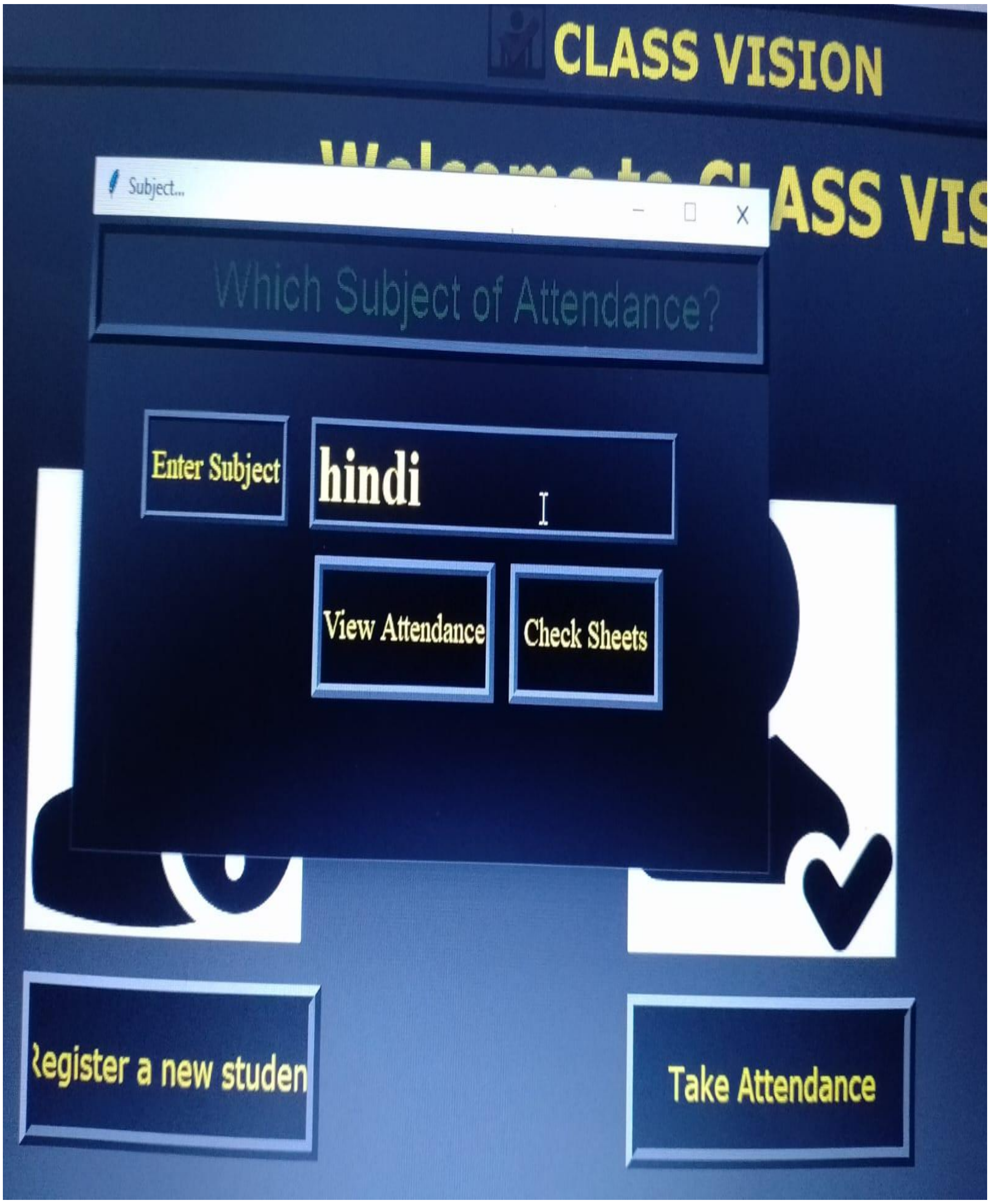
1.Registration form :



2.Subject Attendance Detect the face:

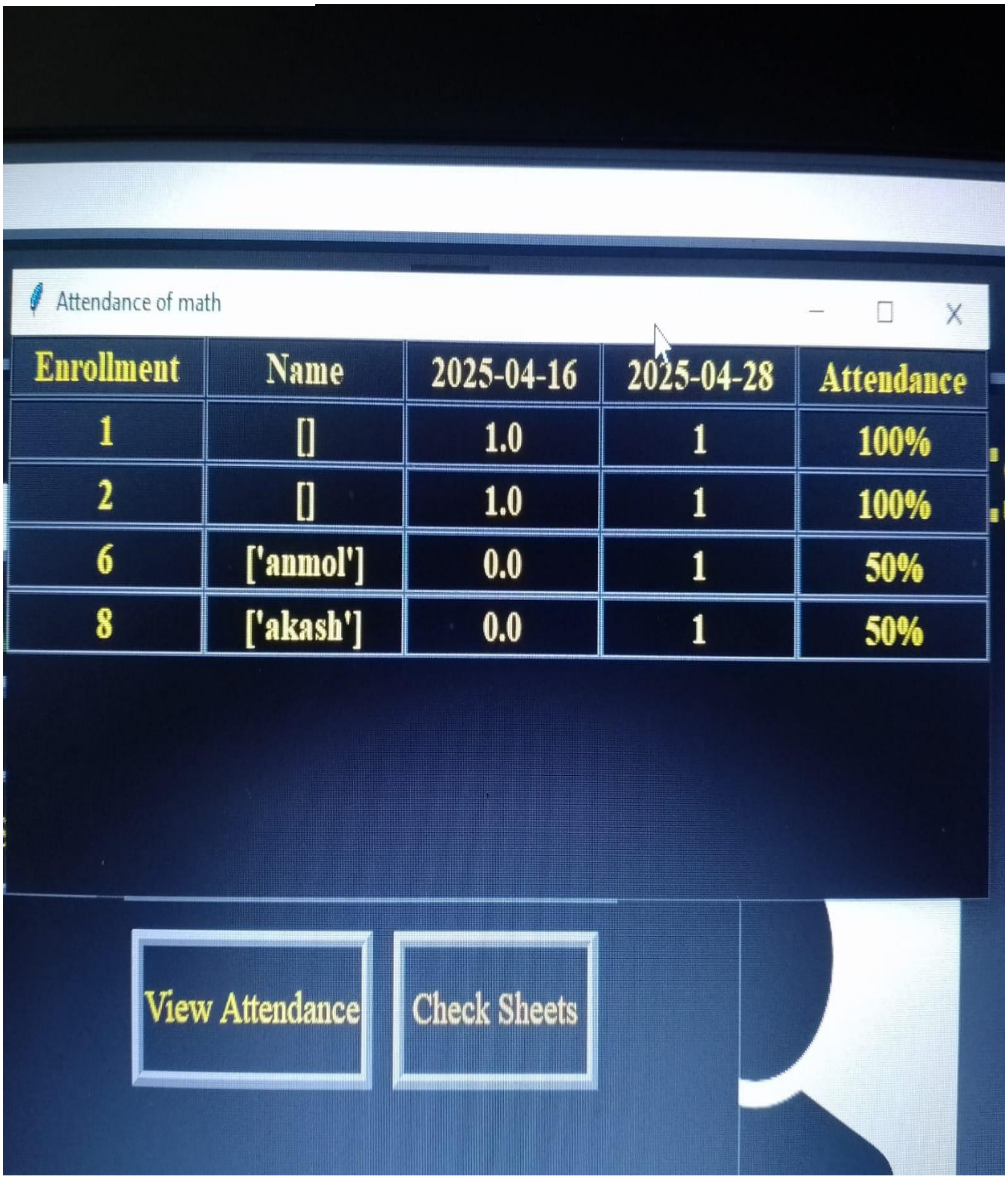


3.Subject by Attendance :



Enter Subject hindi

4.All Attendance



Attendance of math

Enrollment	Name	2025-04-16	2025-04-28	Attendance
1	[]	1.0	1	100%
2	[]	1.0	1	100%
6	['anmol']	0.0	1	50%
8	['akash']	0.0	1	50%

Annexure D
Project Code

Project code file (module)

1. Attendance.py

```python
import tkinter as tk
from tkinter import *
import os, cv2
import shutil
import csv
import numpy as np
from PIL import ImageTk, Image
import pandas as pd
import datetime
import time
import tkinter.font as font
import pyttsx3

# project module
import show_attendance
import takeImage
import trainImage
import automaticAttedance

# engine = pyttsx3.init()
# engine.say("Welcome!")
# engine.say("Please browse through your options..")
# engine.runAndWait()

def text_to_speech(user_text):
    engine = pyttsx3.init()
    engine.say(user_text)
    engine.runAndWait()

haarcasecade_path = "haarcascade_frontalface_default.xml"
trainimagelabel_path = (
    "./TrainingImageLabel/Trainner.yml"
)
trainimage_path = "/TrainingImage"
if not os.path.exists(trainimage_path):
    os.makedirs(trainimage_path)

studentdetail_path = (
    "./StudentDetails/studentdetails.csv"
)
attendance_path = "Attendance"

window = Tk()
window.title("Face Recognizer")
window.geometry("1280x720")
dialog_title = "QUIT"
```

```python
    dialog_text = "Are you sure want to close?"
    window.configure(background="#1c1c1c")  # Dark theme

    # to destroy screen
    def del_sc1():
        sc1.destroy()

    # error message for name and no
    def err_screen():
        global sc1
        sc1 = tk.Tk()
        sc1.geometry("400x110")
        sc1.iconbitmap("AMS.ico")
        sc1.title("Warning!!")
        sc1.configure(background="#1c1c1c")
        sc1.resizable(0, 0)
        tk.Label(
            sc1,
            text="Enrollment & Name required!!!",
            fg="yellow",
            bg="#1c1c1c",  # Dark background for the error window
            font=("Verdana", 16, "bold"),
        ).pack()
        tk.Button(
            sc1,
            text="OK",
            command=del_sc1,
            fg="yellow",
            bg="#333333",  # Darker button color
            width=9,
            height=1,
            activebackground="red",
            font=("Verdana", 16, "bold"),
        ).place(x=110, y=50)

    def testVal(inStr, acttyp):
        if acttyp == "1":  # insert
            if not inStr.isdigit():
                return False
        return True

    logo = Image.open("UI_Image/0001.png")
    logo = logo.resize((50, 47), Image.LANCZOS)
    logo1 = ImageTk.PhotoImage(logo)
    titl = tk.Label(window, bg="#1c1c1c", relief=RIDGE, bd=10, font=("Verdana", 30, "bold"))
    titl.pack(fill=X)
    l1 = tk.Label(window, image=logo1, bg="#1c1c1c",)
    l1.place(x=470, y=10)

    titl = tk.Label(
        window, text="CLASS VISION", bg="#1c1c1c", fg="yellow", font=("Verdana", 27, "bold"),
    )
    titl.place(x=525, y=12)

    a = tk.Label(
        window,
        text="Welcome to CLASS VISION",
        bg="#1c1c1c",  # Dark background for the main text
        fg="yellow",  # Bright yellow text color
        bd=10,
        font=("Verdana", 35, "bold"),
    )
    a.pack()

    ri = Image.open("UI_Image/register.png")
    r = ImageTk.PhotoImage(ri)
    label1 = Label(window, image=r)
    label1.image = r
    label1.place(x=100, y=270)

    ai = Image.open("UI_Image/attendance.png")
    a = ImageTk.PhotoImage(ai)
    label2 = Label(window, image=a)
    label2.image = a
    label2.place(x=980, y=270)

    vi = Image.open("UI_Image/verifyy.png")
    v = ImageTk.PhotoImage(vi)
    label3 = Label(window, image=v)
    label3.image = v
    label3.place(x=600, y=270)

    def TakeImageUI():
        ImageUI = Tk()
        ImageUI.title("Take Student Image..")
        ImageUI.geometry("780x480")
        ImageUI.configure(background="#1c1c1c")  # Dark background for the image window
        ImageUI.resizable(0, 0)
        titl = tk.Label(ImageUI, bg="#1c1c1c", relief=RIDGE, bd=10, font=("Verdana", 30, "bold"))
        titl.pack(fill=X)
        # image and title
        titl = tk.Label(
            ImageUI, text="Register Your Face", bg="#1c1c1c", fg="green", font=("Verdana", 30, "bold"),
        )
        titl.place(x=270, y=12)

        # heading
        a = tk.Label(
            ImageUI,
            text="Enter the details",
            bg="#1c1c1c",  # Dark background for the details label
            fg="yellow",  # Bright yellow text color
            bd=10,
            font=("Verdana", 24, "bold"),
        )
        a.place(x=280, y=75)

        # ER no
        lbl1 = tk.Label(
            ImageUI,
            text="Enrollment No",
            width=10,
            height=2,
            bg="#1c1c1c",
            fg="yellow",
            bd=5,
            relief=RIDGE,
            font=("Verdana", 14),
        )
        lbl1.place(x=120, y=130)
        txt1 = tk.Entry(
            ImageUI,
            width=17,
            bd=5,
            validate="key",
            bg="#333333",  # Dark input background
            fg="yellow",  # Bright text color for input
            relief=RIDGE,
            font=("Verdana", 18, "bold"),
        )
        txt1.place(x=250, y=130)
        txt1["validatecommand"] = (txt1.register(testVal), "%P", "%d")

        # name
        lbl2 = tk.Label(
            ImageUI,
            text="Name",
            width=10,
            height=2,
            bg="#1c1c1c",
            fg="yellow",
            bd=5,
            relief=RIDGE,
            font=("Verdana", 14),
        )
        lbl2.place(x=120, y=200)
        txt2 = tk.Entry(
            ImageUI,
            width=17,
            bd=5,
            bg="#333333",  # Dark input background
            fg="yellow",  # Bright text color for input
            relief=RIDGE,
            font=("Verdana", 18, "bold"),
        )
        txt2.place(x=250, y=200)
```

```

        lbl3 = tk.Label(
            ImageUI,
            text="Notification",
            width=10,
            height=2,
            bg="#1c1c1c",
            fg="yellow",
            bd=5,
            relief=RIDGE,
            font=("Verdana", 14),
        )
        lbl3.place(x=120, y=270)

        message = tk.Label(
            ImageUI,
            text="",
            width=32,
            height=2,
            bd=5,
            bg="#333333",  # Dark background for messages
            fg="yellow",  # Bright text color for messages
            relief=RIDGE,
            font=("Verdana", 14, "bold"),
        )
        message.place(x=250, y=270)

        def take_image():
            l1 = txt1.get()
            l2 = txt2.get()
            takeImage.TakeImage(
                l1,
                l2,
                haarcasecade_path,
                trainimage_path,
                message,
                err_screen,
                text_to_speech,
            )
            txt1.delete(0, "end")
            txt2.delete(0, "end")

        # take Image button
        # image
        takeImg = tk.Button(
            ImageUI,
            text="Take Image",
            command=take_image,
            bd=10,
            font=("Verdana", 18, "bold"),
            bg="#333333",  # Dark background for the button
            fg="yellow",  # Bright text color for the button
            height=2,
            width=12,
            relief=RIDGE,
        )
        takeImg.place(x=130, y=350)

        def train_image():
            trainImage.TrainImage(
                haarcasecade_path,
                trainimage_path,
                trainimagelabel_path,
                message,
                text_to_speech,
            )

        # train Image function call
        trainImg = tk.Button(
            ImageUI,
            text="Train Image",
            command=train_image,
            bd=10,
            font=("Verdana", 18, "bold"),
            bg="#333333",  # Dark background for the button
            fg="yellow",  # Bright text color for the button
            height=2,
            width=12,
            relief=RIDGE,
        )
        trainImg.place(x=360, y=350)

    r = tk.Button(
        window,
        text="Register a new student",
        command=TakeImageUI,
        bd=10,
        font=("Verdana", 16),
        bg="black",
        fg="yellow",
        height=2,
        width=17,
    )
    r.place(x=100, y=520)

    def automatic_attedance():
        automaticAttedance.subjectChoose(text_to_speech)

    r = tk.Button(
        window,
        text="Take Attendance",
        command=automatic_attedance,
        bd=10,
        font=("Verdana", 16),
        bg="black",
        fg="yellow",
        height=2,
        width=17,
```

```python
    )
    r.place(x=600, y=520)

    def view_attendance():
        show_attendance.subjectchoose(text_to_speech)

    r = tk.Button(
        window,
        text="View Attendance",
        command=view_attendance,
        bd=10,
        font=("Verdana", 16),
        bg="black",
        fg="yellow",
        height=2,
        width=17,
    )
    r.place(x=1000, y=520)
    r = tk.Button(
        window,
        text="EXIT",
        bd=10,
        command=quit,
        font=("Verdana", 16),
        bg="black",
        fg="yellow",
        height=2,
        width=17,
    )
    r.place(x=600, y=660)

    window.mainloop()

```

2 automaticAttendance.py

```
import tkinter as tk
from tkinter import *
import os, cv2
import shutil
import csv
import numpy as np
from PIL import ImageTk, Image
import pandas as pd
import datetime
import time
import tkinter.ttk as tkk
import tkinter.font as font

haarcasecade_path = "haarcascade_frontalface_default.xml"
trainimagelabel_path = (
    "TrainingImageLabel\\Trainner.yml"
)
trainimage_path = "TrainingImage"
studentdetail_path = (
    "StudentDetails\\studentdetails.csv"
)
attendance_path = "Attendance"
# for choose subject and fill attendance
def subjectChoose(text_to_speech):
    def FillAttendance():
        sub = tx.get()
        now = time.time()
        future = now + 20
        print(now)
        print(future)
        if sub == "":
            t = "Please enter the subject name!!!"
            text_to_speech(t)
        else:
            try:
                recognizer = cv2.face.LBPHFaceRecognizer_create()
                try:
                    recognizer.read(trainimagelabel_path)
                except:
                    e = "Model not found,please train model"
                    Notifica.configure(
                        text=e,
                        bg="black",
                        fg="yellow",
                        width=33,
                        font=("times", 15, "bold"),
                    )
                    Notifica.place(x=20, y=250)
                    text_to_speech(e)
                facecasCade = cv2.CascadeClassifier(haarcasecade_path)
                df = pd.read_csv(studentdetail_path)
                cam = cv2.VideoCapture(0)
                font = cv2.FONT_HERSHEY_SIMPLEX
                col_names = ["Enrollment", "Name"]
                attendance = pd.DataFrame(columns=col_names)
                while True:
                    ___, im = cam.read()
                    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
                    faces = facecasCade.detectMultiScale(gray, 1.2, 5)
                    for (x, y, w, h) in faces:
                        global Id

                        Id, conf = recognizer.predict(gray[y : y + h, x :
x + w])
                        if conf < 70:
                            print(conf)
                            global Subject
                            global aa
                            global date
                            global timeStamp
                            Subject = tx.get()
                            ts = time.time()
                            date =
datetime.datetime.fromtimestamp(ts).strftime(
                                "%Y-%m-%d"
                            )
                            timeStamp =
datetime.datetime.fromtimestamp(ts).strftime(
                                "%H:%M:%S"
                            )
                            aa = df.loc[df["Enrollment"] ==
Id]["Name"].values
                            global tt
                            tt = str(Id) + "-" + aa
                            # En='1604501160'+str(Id)
                            attendance.loc[len(attendance)] = [
                                Id,
                                aa,
                            ]
                            cv2.rectangle(im, (x, y), (x + w, y + h), (0,
260, 0), 4)
                            cv2.putText(
                                im, str(tt), (x + h, y), font, 1, (255,
255, 0,), 4
                            )
                        else:
                            Id = "Unknown"
                            tt = str(Id)
                            cv2.rectangle(im, (x, y), (x + w, y + h), (0,
25, 255), 7)
                            cv2.putText(
                                im, str(tt), (x + h, y), font, 1, (0, 25,
255), 4
                            )
                    if time.time() > future:
                        break

                    attendance = attendance.drop_duplicates(
                        ["Enrollment"], keep="first"
                    )
                    cv2.imshow("Filling Attendance...", im)
                    key = cv2.waitKey(30) & 0xFF
                    if key == 27:
                        break

                ts = time.time()
                print(aa)
                # attendance["date"] = date
                # attendance["Attendance"] = "P"
                attendance[date] = 1
                date = datetime.datetime.fromtimestamp(ts).strftime("%Y-
%m-%d")
                timeStamp =
datetime.datetime.fromtimestamp(ts).strftime("%H:%M:%S")
                Hour, Minute, Second = timeStamp.split(":")
                # fileName = "Attendance/" + Subject + ".csv"
                path = os.path.join(attendance_path, Subject)
                if not os.path.exists(path):
                    os.makedirs(path)
                fileName = (
                    f"{path}/"
                    + Subject
                    + "_"
                    + date
                    + "_"
                    + Hour
                    + "-"
                    + Minute
                    + "-"
                    + Second
                    + ".csv"
                )
                attendance = attendance.drop_duplicates(["Enrollment"],
keep="first")
                print(attendance)
                attendance.to_csv(fileName, index=False)

                m = "Attendance Filled Successfully of " + Subject
                Notifica.configure(
                    text=m,
                    bg="black",
                    fg="yellow",
                    width=33,
                    relief=RIDGE,
                    bd=5,
                    font=("times", 15, "bold"),
                )
                text_to_speech(m)
                Notifica.place(x=20, y=250)

                cam.release()
                cv2.destroyAllWindows()

                import csv
                import tkinter

                root = tkinter.Tk()
                root.title("Attendance of " + Subject)
                root.configure(background="black")
                cs = os.path.join(path, fileName)
                print(cs)
                with open(cs, newline="") as file:
                    reader = csv.reader(file)
                    r = 0

                    for col in reader:
                        c = 0
                        for row in col:

                            label = tkinter.Label(
                                root,
                                width=10,
                                height=1,
                                fg="yellow",
                                font=("times", 15, " bold "),
                                bg="black",
                                text=row,
                                relief=tkinter.RIDGE,
                            )
                            label.grid(row=r, column=c)
                            c += 1
                        r += 1
                root.mainloop()
                print(attendance)
            except:
                f = "No Face found for attendance"
                text_to_speech(f)
                cv2.destroyAllWindows()

    ###windo is frame for subject chooser
    subject = Tk()
    # windo.iconbitmap("AMS.ico")
    subject.title("Subject...")
    subject.geometry("580x320")
    subject.resizable(0, 0)
    subject.configure(background="black")
    # subject_logo = Image.open("UI_Image/0004.png")
    # subject_logo = subject_logo.resize((50, 47), Image.ANTIALIAS)
    # subject_logo1 = ImageTk.PhotoImage(subject_logo)
    titl = tk.Label(subject, bg="black", relief=RIDGE, bd=10,
font=("arial", 30))
```

```
    titl.pack(fill=X)
    # l1 = tk.Label(subject, image=subject_logo1, bg="black",)
    # l1.place(x=100, y=10)
    titl = tk.Label(
        subject,
        text="Enter the Subject Name",
        bg="black",
        fg="green",
        font=("arial", 25),
    )
    titl.place(x=160, y=12)
    Notifica = tk.Label(
        subject,
        text="Attendance filled Successfully",
        bg="yellow",
        fg="black",
        width=33,
        height=2,
        font=("times", 15, "bold"),
    )

    def Attf():
        sub = tx.get()
        if sub == "":
            t = "Please enter the subject name!!!"
            text_to_speech(t)
        else:
            os.startfile(
                f"Attendance\\{sub}"
            )

    attf = tk.Button(
        subject,
        text="Check Sheets",
        command=Attf,
        bd=7,
        font=("times new roman", 15),
        bg="black",
        fg="yellow",
        height=2,
        width=10,
        relief=RIDGE,
    )
    attf.place(x=360, y=170)

    sub = tk.Label(
        subject,
        text="Enter Subject",
        width=10,
        height=2,
        bg="black",
        fg="yellow",
        bd=5,
        relief=RIDGE,
        font=("times new roman", 15),
    )
    sub.place(x=50, y=100)

    tx = tk.Entry(
        subject,
        width=15,
        bd=5,
        bg="black",
        fg="yellow",
        relief=RIDGE,
        font=("times", 30, "bold"),
    )
    tx.place(x=190, y=100)

    fill_a = tk.Button(
        subject,
        text="Fill Attendance",
        command=FillAttendance,
        bd=7,
        font=("times new roman", 15),
        bg="black",
        fg="yellow",
        height=2,
        width=12,
        relief=RIDGE,
    )
    fill_a.place(x=195, y=170)
    subject.mainloop()
```

3.show_attendance.py

```
import pandas as pd
from glob import glob
import os
import tkinter
import csv
import tkinter as tk
from tkinter import *

def subjectchoose(text_to_speech):
    def calculate_attendance():
        Subject = tx.get()
        if Subject=="":
            t='Please enter the subject name.'
            text_to_speech(t)

        filenames = glob(
            f"Attendance\\{Subject}\\{Subject}*.csv"
        )
        df = [pd.read_csv(f) for f in filenames]
        newdf = df[0]
        for i in range(1, len(df)):
            newdf = newdf.merge(df[i], how="outer")
        newdf.fillna(0, inplace=True)
        newdf["Attendance"] = 0
        for i in range(len(newdf)):
            newdf["Attendance"].iloc[i] = str(int(round(newdf.iloc[i, 2:-
1].mean() * 100)))+'%'
            #newdf.sort_values(by=['Enrollment'],inplace=True)
        newdf.to_csv(f"Attendance\\{Subject}\\attendance.csv",
index=False)

        root = tkinter.Tk()
        root.title("Attendance of "+Subject)
        root.configure(background="black")
        cs = f"Attendance\\{Subject}\\attendance.csv"
        with open(cs) as file:
            reader = csv.reader(file)
            r = 0

            for col in reader:
                c = 0
                for row in col:

                    label = tkinter.Label(
                        root,
                        width=10,
                        height=1,
                        fg="yellow",
                        font=("times", 15, " bold "),
                        bg="black",
                        text=row,
                        relief=tkinter.RIDGE,
                    )
                    label.grid(row=r, column=c)
                    c += 1
                r += 1
        root.mainloop()
        print(newdf)

    subject = Tk()
    # windo.iconbitmap("AMS.ico")
    subject.title("Subject...")
    subject.geometry("580x320")
    subject.resizable(0, 0)
    subject.configure(background="black")
    # subject_logo = Image.open("UI_Image/0004.png")
    # subject_logo = subject_logo.resize((50, 47), Image.ANTIALIAS)
    # subject_logo1 = ImageTk.PhotoImage(subject_logo)
    titl = tk.Label(subject, bg="black", relief=RIDGE, bd=10,
font=("arial", 30))
    titl.pack(fill=X)
    # l1 = tk.Label(subject, image=subject_logo1, bg="black",)
    # l1.place(x=100, y=10)
    titl = tk.Label(
        subject,
        text="Which Subject of Attendance?",
        bg="black",
        fg="green",
        font=("arial", 25),
    )
    titl.place(x=100, y=12)

    def Attf():
        sub = tx.get()
        if sub == "":
            t="Please enter the subject name!!!"
            text_to_speech(t)
        else:
            os.startfile(
            f"Attendance\\{sub}"
            )



    attf = tk.Button(
        subject,
        text="Check Sheets",
        command=Attf,
        bd=7,
        font=("times new roman", 15),
        bg="black",
        fg="yellow",
        height=2,
        width=10,
        relief=RIDGE,
    )
    attf.place(x=360, y=170)

    sub = tk.Label(
        subject,
        text="Enter Subject",
        width=10,
        height=2,
        bg="black",
        fg="yellow",
        bd=5,
        relief=RIDGE,
        font=("times new roman", 15),
    )
    sub.place(x=50, y=100)

    tx = tk.Entry(
        subject,
        width=15,
        bd=5,
        bg="black",
        fg="yellow",
        relief=RIDGE,
        font=("times", 30, "bold"),
    )
    tx.place(x=190, y=100)

    fill_a = tk.Button(
        subject,
        text="View Attendance",
        command=calculate_attendance,
        bd=7,
        font=("times new roman", 15),
        bg="black",
        fg="yellow",
        height=2,
        width=12,
        relief=RIDGE,
    )
    fill_a.place(x=195, y=170)
    subject.mainloop()
```

4.takeImage.py

```
import csv
import os, cv2
import numpy as np
import pandas as pd
import datetime
import time




# take Image of user
def TakeImage(l1, l2, haarcasecade_path, trainimage_path, message,
err_screen,text_to_speech):
    if (l1 == "") and (l2==""):
        t='Please Enter the your Enrollment Number and Name.'
        text_to_speech(t)
    elif l1=='':
        t='Please Enter the your Enrollment Number.'
        text_to_speech(t)
    elif l2 == "":
        t='Please Enter the your Name.'
        text_to_speech(t)
    else:
        try:
            cam = cv2.VideoCapture(0)
            detector = cv2.CascadeClassifier(haarcasecade_path)
            Enrollment = l1
            Name = l2
            sampleNum = 0
            directory = Enrollment + "_" + Name
            path = os.path.join(trainimage_path, directory)
            os.mkdir(path)
            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)
                    sampleNum = sampleNum + 1
                    cv2.imwrite(
                        f"{path}\ "
                        + Name
                        + "_"
                        + Enrollment
                        + "_"
                        + str(sampleNum)
                        + ".jpg",
                        gray[y : y + h, x : x + w],
                    )
                    cv2.imshow("Frame", img)
                if cv2.waitKey(1) & 0xFF == ord("q"):
                    break
                elif sampleNum > 50:
                    break
            cam.release()
            cv2.destroyAllWindows()
            row = [Enrollment, Name]
            with open(
                "StudentDetails/studentdetails.csv",
                "a+",
            ) as csvFile:
                writer = csv.writer(csvFile, delimiter=",")
                writer.writerow(row)
                csvFile.close()
            res = "Images Saved for ER No:" + Enrollment + " Name:" + Name
            message.configure(text=res)
            text_to_speech(res)
        except FileExistsError as F:
            F = "Student Data already exists"
            text_to_speech(F)
```

5.takemanually.py

```
import tkinter as tk
from tkinter import Message, Text
import os, cv2
import shutil
import csv
import numpy as np
from PIL import ImageTk, Image
import pandas as pd
import datetime
import time
import tkinter.ttk as tkk
import tkinter.font as font

ts = time.time()
Date = datetime.datetime.fromtimestamp(ts).strftime("%Y_%m_%d")
timeStamp = datetime.datetime.fromtimestamp(ts).strftime("%H:%M:%S")
Time = datetime.datetime.fromtimestamp(ts).strftime("%H:%M:%S")
Hour, Minute, Second = timeStamp.split(":")
d = {}
index = 0
####GUI for manually fill attendance
def manually_fill():
    global sb
    sb = tk.Tk()
    sb.iconbitmap("AMS.ico")
    sb.title("Enter subject name...")
    sb.geometry("580x320")
    sb.configure(background="snow")

    def err_screen_for_subject():
        def ec_delete():
            ec.destroy()

        global ec
        ec = tk.Tk()
        ec.geometry("300x100")
        ec.iconbitmap("AMS.ico")
        ec.title("Warning!!")
        ec.configure(background="snow")
        tk.Label(
            ec,
            text="Please enter subject name!!!",
            fg="red",
            bg="white",
            font=("times", 16, " bold "),
        ).pack()
        tk.Button(
            ec,
            text="OK",
            command=ec_delete,
            fg="black",
            bg="lawn green",
            width=9,
            height=1,
            activebackground="Red",
            font=("times", 15, " bold "),
        ).place(x=90, y=50)

    def fill_attendance():

        ##Create table for Attendance
        global subb
        subb = SUB_ENTRY.get()

        if subb == "":
            err_screen_for_subject()
        else:
            sb.destroy()
            MFW = tk.Tk()
            MFW.iconbitmap("AMS.ico")
            MFW.title("Manually attendance of " + str(subb))
            MFW.geometry("880x470")
            MFW.configure(background="snow")

            def del_errsc2():
                errsc2.destroy()

            def err_screen1():
                global errsc2
                errsc2 = tk.Tk()
                errsc2.geometry("330x100")
                errsc2.iconbitmap("AMS.ico")
                errsc2.title("Warning!!")
                errsc2.configure(background="snow")
                tk.Label(
                    errsc2,
                    text="Please enter Student & Enrollment!!!",
                    fg="red",
                    bg="white",
                    font=("times", 16, " bold "),
                ).pack()
                tk.Button(
                    errsc2,
                    text="OK",
                    command=del_errsc2,
                    fg="black",
                    bg="lawn green",
                    width=9,
                    height=1,
                    activebackground="Red",
                    font=("times", 15, " bold "),
                ).place(x=90, y=50)

            def testVal(inStr, acttyp):
                if acttyp == "1":  # insert
                    if not inStr.isdigit():
                        return False
                return True

            ENR = tk.Label(
                MFW,
                text="Enter Enrollment",
                width=15,
                height=2,
                fg="white",
                bg="blue2",
                font=("times", 15, " bold "),
            )
            ENR.place(x=30, y=100)

            STU_NAME = tk.Label(
                MFW,
                text="Enter Student name",
                width=15,
                height=2,
                fg="white",
                bg="blue2",
                font=("times", 15, " bold "),
            )
            STU_NAME.place(x=30, y=200)

            global ENR_ENTRY
            ENR_ENTRY = tk.Entry(
                MFW,
                width=20,
                validate="key",
                bg="yellow",
                fg="red",
                font=("times", 23, " bold "),
            )
            ENR_ENTRY["validatecommand"] = (ENR_ENTRY.register(testVal),
"%P", "%d")
            ENR_ENTRY.place(x=290, y=105)

            def remove_enr():
                ENR_ENTRY.delete(first=0, last=22)

            STUDENT_ENTRY = tk.Entry(
                MFW, width=20, bg="yellow", fg="red", font=("times", 23, "
bold ")
            )
            STUDENT_ENTRY.place(x=290, y=205)

            def remove_student():
                STUDENT_ENTRY.delete(first=0, last=22)
            ####get important variable

            def enter_data_DB():
                global index
                global d
                ENROLLMENT = ENR_ENTRY.get()
                STUDENT = STUDENT_ENTRY.get()
                if ENROLLMENT == "":
                    err_screen1()
                elif STUDENT == "":
                    err_screen1()
                else:
                    if index == 0:
                        d = {
                            index: {"Enrollment": ENROLLMENT, "Name":
STUDENT, Date: 1}
                        }
                        index += 1
                        ENR_ENTRY.delete(0, "end")
                        STUDENT_ENTRY.delete(0, "end")
                    else:
                        d[index] = {"Enrollment": ENROLLMENT, "Name":
STUDENT, Date: 1}
                        index += 1
                        ENR_ENTRY.delete(0, "end")
                        STUDENT_ENTRY.delete(0, "end")
                    # TODO implement CSV code
                print(d)

            def create_csv():
                df = pd.DataFrame(d)
                csv_name = (
                    "Attendance(Manually)/"
                    + subb
                    + "_"
                    + Date
                    + "_"
                    + Hour
                    + "-"
                    + Minute
                    + "-"
                    + Second
                    + ".csv"
                )
                df.to_csv(csv_name)
                O = "CSV created Successfully"
                Notifi.configure(
                    text=O,
                    bg="Green",
                    fg="white",
                    width=33,
                    font=("times", 19, "bold"),
                )
                Notifi.place(x=180, y=380)
                """import csv
                import tkinter

                root = tkinter.Tk()
                root.title("Attendance of " + subb)
                root.configure(background="snow")
                with open(csv_name, newline="") as file:
                    reader = csv.reader(file)
                    r = 0

                    for col in reader:
                        c = 0
                        for row in col:
                            # i've added some styling
                            label = tkinter.Label(
                                root,
                                width=13,
                                height=1,
                                fg="black",
                                font=("times", 13, " bold "),
                                bg="lawn green",
                                text=row,
                                relief=tkinter.RIDGE,
                            )
                            label.grid(row=r, column=c)
                            c += 1
                        r += 1
                root.mainloop()"""

            Notifi = tk.Label(
                MFW,
                text="CSV created Successfully",
                bg="Green",
                fg="white",
                width=33,
                height=2,
                font=("times", 19, "bold"),
            )

            c1ear_enroll = tk.Button(
                MFW,
                text="Clear",
                command=remove_enr,
                fg="black",
                bg="deep pink",
                width=10,
                height=1,
                activebackground="Red",
                font=("times", 15, " bold "),
            )
            c1ear_enroll.place(x=690, y=100)

            c1ear_student = tk.Button(
                MFW,
                text="Clear",
                command=remove_student,
                fg="black",
                bg="deep pink",
                width=10,
                height=1,
                activebackground="Red",
                font=("times", 15, " bold "),
            )
            c1ear_student.place(x=690, y=200)

            DATA_SUB = tk.Button(
                MFW,
                text="Enter Data",
                command=enter_data_DB,
                fg="black",
                bg="lime green",
                width=20,
                height=2,
                activebackground="Red",
                font=("times", 15, " bold "),
            )
            DATA_SUB.place(x=170, y=300)

            MAKE_CSV = tk.Button(
                MFW,
                text="Convert to CSV",
                command=create_csv,
                fg="black",
                bg="red",
                width=20,
                height=2,
                activebackground="Red",
                font=("times", 15, " bold "),
            )
            MAKE_CSV.place(x=570, y=300)
            # TODO remove check sheet
            def attf():
                import subprocess

                subprocess.Popen(
                    r'explorer
/select,"C:/Users/patel/OneDrive/Documents/E/FBAS/Attendance(Manually)"'
                )

            attf = tk.Button(
                MFW,
                text="Check Sheets",
                command=attf,
                fg="black",
                bg="lawn green",
                width=12,
                height=1,
                activebackground="Red",
                font=("times", 14, " bold "),
            )
            attf.place(x=730, y=410)

            MFW.mainloop()

    SUB = tk.Label(
        sb,
        text="Enter Subject",
        width=15,
        height=2,
        fg="white",
        bg="blue2",
        font=("times", 15, " bold "),
    )
    SUB.place(x=30, y=100)

    global SUB_ENTRY

    SUB_ENTRY = tk.Entry(
        sb, width=20, bg="yellow", fg="red", font=("times", 23, " bold ")
    )
    SUB_ENTRY.place(x=250, y=105)

    fill_manual_attendance = tk.Button(
        sb,
        text="Fill Attendance",
        command=fill_attendance,
        fg="white",
        bg="deep pink",
        width=20,
        height=2,
        activebackground="Red",
        font=("times", 15, " bold "),
    )
    fill_manual_attendance.place(x=250, y=160)
    sb.mainloop()
```

6.test.py

```
import requests
import cv2
import numpy as np

url = "http://192.168.0.6:8080/shot.jpg"

while True:
    cam = requests.get(url)
    imgNp = np.array(bytearray(cam.content), dtype=np.uint8)
    img = cv2.imdecode(imgNp, -1)
    cv2.imshow("cam", img)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break
```

7.trainImage.py

```
import csv
import os, cv2
import numpy as np
import pandas as pd
import datetime
import time
from PIL import ImageTk, Image



# Train Image
def TrainImage(haarcasecade_path, trainimage_path, trainimagelabel_path,
message,text_to_speech):
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    detector = cv2.CascadeClassifier(haarcasecade_path)
    faces, Id = getImagesAndLables(trainimage_path)
    recognizer.train(faces, np.array(Id))
    recognizer.save(trainimagelabel_path)
    res = "Image Trained successfully"  # +",".join(str(f) for f in Id)
    message.configure(text=res)
    text_to_speech(res)



def getImagesAndLables(path):
    # imagePath = [os.path.join(path, f) for d in os.listdir(path) for f
in d]
    newdir = [os.path.join(path, d) for d in os.listdir(path)]
    imagePath = [
        os.path.join(newdir[i], f)
        for i in range(len(newdir))
        for f in os.listdir(newdir[i])
    ]
    faces = []
    Ids = []
    for imagePath in imagePath:
        pilImage = Image.open(imagePath).convert("L")
        imageNp = np.array(pilImage, "uint8")
        Id = int(os.path.split(imagePath)[-1].split("_")[1])
        faces.append(imageNp)
        Ids.append(Id)
    return faces, Ids
```