

# Preface

It brings me great pleasure to deliver the project on the "**FaceTrek Attendance - Face recognition Attendance System**" that was prepared honestly, on time, and with the utmost care.

Technology has assimilated into our daily lives in the quickly changing world of today. In the world of identity verification and attendance tracking, technological improvements have had a significant impact. Traditional means of manually recording attendance, such as paper registers and swipe cards, take a lot of time and are prone to fraud and inaccuracies. The Face Recognition Attendance System, a cutting-edge system that uses facial recognition technology to simplify attendance management, is developed in response to these difficulties.

An important advancement in the way organizations, institutions, and corporations track and manage attendance is the Face Recognition Attendance System. Modern facial recognition algorithms and high-resolution cameras are used by this cutting-edge system to identify and confirm people's identities quickly and accurately. This technology provides a safe, contact less, and effective way to track attendance by utilizing the special and inborn characteristics of the human face.

The goal of this project is to give readers a thorough understanding of the Face Recognition Attendance System, from its fundamental ideas and parts to how it works. We explore the algorithms, hardware, and software that power facial recognition technology, delving into its deep nuances. We also go over the privacy issues and ethical issues surrounding the usage of facial recognition technology, highlighting the significance of careful deployment and data security.

Throughout this project, we will embark on a journey to design, create, and deploy a working Face Recognition Attendance System. We will discuss the numerous phases involved in constructing such a system, including data gathering, model training, system integration, and user interface design. Readers will have a thorough blueprint for building their own face recognition attendance system or receive insightful knowledge of the technology underlying this creative solution by the conclusion of this assignment.

The Face Recognition Attendance System is an example of how technological improvements have the potential to improve efficiency, accuracy, and security in attendance management. This project provides a road map for individuals looking to use facial recognition technology to completely transform how attendance is tracked and handled.

We believe that this initiative will encourage and enable people, groups, and institutions to investigate the potential of facial recognition technology and its revolutionary effects on attendance monitoring.

This project has undergone meticulous scrutiny to ensure that it is flawless and beneficial in every way.

Shubham Bhalerao

# Acknowledgment

This Face Recognition Attendance System project's successful completion was made possible by the concerted efforts, leadership, and assistance of numerous people and organizations. I would like to sincerely thank everyone who contributed significantly to making this initiative a success.

First and foremost, I would like to express my sincere gratitude to Prof. Manoj Singh, my project supervisor, whose knowledge, support, and priceless insights significantly influenced the course of my work. Throughout the project, Prof. Manoj Singh gave me constant support and guided me through the challenges of facial recognition technology.

Additionally, I owe a debt of gratitude to the faculty, staff, and administration of SIES College of Arts, Science, and Commerce, whose expertise and resources proved to be crucial to my research and development endeavors. They have consistently served as an inspiration with their commitment to creating a supportive learning environment.

I would like to express my gratitude to the individuals who voluntarily agreed to take part in this study, since their participation was essential in gathering the information required for developing and validating my facial recognition model. The completion of this project would not have been feasible without their assistance.

The open-source community and the creators of the many libraries and technologies that I tapped into while creating my face recognition attendance system deserve recognition. I appreciate their dedication to advancing technology and their significant contributions.

Special thanks go out to my peers and friends for their emotional support and encouragement throughout the highs and lows of this endeavor. I was inspired to continue working by their zeal and faith in our job.

Finally, I want to thank my families from the bottom of my hearts for all their support, patience, and understanding. Their support and confidence in my talents have served as my inspiration.

I sincerely thank everyone who helped with this initiative, whether they were directly involved or not. Your assistance has been crucial in helping me realize my vision.

Mr. Shubham Bhalerao

SIES College of Arts, Science and Commerce

Date

## Contents

### Preliminary Investigation

1.1 ) Innovativeness .....	8
1.2) Benefits of FaceTrek Attendance .....	9
1.3) Existing System .....	10
1.4) Limitation of Existing System .....	10
1.5) Risk Management .....	10
1.5.1) Lack of consent	
1.5.2) Predatory Marketing	
1.5.3) Identity Fraud	
1.5.4) Security Issues	
1.6) Objectives .....	11
1.7) Proposed System .....	11
1.7.1) Data Collection and Preprocessing	
1.7.2) Face Detection	
1.7.3) Face Recognition Model	
1.7.4) Database Management	
1.7.5) User Registration Model	
1.7.6) User Interface	
1.8) Advantages of Proposed System .....	12
1.9) Tools and Models .....	12
1.10) Modules (Functional Requirements) .....	13
1.11) Non-Functional Modules .....	13
1.11.1) Usability	
1.11.2) Efficiency	
1.11.3) Reliability	

1.11.4) Maintainability	
1.12) Maintenance and Costs .....	14
1.13) System Requirements .....	15
1.13.1) Hardware Requirements (Minimum)	
1.13.2) Software Requirements	
1.13.3) Software Requirement for Development of FaceTrek Attendance	
1.14) Feasibility Study .....	15
1.14.1) Project Scope and Objectives	
1.14.2) Market Analysis	
1.14.3) Technical Feasibility	
1.14.4) Technical Requirements	
1.14.5) Resource Availability	
1.15) Stake Holders .....	16
1.16) Events .....	16
1.17) Gantt Chart .....	18

## System Analysis

2.1) Fact-Finding technique .....	19
2.1.1) Interview - Based	
2.1.2) Documentation and Record View	
2.2) Event Table .....	19
2.3) ER Diagram .....	23
2.4) Class Diagram .....	23
2.5) Use-case Diagram .....	24
2.6) Sequence Diagram .....	26
2.7) Activity Diagram .....	28
2.8) System Design .....	30
2.9) Conclusion .....	31

<b>System Implementation (Code)</b> .....	32
<b>System Screenshots</b> .....	38
<b>Scope for Future Enhancement</b> .....	42
<b>Bibliography</b> .....	43

# Preliminary Investigation

## Innovativeness

With no longer requiring manual tracking of attendance, the suggested method will reduce paperwork. The overall amount of time required to record attendance will be reduced by the new system.

To ensure the accuracy of the attendance data, the new system will collect individual attendance using facial recognition technology.

The prior system needs to be evolved to increase productivity, ensure data accuracy, and allow access to the information for those legitimate purposes.

Without human mistakes, facial recognition software can accurately manage time and attendance. With a facial recognition time monitoring system, you won't ever have to be concerned about time fraud or "buddy punching."

Employees may log in and out of the building faster with facial recognition because they don't have to contact the system's surface. This reduces the transmission of illnesses while also saving time.

The lack of authorization :- Facial recognition software only requires a clear image of the subject's face to work therefore data can be easily acquired in public locations.

Predatory marketing :- Some businesses may utilize software that analyzes facial expressions to prey on susceptible clients.

Identification fraud :- Criminals who have amassed sufficient personal data about you may engage in identity fraud.

In any setting, the system will mark and record attendance.

Since this system is automated, attendance will be recorded in accordance with that, greatly boosting accuracy, and a report on attendance will then be generated.

These are the key components of this technology:- 1) Face recognition 2) Face identification.

1) Face detection: A computer technology that recognizes human faces in digital photos, face detection is utilized in a range of applications.

2) Face recognition: A facial recognition system is a computer program that can recognize or authenticate a person from a digital image.

Steps to use FaceTrek Attendance on computer:

1. The representative or user of a specific department must first sign up for the system.
2. After that, he or she must enter valid credentials to access the system.
3. After proper validation, he or she will add the student, teacher, and staff data to the system.
4. After that, he or she will input the samples of their faces into the system.
5. After that, he or she will use the sample image data to train the facial recognition model.
6. The representative or user will then be prepared to take attendance using face recognition on each day of class or school.

Uses of FaceTrek Attendance:

- A proper confirmation message is given to every person (present or absent) by mail and standard SMS texts.
- Attendance is managed without human intervention.
- Reports or attendance records are automatically generated in CSV files.
- Create an invoice for each task that is successfully finished.
- Searching is permitted on all individual data (students, teachers, employees, and users from the admin side) via a straightforward, user-friendly GUI.
- Safeguarding attendance records with backups.

## **Benefits of FaceTrek Attendance:**

- Less human mistake.
- Ease of maintenance.
- Time savings.
- Backup of attendance records.
- Invoice for successful operations.
- Confirmation messages to all users.
- Accelerate the work.
- No possibility of proxy.
- Attendance based on given or allotted period.

## Existing System

- Students must manually enter data into the current system.
- Registers written by hand will be used to track attendance here.
- Keeping track of the user's record will be a laborious task.
- This requires more human effort.
- Since records are kept in manually written registers, retrieving information is difficult.
- This application needs accurate data entered the appropriate fields.
- If the incorrect inputs are made, the application might not function properly, making it challenging for the user to utilize.

## Limitation of Existing System:

- It takes more time.
- Involves a lot of paperwork.
- Increases the possibility of human mistakes.
- Improperly manages attendance records.
- Is less accurate.
- Increases the likelihood of proxy.
- Lack of a certification of attendance for the person.
- Data collision risks across classes.

## Risk Management

### Lack of consent:

- Facial recognition software only needs a clear picture of the subject's face to start collecting data, this can be done simply in public settings.

### Predatory marketing:

- Software that analyzes facial expressions may be used by some businesses to engage in predatory marketing, preying on weak clients.

### Identity fraud:

- Identity fraud is a crime that can be committed by criminals who have gathered enough personal information about you.

### Security Issues:

- Beware of attacks on the facial recognition attendance system using print photos, smartphone photos, replays, photo masks and 3D masks attack.

## Objectives

- A software program called an attendance management system was created to ensure that students, teachers, and staff show up to class each day.
- The staff members in charge of the department oversee recording the individuals' attendance in this instance.
- After recording each person's attendance and any absences, notification verification is carried out.
- When an unidentified person is recognized and for all successfully executed operations, an invoice is created.
- Attendance is done at predetermined intervals, such as for a person for only six or twelve months, and then when that person appears in front of the camera, my model will depict them as an unknown person.

## Proposed System

The proposed system is evolved to address the shortcomings of the current system. The proposed system in a Face Recognition Attendance System project normally comprises of several different parts and technologies that cooperate to produce the intended functionality. The following are the essential elements of the FaceTrek Attendance-Face Recognition Attendance System:

### Data Collection and Preprocessing:

- Image preprocessing techniques to improve image quality, such as noise reduction and face alignment; high-resolution webcams for obtaining facial images.

### Face Detection:

- A face identification method to find and extract faces from the collected photos (Haar cascades).

### Face Recognition Model:

- Using a varied dataset of facial photos of well-known people to train the model.

### Database Management:

- A database to keep track of registered users' attendance history and facial template data. I have utilized the MongoDB NoSQL database.

### User Registration Module:

- A module for adding new users, taking their photos, and putting their biometric templates in a folder.

### User Interface:

- An intuitive user interface that allows users and administrators to communicate with the system.

The Present system works by obtaining 100 image samples of an individual's face and then identifying it based on those 100 image samples. It then saves the attendance records in a CSV file and sends an email and SMS to the person as confirm their attendance.

The user must first register in the system, receive an OTP and a confirmation email via email, and then log in to store student, teacher, and staff data as well as to identify them and take attendance.

The system will be fully under the admins' control. In addition, he will save the CSV attendance file to another folder and automatically produce a new CSV file for the following day's attendance. He may change the details of every individual, mark attendance, and send emails and SMS to those who are absent. Additionally, each individual window has a search system for conducting searches.

Here, my project attempts to cut down on paperwork and save time to produce accurate attendance results. The system offers the best user interface, and improper behavior such as proxy is not tolerated. The suggested system can be used to produce effective reports. With minimal time spent, we can obtain data whenever it is needed.

## Advantages of Proposed System

- Simple to manage.
- User-friendly
- Sending SMS and emails of confirmation
- Saving attendance information as a CSV file
- Speed
- Invoice for each activity that is successfully completed.
- Appropriate input field validation.
- Attendance depending on the allowed or designated time.

## Tools and Model

- Python-based libraries such as tkinter, re, time, datetime, pymongo, os, shutil, csv, gtts, playsound, twilio.rest, tkcalendar, PIL, cv2, numpy.
- Agile Model. It is a flexible and iterative approach that allows for ongoing adaption to change requirements and priorities.

## Modules (Functional Requirements):

- Admin Module.
- User Registration and Enrollment Module.
- Face Recognition Module.
- Face Detection Module.
- Attendance Tracking Module.
- User Interface Module.
- Camera Interface Module.
- Notification Module.
- Database Module.
- Data Storage Module.
- Machine Learning Model Training Module.

## Non – Functional Requirements:

Non-functional requirements are limitations that must be followed when developing an application.

### Usability:

- The clients or stakeholders will use the program or product that I am producing.

### Efficiency:

- Time complexity is decreased by my software's ability to complete tasks more quickly, such as issuing notifications and recording attendance.

### Reliability:

- The program I'm working on is intended to deliver a range of services as anticipated by the stakeholders.

### Maintainability:

- The software I'm creating will have high performance features, such as manual data updates without losing previously stored data.

## Maintenance and Estimated Costs

Costing and other fees are added in accordance with the company's regulations. Pricing can be adjusted to meet new needs.

- Price (per month salary) : Rs. 60,000
- 6 months' salary : Rs. 3,60,000

LIST	COST
Hardware System Requirement	1,90,000
Software System Requirement	2,70,000
Designing Charge	4,50,000
Functionality Charge	5,60,000
Designer & Developer Team Salary	3,60,000 (for six months)
Company Profit	1,40,000
Maintenance Charges	50,000 (per month)
Miscellaneous Charges	70,000
Tax (10%)	93,000
<b>Total Amount</b>	<b>2,083,000</b>

## System Requirements

### Hardware Requirements (Minimum) :

Processor:- Intel i3 Processor and above.

Primary Memory:- 4GB RAM (Random Access Memory) and above.

Storage:- 8GB hard disk and above.

### Software Requirements:

Operating System:- Microsoft Windows (7,8, and 10), Linux, and Mac OS.

Software:- Python IDLE.

### Software Requirement for Development of FaceTrek Attendance:

Front-end:- Python (tkinter)

Package:- tkinter, re, time, datetime, pymongo, os, shutil, csv, gtts, playsound, twilio.rest, tkcalendar, PIL, cv2, numpy.

Back-end:- Python (pymongo).

Database:- MongoDB.

Software:- Python IDLE, MongoDB Compass.

## Feasibility Study

Once the issue is properly identified, a feasibility study is carried out. The goal is to decide on a solution to an issue fast and cheaply. Details of operation and management are included.

The system has undergone tests to determine its viability in the following areas:

### Project Scope and Objectives:

- The major goal of this automated attendance system for face detection and recognition is to offer face recognition in a real-time environment so that employees, teachers, and students can observe and mark their daily attendance so that their presence can be kept track of.

### Market Analysis:

- The schools/colleges that are having trouble managing attendance on paperwork are the system's target market.
- I investigated the current attendance face recognition systems and tried to outperform them.

**Technical Feasibility:**

- This technique will be highly beneficial for recording attendance using facial data, reducing the likelihood of proxy scenarios.

**Technical Requirements:**

- The only technical requirements are a competent laptop or computer with fair internet connectivity and a webcam.

**Resource Availability:**

- The individual or user who will utilize the system should have a basic understanding of computers and operating systems.
- If the computer or laptop lacks speakers or a camera, additional external speakers and webcams can be required.

## Stakeholders

- Admin
- Users
- Software Developer
- Students (Using via Face Detection)
- Teachers (Using via Face Detection)
- Staffs (Using via Face Detection)
- Unknown Persons (Using via Face Detection)

## Events

### 1. Admin

- Login
- Logout
- Add user information.
- Update users' data
- Eliminate users' data.
- Refresh admin dashboard
- Check out Students data.
- Access Staffs data
- Access Teachers data
- Full system access
- Run a user data search.
- Set all input fields to their default values.
- Notifying absent people via SMS and email with a confirmation message.
- System reset for attendance the following day.

2. Users

- Self-register
- Verify with OTP
- User Login
- User Logout
- Add individuals data.
- Update individuals data
- Delete individuals data.
- Refresh each panel or window separately.
- Conduct individual data searches.
- Restore all input fields to their initial settings.
- Take/update individual photo samples.
- Develop the facial recognition algorithm.
- Record attendance of people
- Export the CSV file for attendance.
- Import the CSV file for attendance.
- Update the CSV file for attendance.
- Looking at samples of individuals photos

3. Software Developer

- Monitoring and support
- Updates and maintenance
- Data Management
- User feedback

4. Students

- Appear in front of the camera and record your attendance.

5. Teachers

- Appear in front of the camera and record your attendance.

6. Staffs

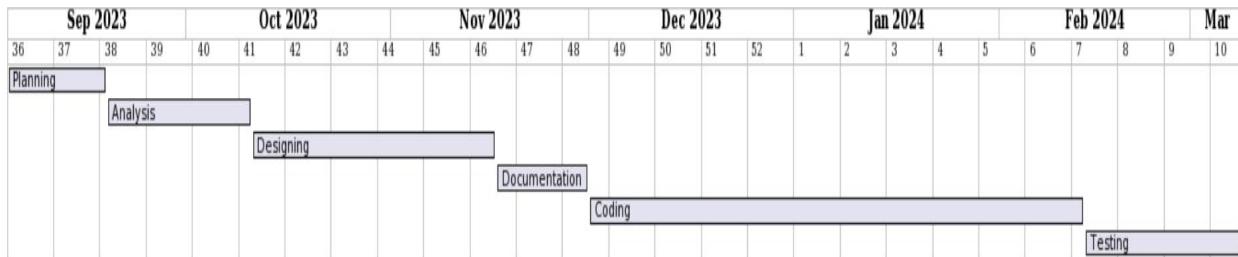
- Appear in front of the camera and record your attendance.

7. Unknown persons

- Appear in front of the camera.

## Gantt Chart

Planned



Phase Title	Start Date	End Date	No. Of Days	Signature
Planning	04/09/2023	18/09/2023	15	
Analysis	19/09/2023	10/10/2023	22	
Designing	11/10/2023	16/11/2023	37	
Documentation	17/11/2023	30/11/2023	14	
Coding	01/12/2023	13/02/2024	75	
Testing	14/02/2024	08/03/2024	24	

# System Analysis

## Fact-Finding Technique

**Interview - Based** :- The interviews I conducted with the Head of Department (HOD), Principal, Teachers, Staff, Students, and other stakeholders gave me a great understanding of how important it is for them to have an attendance system that uses facial recognition and how the various attendance procedures work. The information I acquired greatly aided them in meeting their needs and simplifying their work.

**Documentation and Record View** :- This kind of review assists us in learning about the procedures used while taking attendance, such as how the college as a whole manages attendance, how courses are handled when faculty members are absent, and what challenges they encounter.

## Event Table

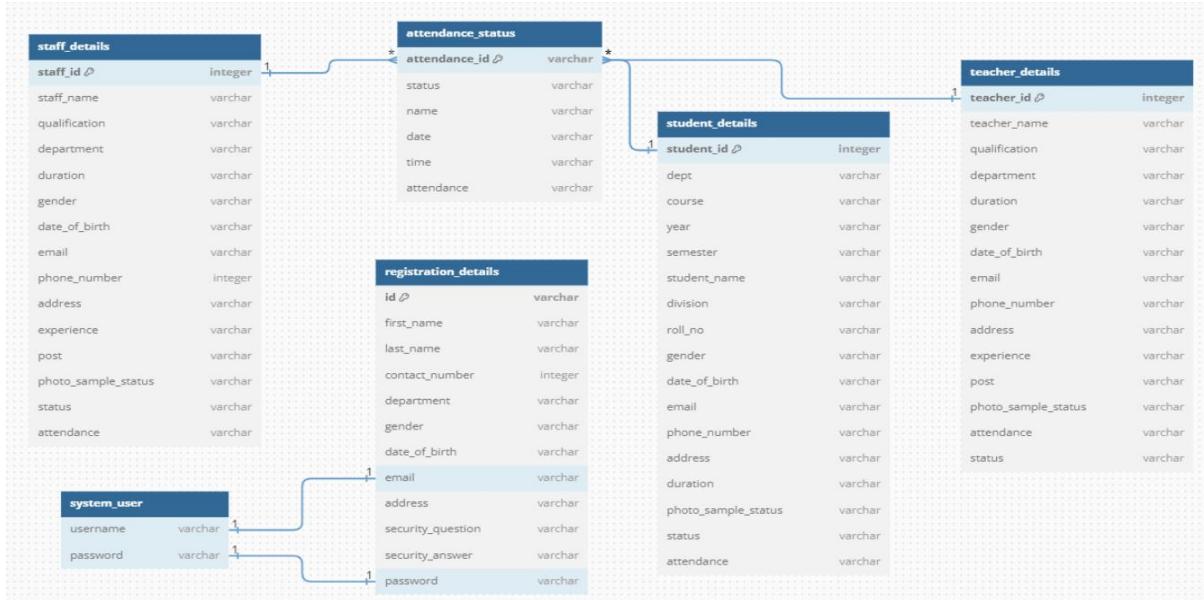
Sr no.	Event	Trigger	Source	Activity	Response	Destination
1.	Admin Login	Admin logs in	Admin	Input username and passwords	Admin dashboard or Error	Admin
2.	Admin Logout	Admin logs out	Admin	Click logout button	Login window	Admin
3.	Add user information	Add user information details	Admin	Enter User Details	User Added or Error	Admin, User
4.	Update user's data	Update user information details	Admin	Enter User Updated Details	Details Updated or Error	Admin, User
5.	Eliminate user's data	Delete user information details	Admin	Enter users date of birth	User deleted or Error	Admin, User
6.	Refresh admin dashboard	Admin dashboard gets refreshed	Admin	Click the refresh button	Dashboard refreshed	Admin
7.	Check out students' data	Viewing student's full database details	Admin	Click the student data button	Student's data window	Admin
8.	Access teachers' data	Viewing teacher's full database details	Admin	Click the teacher's data button	Teacher's data window	Admin

9.	Access staffs' data	Viewing staff's full database details	Admin	Click the staff data button	Staff's data window	Admin
10.	Full system access	Access to the FaceTrek Attendance system	Admin	Click the home button	FaceTrek Attendance System	Admin
11.	Run a user data search	Perform search on user data with parameters	Admin	Choose parameter from combo box input detail and click search button	Searched user data or No records found	Admin
12.	Set all input fields to their default values	Setting fields to their default values	Admin	Click the reset button	Fields reset to their default values	Admin
13.	Notifying absent people via SMS and email with a confirmation message.	Sending absent notifications	Admin	Click the Absent Message button	Notifications sends successfully or Error	Admin, teachers, students, staffs
14.	System reset for attendance the following day	Saves CSV to another folder, new CSV file is created, and the attendance fields of all individuals is set to absent.	Admin	Click Reset Data button	CSV file saved, a new CSV file is created, and all attendance fields of individuals is set to absent in database.	Admin
15.	Self – registered	Add details to register	User	Enter proper details, use strong password.	User is registered Successfully or Error occurred.	User
16.	Verify using OTP	Verify OTP to register	User	Verify OTP for completing registration process.	User is registered Successfully or Error occurred.	User
17.	User Login	User Logs in	User	Enter username and password	Successful Login or Error	User
18.	User Logout	User Logs out	User	Clicks exit button	Successful Logouts	User
19.	Add individuals data	Add staff, student, and teacher	User	Enter staff, student, and teacher details	Details added or Error	User

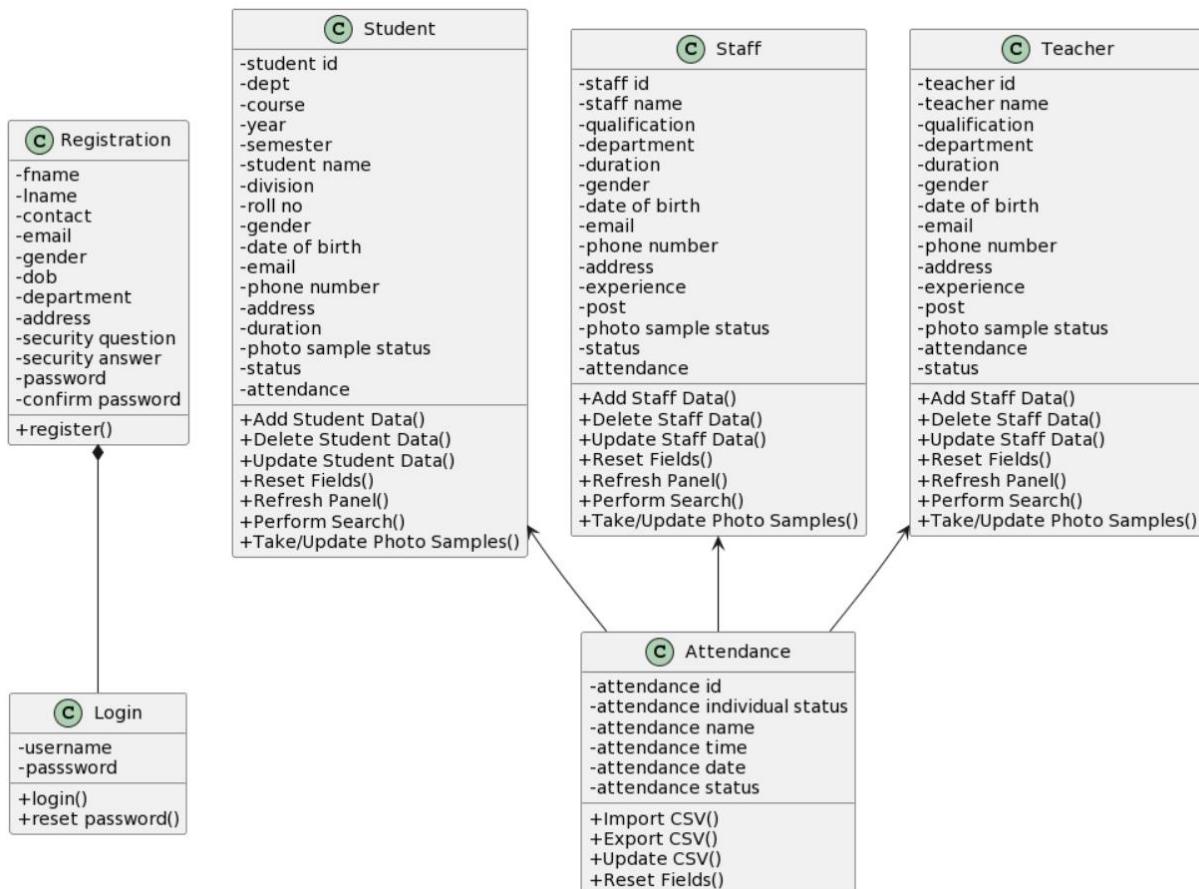
		information				
20.	Update individuals data	Update staff, student, and teacher information	User	Enter staff, student, and teacher details	Details updated or Error	User
21.	Delete individuals data	Delete staff, student, and teacher information	User	Id of individuals must be entered	Details deleted or Error	User
22.	Refresh each panel or window separately.	All individual windows/panel gets refreshed	User	Clicks refresh button	Individuals panel/window gets refreshed	User
23.	Conduct individual data searches.	Perform search on individual data with parameters	User	Choose parameter from combo box input detail and click search button	Searched user data or No records found	User
24.	Restore all input fields to their initial settings.	Setting fields to their default values	User	Click the reset button	Fields reset to their default values	User
25.	Take/update individual photo samples.	Take/Update photo samples	User	Click the take/update photo sample button	Photo samples taken/updated	User
26.	Develop the facial recognition algorithm.	Train the face recognition model	User	Click Train Data button	Model is trained with 100 photo samples of everyone	User
27.	Record attendance of individuals	Take Attendance	User	Click Recognize button	Attendance is taken or Error	User, students, teachers, staffs
28.	Export the CSV file for attendance	Export the updated CSV attendance file	User	Click Export button	File exported successfully or Error	User
29.	Import the CSV file for attendance	Import the attendance CSV file	User	Click Import button	File imported successfully or Error	User
30.	Update the CSV file for attendance	Update the imported CSV file	User	Update attendance details and click update button	Attendance CSV file updated successfully or Error	User

31.	Looking at samples of people's photos	Browse through 100 photo samples of individuals	User	Click Photos button	Sample Photos folder opens	User
32.	Monitoring and support	Monitors the system	Software Developer	Checks the system	System checking	Software Developer
33.	Updates and Maintenance	Provide updates to system	Software Developer	Check for updates	Updates done	Software Developer
34.	Data Management	Manage the data of stakeholders and database	Software Developer	Routine check	Data management	Software Developer
35.	User feedback	Take feedback from user	Software Developer	Take feedback via system or forms	Feedback taken	Software Developer
36.	Appear in front of the camera and record your attendance.	Give Attendance	Students	Comes in front of camera	Attendance done or Error	Students
37.	Appear in front of the camera and record your attendance.	Give Attendance	Teachers	Comes in front of camera	Attendance done or Error	Teachers
38.	Appear in front of the camera and record your attendance.	Give Attendance	Staffs	Comes in front of camera	Attendance done or Error	Staffs
39.	Appear in front of the camera.	-----	Unknown Persons	Comes in front of camera	Unknown face detected	Unknown Persons

## ER Diagram

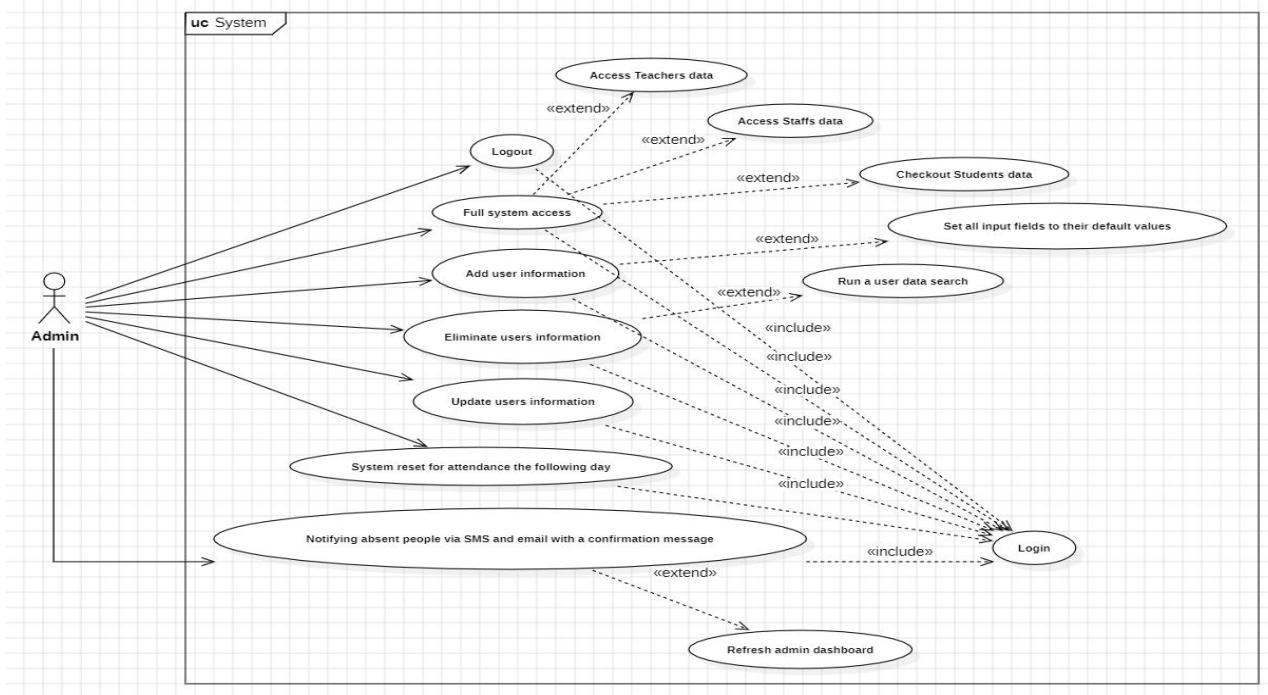


## Class Diagram

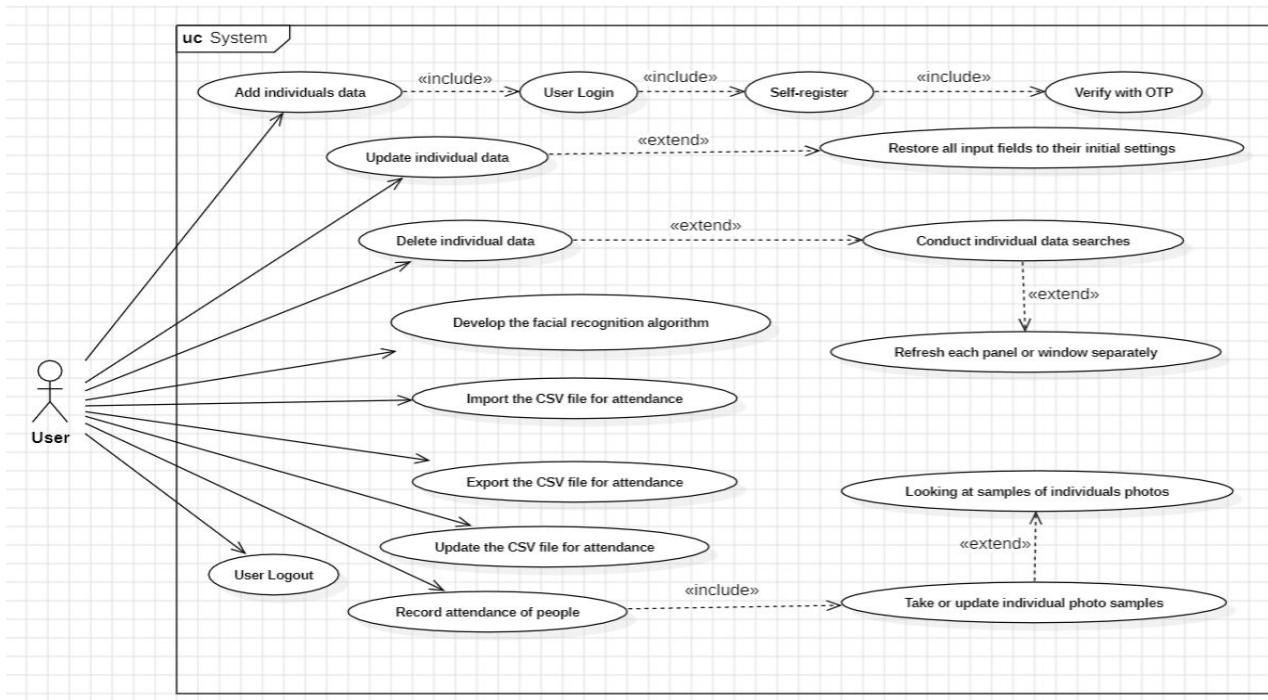


## Use Case Diagrams

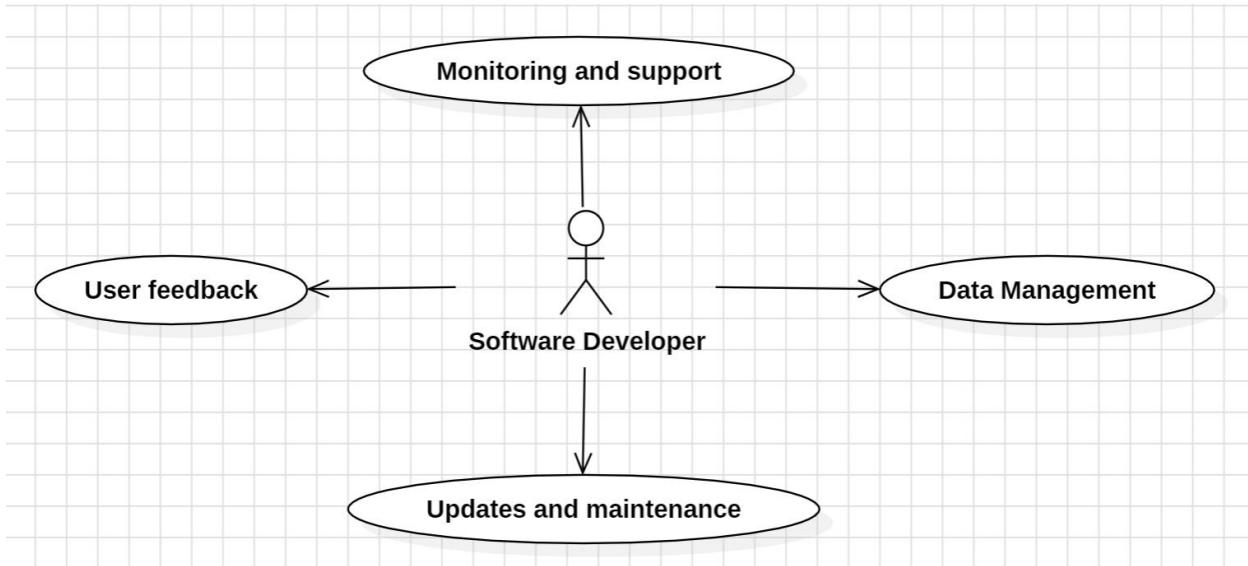
### 1) Admin



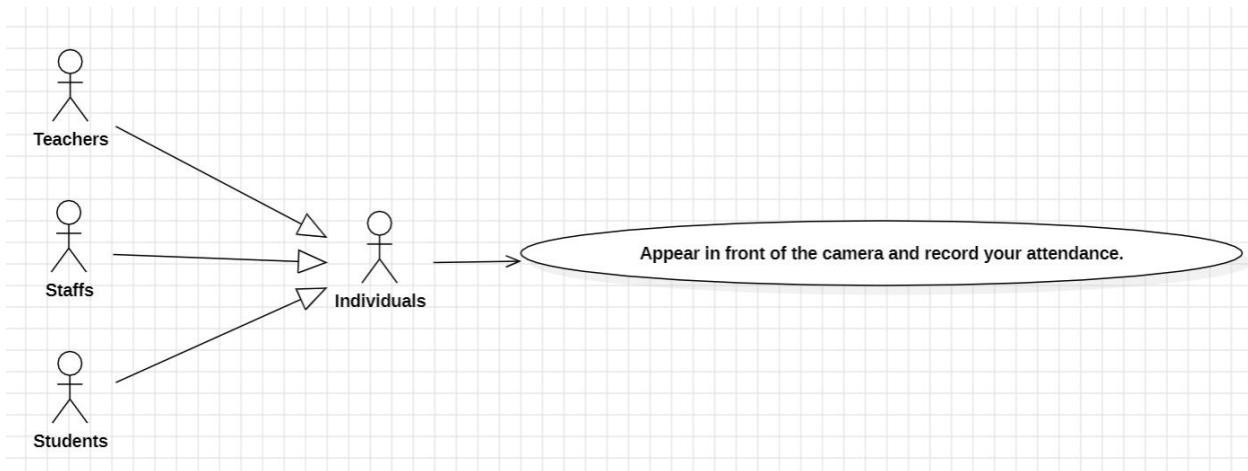
### 2) User



### 3) Software Developer

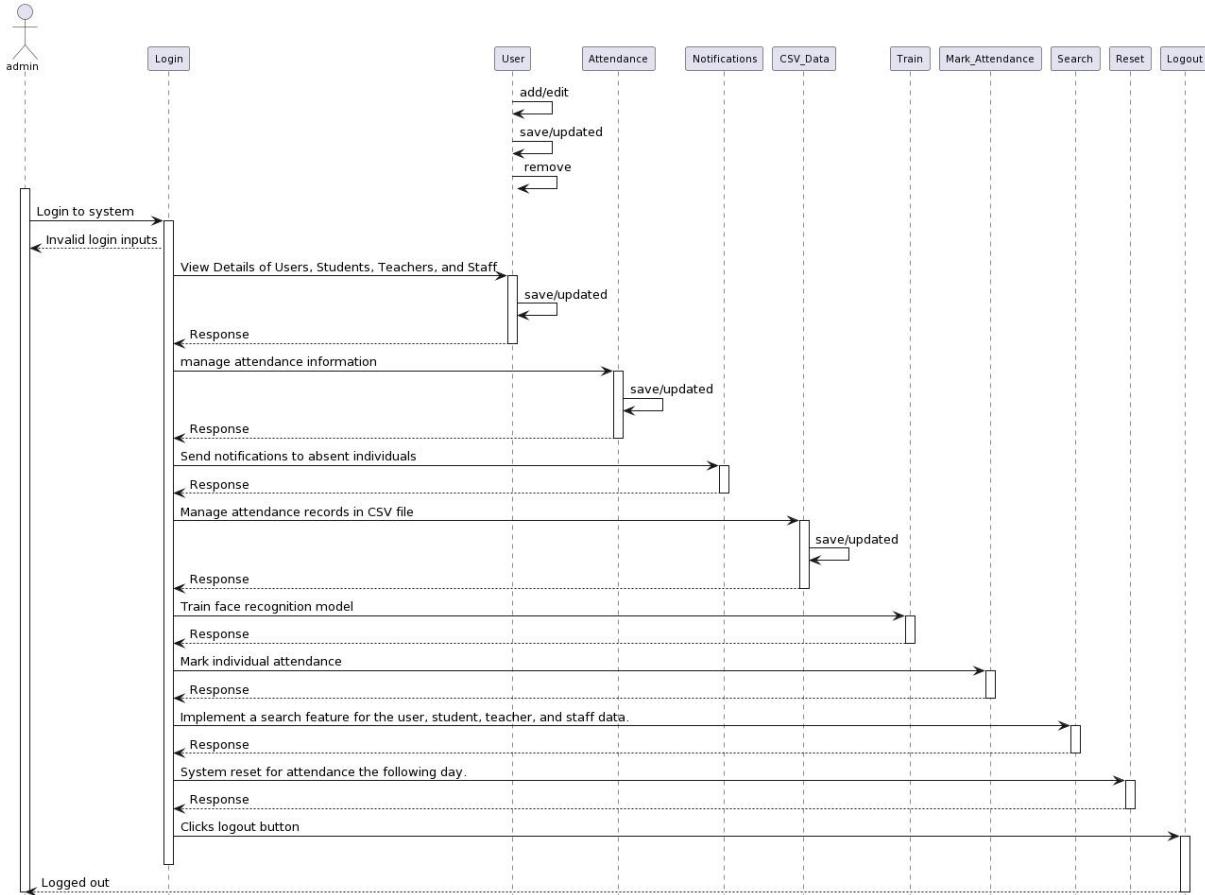


### 4) Individuals (Staffs, Teachers, Students)

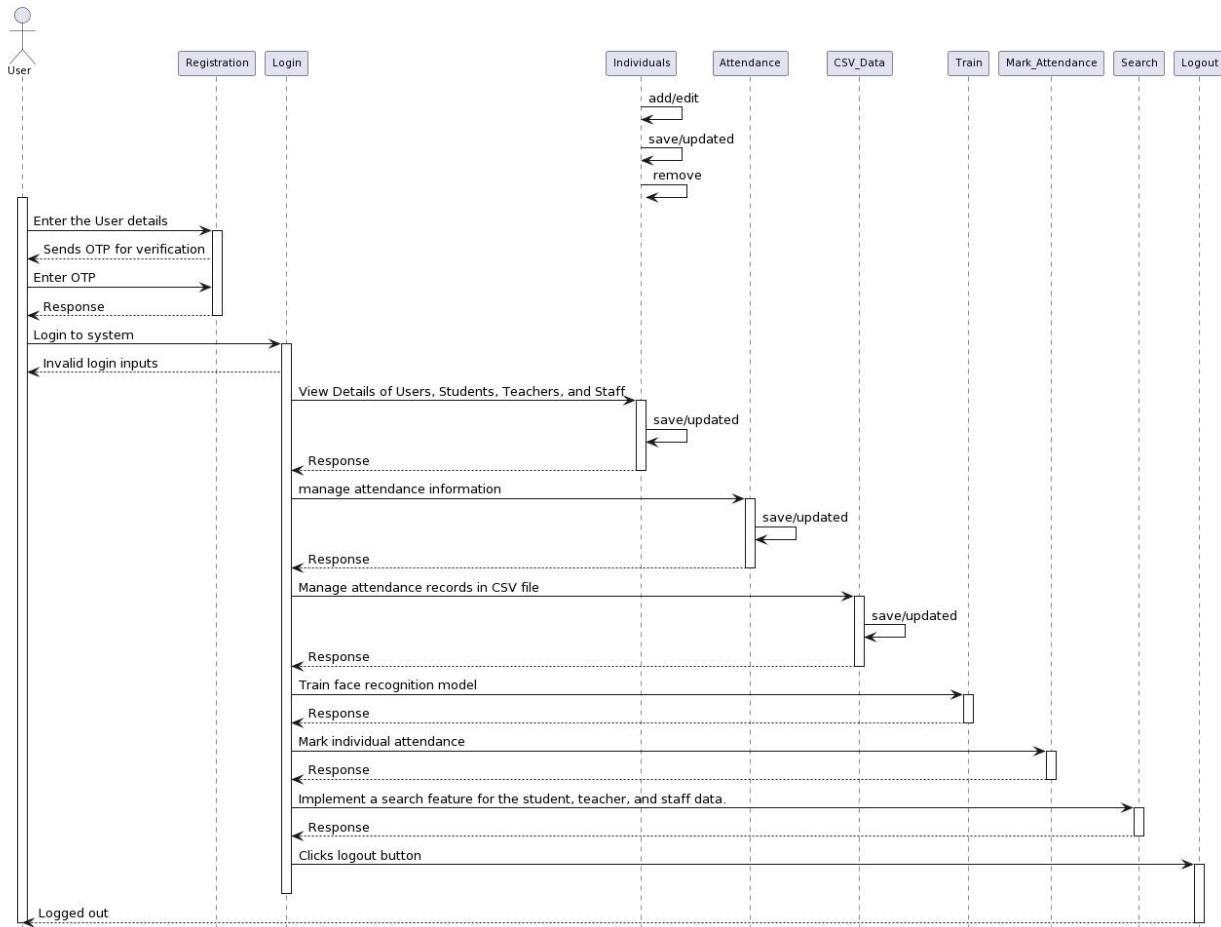


## Sequence Diagrams

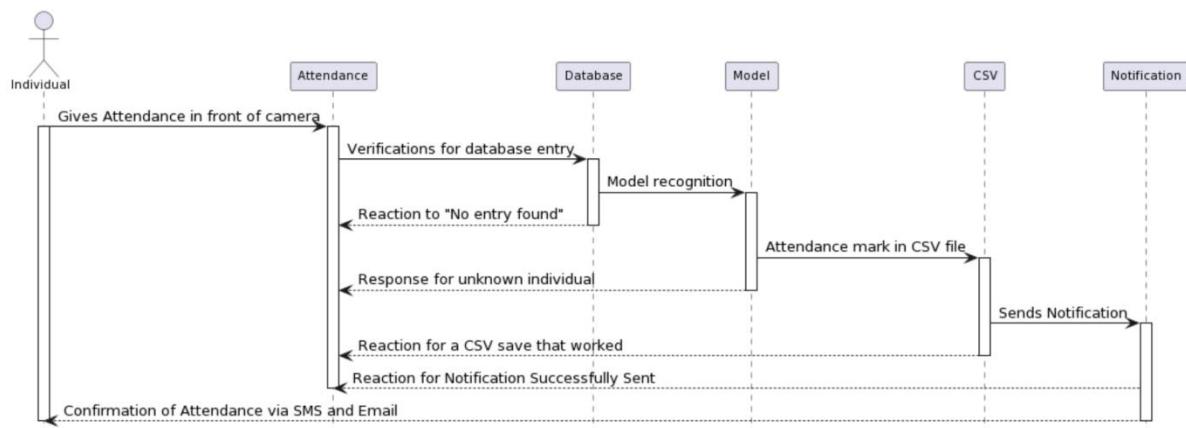
### 1) Admin



## 2) User

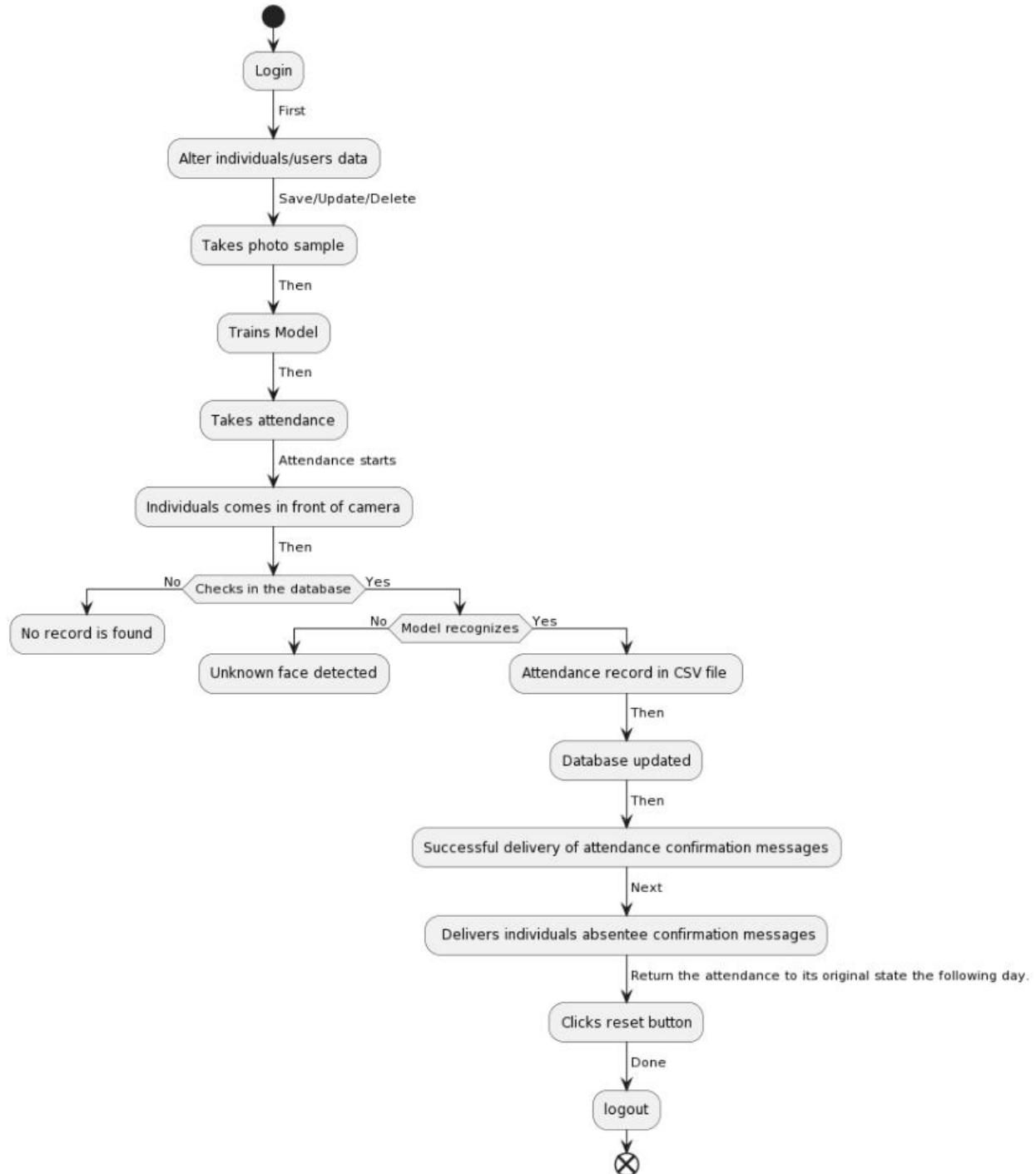


## 3) Individual

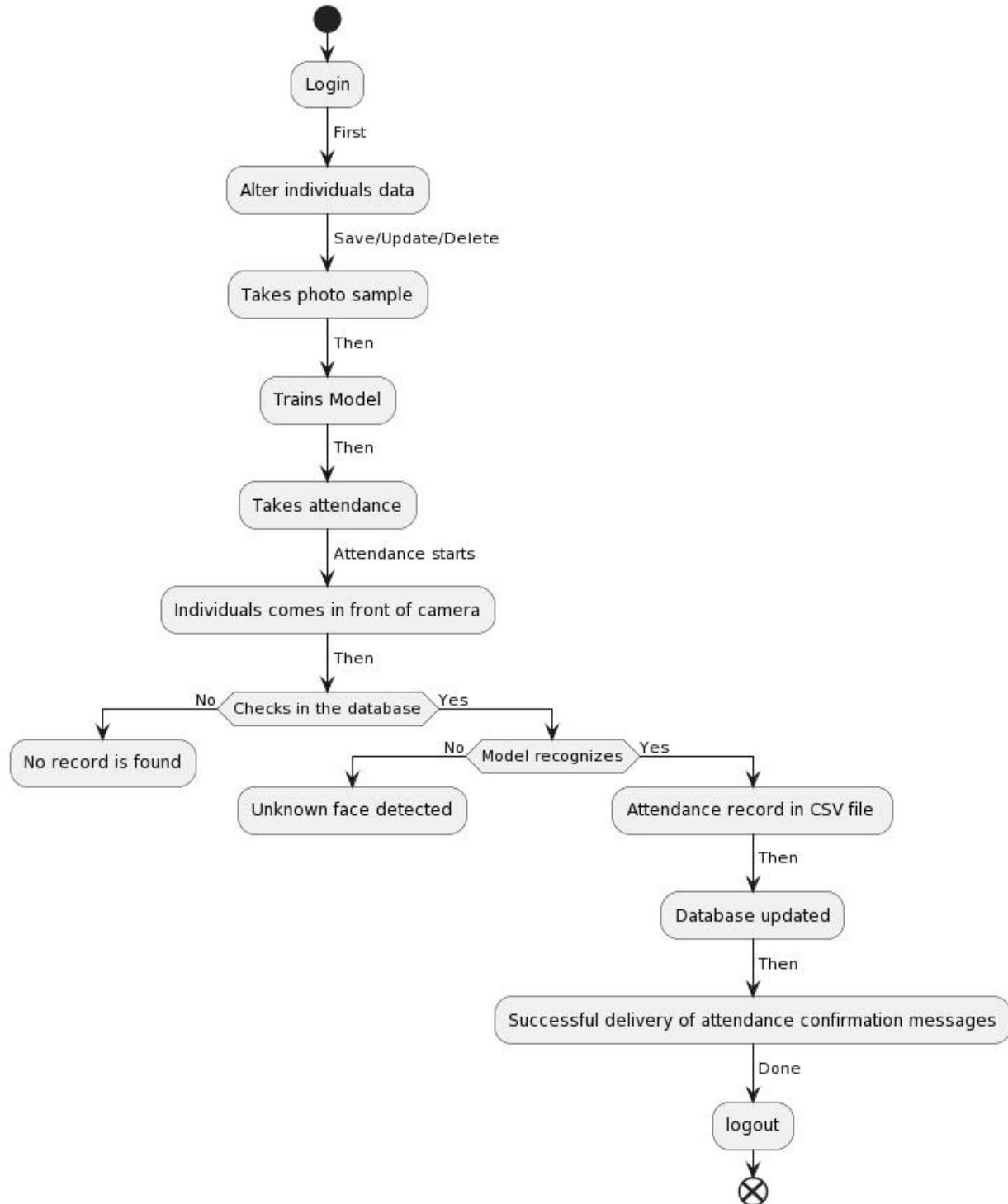


## Activity Diagram

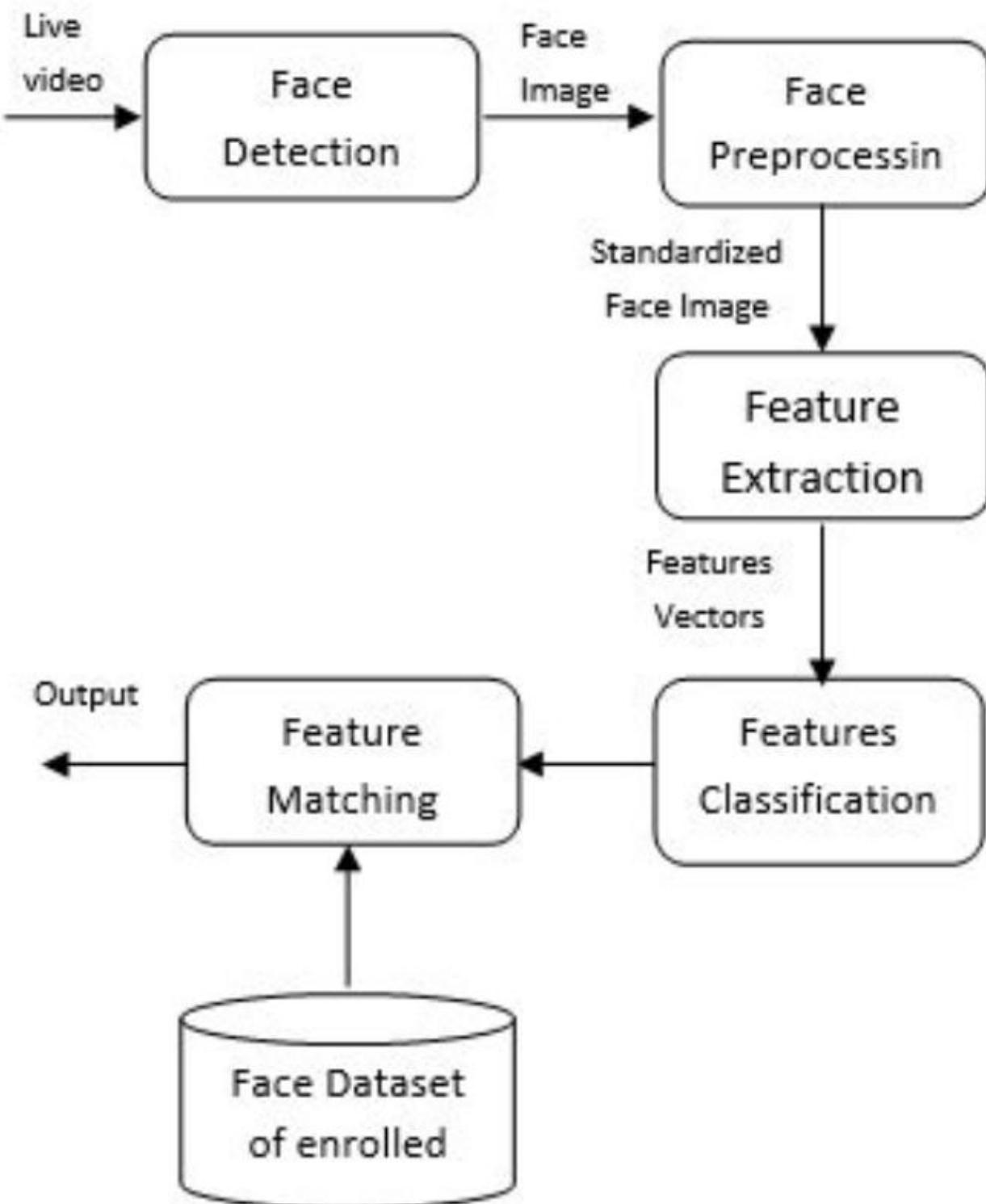
### 1) Admin



## 2) User



## System Design



## Conclusion

In conclusion, the implementation of a "**FaceTrek Attendance - Face recognition Attendance System**" represents a significant leap forward in the realm of workforce management. This technology not only streamlines the traditional attendance tracking process but also enhances security and accuracy. By leveraging facial recognition, organizations can mitigate the challenges associated with manual attendance systems, such as time fraud and buddy punching.

The face-based attendance system offers a user-friendly and non-intrusive method for employees to clock in and out, promoting efficiency and saving valuable time. Its integration with advanced algorithms ensures a high level of accuracy, reducing the likelihood of errors that may occur with traditional methods.

Moreover, the system contributes to a safer and healthier workplace environment by minimizing physical contact with attendance devices, a particularly important consideration in times of health concerns or global crises. The automation provided by "**FaceTrek Attendance - Face recognition Attendance System**" allows HR departments to focus on more strategic tasks, fostering productivity and strategic planning within the organization.

As we move towards an increasingly digitized era, "**FaceTrek Attendance - Face recognition Attendance System**" not only reflect technological advancement but also underscore a commitment to modern and efficient workforce management practices. Embracing this technology can lead to improved organizational processes, increased transparency, and ultimately, a more engaged and satisfied workforce.

During the documentation phase, I made plans for the "**FaceTrek Attendance - Face recognition Attendance System**" design. Additionally, I researched the recognition model I would utilize for this project and came to the conclusion that the Haar Cascade Face Recognition Algorithm would yield the greatest results overall. Additionally, I've learned about a variety of Python packages that I might use for my project and database needs. I've made an effort to enhance the system's functionality to make it more dependable and interactive.

This system may encounter numerous difficulties, but I'm willing to adapt it further in order to get expertise from the industry and strive to make this project even better.

# System Implementation (Code)

```
C:\Users\Shubham\Desktop\project\Face_Recognition_Attendance_Project\attendance.py (Face_Recognition_Attendance_Project) - Sublime Text (UNREGISTERED)

FOLDERS
└─ Face_Recognition_Attendance_Project
    └─ attendance.py

attendance.py
1  # Required Libraries
2  import cv2
3  from tkinter import*
4  from tkinter import ttk
5  from PIL import Image,ImageTk
6  from tkinter import messagebox
7  import time
8  Import os # For OS Operations
9  Import re # To Restrict Name Input To Alphabates Only
10 Import csv
11 From tkinter import filedialog # Window For Selecting Files
12 For creating invoice
13 Import gits
14 Import playsound
15
16 my_data=[] # Load Data From The CSV File Into Memory
17
18 # Main class
19 class Attendance:
20     def __init__(self,root):
21         self.root = root
22         self.root.geometry("1500x800x400")
23         self.root.title("Face Recognition System")
24
25         # Variables
26         self.var_attendance_id = StringVar()
27         self.var_attendance_individual_status = StringVar()
28         self.var_attendance_time = StringVar()
29         self.var_attendance_date = StringVar()
30         self.var_attendance_status = StringVar()
31
32
33         # First Top Image
34         img = Image.open("images\main\img1.jpg")
35         img = img.resize((550, 130), Image.Resampling.LANCZOS)
36         self.photoinc1 = ImageTk.PhotoImage(img)
37         f_l1b1 = Label(self.root,image=self.photoinc1)
38         f_l1b1.place(x=0,y=0,width=550,height=130)
39
40         # Second Top Image
41         img1 = Image.open("images\main\img2.jpg")
42         img1 = img1.resize((550, 130), Image.Resampling.LANCZOS)
43         self.photoinc2 = ImageTk.PhotoImage(img1)
44         f_l1b1 = Label(self.root,image=self.photoinc2)
45         f_l1b1.place(x=0,y=0,width=550,height=130)
46
47         # Third Top Image
48         img2 = Image.open("images\main\img3.png")
49         img2 = img2.resize((520, 130), Image.Resampling.LANCZOS)
50         self.photoinc3 = ImageTk.PhotoImage(img2)
51         f_l1b1 = Label(self.root,image=self.photoinc3)
52         f_l1b1.place(x=0,y=0,width=520,height=130)
53
54         # Top Title Label
55         title_label = Label(self.root,text="ATTENDANCE MANAGEMENT SYSTEM",font=("times new roman",35,"bold"),bg="white",fg="red")
56         title_label.place(x=10,y=100, width=1300, height=50)
```

C:\Users\Shubham\Desktop\project\Face\_Recognition\_Attendance\_Project\attendance.py (Face\_Recognition\_Attendance\_Project) - Sublime Text (UNREGISTERED)

**FOLDERS**

- Face\_Recognition\_Attendance\_Project
- \_\_pycache\_\_
- attendance
- Data
- images
- admin.py
- attendance.csv
- attendance.py
- classifier.xml
- face\_recognition.py
- haar cascade frontalface\_default.xml
- login.py
- main.py
- registration.py
- staff.py
- staff\_data.py
- student.py
- student\_data.py
- teacher.py
- teacher\_data.py
- train.py

attendance.py

```
197 # ***** Function Declaration *****
198
199 # ***** Reading Data From CSV File *****
200 def fetch_data(self,file):
201     self.Attendance_Report_table.delete(*self.Attendance_Report_table.get_children())
202     for row in file:
203         self.Attendance_Report_table.insert("",END,values=row)
204
205 # ***** Importing CSV File *****
206 def import_csv(self):
207     global mydata, file_name
208     mydata = []
209     file_name = filedialog.askopenfilename(initialdir=os.getcwd(),title="Open CSV",filetypes=(("CSV File","*.csv"),("All File","*.*")),parent=self.root)
210     with open(file_name, "r") as csvfile:
211         csv_read = csv.reader(csvfile, delimiter=',')
212         header = next(csv_read) # Skip The First Row
213         for row in csv_read:
214             mydata.append(row)
215         self.data_update(mydata)
216
217 # ***** Exporting CSV File *****
218 def export_csv(self):
219     try:
220         if len(mydata) < 1:
221             messagebox.showerror("Error","No Data Found To Export",parent = self.root)
222         else:
223             export_file_name = filedialog.asksaveasfilename(initialdir=os.getcwd(),title="Open CSV",filetypes=(("CSV File","*.csv"),("All File","*.*")),parent=self.root)
224             with open(export_file_name,mode='w',newline='') as myfile:
225                 export_write = csv.writer(myfile,delimiter=',')
226                 for i in mydata:
227                     export_write.writerow(i)
228             messagebox.showinfo("Success","Data Exported To File "+self.export_file_name+" Successfully",parent = self.root)
229     except Exception as e:
230         messagebox.showerror("Error",f"Due to {str(e)}",parent=self.root)
231
232 # ***** Updating CSV File *****
233 def update_csv(self):
234     if self.var_attendance_date.get() == "" or self.var_attendance_status.get() == "Select Status" or self.var_attendance_individual_status.get() == "Select Status" or self.var_attendance_id.get() == "":
235         messagebox.showerror("Error", "All Fields Are Required",parent=self.root)
236     else:
237         Update = messagebox.askyesno("Update", "Do You Want To Update This Attendance Record?",parent=self.root)
238         if Update == True:
239             selected_item = self.Attendance_Report_table.selection()
240             if selected_item:
241                 selected_item = selected_item[0]
242                 selected_id = self.Attendance_Report_table.item(selected_item, "values")[0]
243                 date = self.var_attendance_date.get()
244                 status = self.var_attendance_status.get()
245                 individual_status = self.var_attendance_individual_status.get()
246                 with open(file_name, 'r') as csvfile: # Read The CSV Data To Data List
247                     reader = csv.reader(csvfile)
248                     for row in reader:
249                         data.append(row)
250
251                 for row in data: # Update The Data List W.R.T Id Field
252                     if row[0] == selected_id:
253                         row[1] = date
254                         row[2] = status
255                         row[3] = individual_status
256
257                         data.append(row)
258
259                         for row in data:
260                             if row[0] == selected_id:
261                                 row[1] = date
262                                 row[2] = status
263                                 row[3] = individual_status
264
265                         data.append(row)
266
267                         for row in data:
268                             if row[0] == selected_id:
269                                 row[1] = date
270                                 row[2] = status
271                                 row[3] = individual_status
272
273                         data.append(row)
274
275                         for row in data:
276                             if row[0] == selected_id:
277                                 row[1] = date
278                                 row[2] = status
279                                 row[3] = individual_status
280
281                         data.append(row)
282
283                         for row in data:
284                             if row[0] == selected_id:
285                                 row[1] = date
286                                 row[2] = status
287                                 row[3] = individual_status
288
289                         data.append(row)
290
291                         for row in data:
292                             if row[0] == selected_id:
293                                 row[1] = date
294                                 row[2] = status
295                                 row[3] = individual_status
296
297                         data.append(row)
298
299                         for row in data:
300                             if row[0] == selected_id:
301                                 row[1] = date
302                                 row[2] = status
303                                 row[3] = individual_status
304
305                         data.append(row)
306
307                         for row in data:
308                             if row[0] == selected_id:
309                                 row[1] = date
310                                 row[2] = status
311                                 row[3] = individual_status
312
313                         data.append(row)
314
315                         for row in data:
316                             if row[0] == selected_id:
317                                 row[1] = date
318                                 row[2] = status
319                                 row[3] = individual_status
320
321                         data.append(row)
322
323                         for row in data:
324                             if row[0] == selected_id:
325                                 row[1] = date
326                                 row[2] = status
327                                 row[3] = individual_status
328
329                         data.append(row)
330
331                         for row in data:
332                             if row[0] == selected_id:
333                                 row[1] = date
334                                 row[2] = status
335                                 row[3] = individual_status
336
337                         data.append(row)
338
339                         for row in data:
340                             if row[0] == selected_id:
341                                 row[1] = date
342                                 row[2] = status
343                                 row[3] = individual_status
344
345                         data.append(row)
346
347                         for row in data:
348                             if row[0] == selected_id:
349                                 row[1] = date
350                                 row[2] = status
351                                 row[3] = individual_status
352
353                         data.append(row)
354
355                         for row in data:
356                             if row[0] == selected_id:
357                                 row[1] = date
358                                 row[2] = status
359                                 row[3] = individual_status
360
361                         data.append(row)
362
363                         for row in data:
364                             if row[0] == selected_id:
365                                 row[1] = date
366                                 row[2] = status
367                                 row[3] = individual_status
368
369                         data.append(row)
370
371                         for row in data:
372                             if row[0] == selected_id:
373                                 row[1] = date
374                                 row[2] = status
375                                 row[3] = individual_status
376
377                         data.append(row)
378
379                         for row in data:
380                             if row[0] == selected_id:
381                                 row[1] = date
382                                 row[2] = status
383                                 row[3] = individual_status
384
385                         data.append(row)
386
387                         for row in data:
388                             if row[0] == selected_id:
389                                 row[1] = date
390                                 row[2] = status
391                                 row[3] = individual_status
392
393                         data.append(row)
394
395                         for row in data:
396                             if row[0] == selected_id:
397                                 row[1] = date
398                                 row[2] = status
399                                 row[3] = individual_status
400
401                         data.append(row)
402
403                         for row in data:
404                             if row[0] == selected_id:
405                                 row[1] = date
406                                 row[2] = status
407                                 row[3] = individual_status
408
409                         data.append(row)
410
411                         for row in data:
412                             if row[0] == selected_id:
413                                 row[1] = date
414                                 row[2] = status
415                                 row[3] = individual_status
416
417                         data.append(row)
418
419                         for row in data:
420                             if row[0] == selected_id:
421                                 row[1] = date
422                                 row[2] = status
423                                 row[3] = individual_status
424
425                         data.append(row)
426
427                         for row in data:
428                             if row[0] == selected_id:
429                                 row[1] = date
430                                 row[2] = status
431                                 row[3] = individual_status
432
433                         data.append(row)
434
435                         for row in data:
436                             if row[0] == selected_id:
437                                 row[1] = date
438                                 row[2] = status
439                                 row[3] = individual_status
440
441                         data.append(row)
442
443                         for row in data:
444                             if row[0] == selected_id:
445                                 row[1] = date
446                                 row[2] = status
447                                 row[3] = individual_status
448
449                         data.append(row)
450
451                         for row in data:
452                             if row[0] == selected_id:
453                                 row[1] = date
454                                 row[2] = status
455                                 row[3] = individual_status
456
457                         data.append(row)
458
459                         for row in data:
460                             if row[0] == selected_id:
461                                 row[1] = date
462                                 row[2] = status
463                                 row[3] = individual_status
464
465                         data.append(row)
466
467                         for row in data:
468                             if row[0] == selected_id:
469                                 row[1] = date
470                                 row[2] = status
471                                 row[3] = individual_status
472
473                         data.append(row)
474
475                         for row in data:
476                             if row[0] == selected_id:
477                                 row[1] = date
478                                 row[2] = status
479                                 row[3] = individual_status
480
481                         data.append(row)
482
483                         for row in data:
484                             if row[0] == selected_id:
485                                 row[1] = date
486                                 row[2] = status
487                                 row[3] = individual_status
488
489                         data.append(row)
490
491                         for row in data:
492                             if row[0] == selected_id:
493                                 row[1] = date
494                                 row[2] = status
495                                 row[3] = individual_status
496
497                         data.append(row)
498
499                         for row in data:
500                             if row[0] == selected_id:
501                                 row[1] = date
502                                 row[2] = status
503                                 row[3] = individual_status
504
505                         data.append(row)
506
507                         for row in data:
508                             if row[0] == selected_id:
509                                 row[1] = date
510                                 row[2] = status
511                                 row[3] = individual_status
512
513                         data.append(row)
514
515                         for row in data:
516                             if row[0] == selected_id:
517                                 row[1] = date
518                                 row[2] = status
519                                 row[3] = individual_status
520
521                         data.append(row)
522
523                         for row in data:
524                             if row[0] == selected_id:
525                                 row[1] = date
526                                 row[2] = status
527                                 row[3] = individual_status
528
529                         data.append(row)
530
531                         for row in data:
532                             if row[0] == selected_id:
533                                 row[1] = date
534                                 row[2] = status
535                                 row[3] = individual_status
536
537                         data.append(row)
538
539                         for row in data:
540                             if row[0] == selected_id:
541                                 row[1] = date
542                                 row[2] = status
543                                 row[3] = individual_status
544
545                         data.append(row)
546
547                         for row in data:
548                             if row[0] == selected_id:
549                                 row[1] = date
550                                 row[2] = status
551                                 row[3] = individual_status
552
553                         data.append(row)
554
555                         for row in data:
556                             if row[0] == selected_id:
557                                 row[1] = date
558                                 row[2] = status
559                                 row[3] = individual_status
560
561                         data.append(row)
562
563                         for row in data:
564                             if row[0] == selected_id:
565                                 row[1] = date
566                                 row[2] = status
567                                 row[3] = individual_status
568
569                         data.append(row)
570
571                         for row in data:
572                             if row[0] == selected_id:
573                                 row[1] = date
574                                 row[2] = status
575                                 row[3] = individual_status
576
577                         data.append(row)
578
579                         for row in data:
580                             if row[0] == selected_id:
581                                 row[1] = date
582                                 row[2] = status
583                                 row[3] = individual_status
584
585                         data.append(row)
586
587                         for row in data:
588                             if row[0] == selected_id:
589                                 row[1] = date
590                                 row[2] = status
591                                 row[3] = individual_status
592
593                         data.append(row)
594
595                         for row in data:
596                             if row[0] == selected_id:
597                                 row[1] = date
598                                 row[2] = status
599                                 row[3] = individual_status
600
601                         data.append(row)
602
603                         for row in data:
604                             if row[0] == selected_id:
605                                 row[1] = date
606                                 row[2] = status
607                                 row[3] = individual_status
608
609                         data.append(row)
610
611                         for row in data:
612                             if row[0] == selected_id:
613                                 row[1] = date
614                                 row[2] = status
615                                 row[3] = individual_status
616
617                         data.append(row)
618
619                         for row in data:
620                             if row[0] == selected_id:
621                                 row[1] = date
622                                 row[2] = status
623                                 row[3] = individual_status
624
625                         data.append(row)
626
627                         for row in data:
628                             if row[0] == selected_id:
629                                 row[1] = date
630                                 row[2] = status
631                                 row[3] = individual_status
632
633                         data.append(row)
634
635                         for row in data:
636                             if row[0] == selected_id:
637                                 row[1] = date
638                                 row[2] = status
639                                 row[3] = individual_status
640
641                         data.append(row)
642
643                         for row in data:
644                             if row[0] == selected_id:
645                                 row[1] = date
646                                 row[2] = status
647                                 row[3] = individual_status
648
649                         data.append(row)
650
651                         for row in data:
652                             if row[0] == selected_id:
653                                 row[1] = date
654                                 row[2] = status
655                                 row[3] = individual_status
656
657                         data.append(row)
658
659                         for row in data:
660                             if row[0] == selected_id:
661                                 row[1] = date
662                                 row[2] = status
663                                 row[3] = individual_status
664
665                         data.append(row)
666
667                         for row in data:
668                             if row[0] == selected_id:
669                                 row[1] = date
670                                 row[2] = status
671                                 row[3] = individual_status
672
673                         data.append(row)
674
675                         for row in data:
676                             if row[0] == selected_id:
677                                 row[1] = date
678                                 row[2] = status
679                                 row[3] = individual_status
680
681                         data.append(row)
682
683                         for row in data:
684                             if row[0] == selected_id:
685                                 row[1] = date
686                                 row[2] = status
687                                 row[3] = individual_status
688
689                         data.append(row)
690
691                         for row in data:
692                             if row[0] == selected_id:
693                                 row[1] = date
694                                 row[2] = status
695                                 row[3] = individual_status
696
697                         data.append(row)
698
699                         for row in data:
700                             if row[0] == selected_id:
701                                 row[1] = date
702                                 row[2] = status
703                                 row[3] = individual_status
704
705                         data.append(row)
706
707                         for row in data:
708                             if row[0] == selected_id:
709                                 row[1] = date
710                                 row[2] = status
711                                 row[3] = individual_status
712
713                         data.append(row)
714
715                         for row in data:
716                             if row[0] == selected_id:
717                                 row[1] = date
718                                 row[2] = status
719                                 row[3] = individual_status
720
721                         data.append(row)
722
723                         for row in data:
724                             if row[0] == selected_id:
725                                 row[1] = date
726                                 row[2] = status
727                                 row[3] = individual_status
728
729                         data.append(row)
730
731                         for row in data:
732                             if row[0] == selected_id:
733                                 row[1] = date
734                                 row[2] = status
735                                 row[3] = individual_status
736
737                         data.append(row)
738
739                         for row in data:
740                             if row[0] == selected_id:
741                                 row[1] = date
742                                 row[2] = status
743                                 row[3] = individual_status
744
745                         data.append(row)
746
747                         for row in data:
748                             if row[0] == selected_id:
749                                 row[1] = date
750                                 row[2] = status
751                                 row[3] = individual_status
752
753                         data.append(row)
754
755                         for row in data:
756                             if row[0] == selected_id:
757                                 row[1] = date
758                                 row[2] = status
759                                 row[3] = individual_status
760
761                         data.append(row)
762
763                         for row in data:
764                             if row[0] == selected_id:
765                                 row[1] = date
766                                 row[2] = status
767                                 row[3] = individual_status
768
769                         data.append(row)
770
771                         for row in data:
772                             if row[0] == selected_id:
773                                 row[1] = date
774                                 row[2] = status
775                                 row[3] = individual_status
776
777                         data.append(row)
778
779                         for row in data:
780                             if row[0] == selected_id:
781                                 row[1] = date
782                                 row[2] = status
783                                 row[3] = individual_status
784
785                         data.append(row)
786
787                         for row in data:
788                             if row[0] == selected_id:
789                                 row[1] = date
790                                 row[2] = status
791                                 row[3] = individual_status
792
793                         data.append(row)
794
795                         for row in data:
796                             if row[0] == selected_id:
797                                 row[1] = date
798                                 row[2] = status
799                                 row[3] = individual_status
800
801                         data.append(row)
802
803                         for row in data:
804                             if row[0] == selected_id:
805                                 row[1] = date
806                                 row[2] = status
807                                 row[3] = individual_status
808
809                         data.append(row)
810
811                         for row in data:
812                             if row[0] == selected_id:
813                                 row[1] = date
814                                 row[2] = status
815                                 row[3] = individual_status
816
817                         data.append(row)
818
819                         for row in data:
820                             if row[0] == selected_id:
821                                 row[1] = date
822                                 row[2] = status
823                                 row[3] = individual_status
824
825                         data.append(row)
826
827                         for row in data:
828                             if row[0] == selected_id:
829                                 row[1] = date
830                                 row[2] = status
831                                 row[3] = individual_status
832
833                         data.append(row)
834
835                         for row in data:
836                             if row[0] == selected_id:
837                                 row[1] = date
838                                 row[2] = status
839                                 row[3] = individual_status
840
841                         data.append(row)
842
843                         for row in data:
844                             if row[0] == selected_id:
845                                 row[1] = date
846                                 row[2] = status
847                                 row[3] = individual_status
848
849                         data.append(row)
850
851                         for row in data:
852                             if row[0] == selected_id:
853                                 row[1] = date
854                                 row[2] = status
855                                 row[3] = individual_status
856
857                         data.append(row)
858
859                         for row in data:
860                             if row[0] == selected_id:
861                                 row[1] = date
862                                 row[2] = status
863                                 row[3] = individual_status
864
865                         data.append(row)
866
867                         for row in data:
868                             if row[0] == selected_id:
869                                 row[1] = date
870                                 row[2] = status
871                                 row[3] = individual_status
872
873                         data.append(row)
874
875                         for row in data:
876                             if row[0] == selected_id:
877                                 row[1] = date
878                                 row[2] = status
879                                 row[3] = individual_status
880
881                         data.append(row)
882
883                         for row in data:
884                             if row[0] == selected_id:
885                                 row[1] = date
886                                 row[2] = status
887                                 row[3] = individual_status
888
889                         data.append(row)
890
891                         for row in data:
892                             if row[0] == selected_id:
893                                 row[1] = date
894                                 row[2] = status
895                                 row[3] = individual_status
896
897                         data.append(row)
898
899                         for row in data:
900                             if row[0] == selected_id:
901                                 row[1] = date
902                                 row[2] = status
903                                 row[3] = individual_status
904
905                         data.append(row)
906
907                         for row in data:
908                             if row[0] == selected_id:
909                                 row[1] = date
910                                 row[2] = status
911                                 row[3] = individual_status
912
913                         data.append(row)
914
915                         for row in data:
916                             if row[0] == selected_id:
917                                 row[1] = date
918                                 row[2] = status
919                                 row[3] = individual_status
920
921                         data.append(row)
922
923                         for row in data:
924                             if row[0] == selected_id:
925                                 row[1] = date
926                                 row[2] = status
927                                 row[3] = individual_status
928
929                         data.append(row)
930
931                         for row in data:
932                             if row[0] == selected_id:
933                                 row[1] = date
934                                 row[2] = status
935                                 row[3] = individual_status
936
937                         data.append(row)
938
939                         for row in data:
940                             if row[0] == selected_id:
941                                 row[1] = date
942                                 row[2] = status
943                                 row[3] = individual_status
944
945                         data.append(row)
946
947                         for row in data:
948                             if row[0] == selected_id:
949                                 row[1] = date
950                                 row[2] = status
951                                 row[3] = individual_status
952
953                         data.append(row)
954
955                         for row in data:
956                             if row[0] == selected_id:
957                                 row[1] = date
958                                 row[2] = status
959                                 row[3] = individual_status
960
961                         data.append(row)
962
963                         for row in data:
964                             if row[0] == selected_id:
965                                 row[1] = date
966                                 row[2] = status
967                                 row[3] = individual_status
968
969                         data.append(row)
970
971                         for row in data:
972                             if row[0] == selected_id:
973                                 row[1] = date
974                                 row[2] = status
975                                 row[3] = individual_status
976
977                         data.append(row)
978
979                         for row in data:
980                             if row[0] == selected_id:
981                                 row[1] = date
982                                 row[2] = status
983                                 row[3] = individual_status
984
985                         data.append(row)
986
987                         for row in data:
988                             if row[0] == selected_id:
989                                 row[1] = date
990                                 row[2] = status
991                                 row[3] = individual_status
992
993                         data.append(row)
994
995                         for row in data:
996                             if row[0] == selected_id:
997                                 row[1] = date
998                                 row[2] = status
999                                 row[3] = individual_status
1000
1001                         data.append(row)
1002
1003                         for row in data:
1004                             if row[0] == selected_id:
1005                                 row[1] = date
1006                                 row[2] = status
1007                                 row[3] = individual_status
1008
1009                         data.append(row)
1010
1011                         for row in data:
1012                             if row[0] == selected_id:
1013                                 row[1] = date
1014                                 row[2] = status
1015                                 row[3] = individual_status
1016
1017                         data.append(row)
1018
1019                         for row in data:
1020                             if row[0] == selected_id:
1021                                 row[1] = date
1022                                 row[2] = status
1023                                 row[3] = individual_status
1024
1025                         data.append(row)
1026
1027                         for row in data:
1028                             if row[0] == selected_id:
1029                                 row[1] = date
1030                                 row[2] = status
1031                                 row[3] = individual_status
1032
1033                         data.append(row)
1034
1035                         for row in data:
1036                             if row[0] == selected_id:
1037                                 row[1] = date
1038                                 row[2] = status
1039                                 row[3] = individual_status
1040
1041                         data.append(row)
1042
1043                         for row in data:
1044                             if row[0] == selected_id:
1045                                 row[1] = date
1046                                 row[2] = status
1047                                 row[3] = individual_status
1048
1049                         data.append(row)
1050
1051                         for row in data:
1052                             if row[0] == selected_id:
1053                                 row[1] = date
1054                                 row[2] = status
1055                                 row[3] = individual_status
1056
1057                         data.append(row)
1058
1059                         for row in data:
1060                             if row[0] == selected_id:
1061                                 row[1] = date
1062                                 row[2] = status
1063                                 row[3] = individual_status
1064
1065                         data.append(row)
1066
1067                         for row in data:
1068                             if row[0] == selected_id:
1069                                 row[1] = date
1070                                 row[2] = status
1071                                 row[3] = individual_status
1072
1073                         data.append(row)
1074
1075                         for row in data:
1076                             if row[0] == selected_id:
1077                                 row[1] = date
1078                                 row[2] = status
1079                                 row[3] = individual_status
1080
1081                         data.append(row)
1082
1083                         for row in data:
1084                             if row[0] == selected_id:
1085                                 row[1] = date
1086                                 row[2] = status
1087                                 row[3] = individual_status
1088
1089                         data.append(row)
1090
1091                         for row in data:
1092                             if row[0] == selected_id:
1093                                 row[1] = date
1094                                 row[2] = status
1095                                 row[3] = individual_status
1096
1097                         data.append(row)
1098
1099                         for row in data:
1100                             if row[0] == selected_id:
1101                                 row[1] = date
1102                                 row[2] = status
1103                                 row[3] = individual_status
1104
1105                         data.append(row)
1106
1107                         for row in data:
1108                             if row[0] == selected_id:
1109                                 row[1] = date
1110                                 row[2] = status
1111                                 row[3] = individual_status
1112
1113                         data.append(row)
1114
1115                         for row in data:
1116                             if row[0] == selected_id:
1117                                 row[1] = date
1118                                 row[2] = status
1119                                 row[3] = individual_status
1120
1121                         data.append(row)
1122
1123                         for row in data:
1124                             if row[0] == selected_id:
1125                                 row[1] = date
1126                                 row[2] = status
1127                                 row[3] = individual_status
1128
1129                         data.append(row)
1130
1131                         for row in data:
1132                             if row[0] == selected_id:
1133                                 row[1] = date
1134                                 row[2] = status
1135                                 row[3] = individual_status
1136
1137                         data.append(row)
1138
1139                         for row in data:
1140                             if row[0] == selected_id:
1141                                 row[1] = date
1142                                 row[2] = status
1143                                 row[3] = individual_status
1144
1145                         data.append(row)
1146
1147                         for row in data:
1148                             if row[0] == selected_id:
1149                                 row[1] = date
1150                                 row[2] = status
1151                                 row[3] = individual_status
1152
1153                         data.append(row)
1154
1155                         for row in data:
1156                             if row[0] == selected_id:
1157                                 row[1] = date
1158                                 row[2] = status
1159                                 row[3] = individual_status
1160
1161                         data.append(row)
1162
1163                         for row in data:
1164                             if row[0] == selected_id:
1165                                 row[1] = date
1166                                 row[2] = status
1167                                 row[3] = individual_status
1168
1169                         data.append(row)
1170
1171                         for row in data:
1172                             if row[0] == selected_id:
1173                                 row[1] = date
1174                                 row[2] = status
1175                                 row[3] = individual_status
1176
1177                         data.append(row)
1178
1179                         for row in data:
1180                             if row[0] == selected_id:
1181                                 row[1] = date
1182                                 row[2] = status
1183                                 row[3] = individual_status
1184
1185                         data.append(row)
1186
1187                         for row in data:
1188                             if row[0] == selected_id:
1189                                 row[1] = date
1190                                 row[2] = status
1191                                 row[3] = individual_status
1192
1193                         data.append(row)
1194
1195                         for row in data:
1196                             if row[0] == selected_id:
1197                                 row[1] = date
1198                                 row[2] = status
1199                                 row[3] = individual_status
1200
1201                         data.append(row)
1202
1203                         for row in data:
1204                             if row[0] == selected_id:
1205                                 row[1] = date
1206                                 row[2] = status
1207                                 row[3] = individual_status
1208
1209                         data.append(row)
1210
1211                         for row in data:
1212                             if row[0] == selected_id:
1213                                 row[1] = date
1214                                 row[2] = status
1215                                 row[3] = individual_status
1216
1217                         data.append(row)
1218
1219                         for row in data:
1220                             if row[0] == selected_id:
1221                                 row[1] = date
1222                                 row[2] = status
1223                                 row[3] = individual_status
1224
1225                         data.append(row)
1226
1227                         for row in data:
1228                             if row[0] == selected_id:
1229                                 row[1] = date
1230                                 row[2] = status
1231                                 row[3] = individual_status
1232
1233                         data.append(row)
1234
1235                         for row in data:
1236                             if row[0] == selected_id:
1237                                 row[1] = date
1238                                 row[2] = status
1239                                 row[3] = individual_status
1240
1241                         data.append(row)
1242
1243                         for row in data:
1244                             if row[0] == selected_id:
1245                                 row[1] = date
1246                                 row[2] = status
1247                                 row[3] = individual_status
1248
1249                         data.append(row)
1250
1251                         for row in data:
1252                             if row[0] == selected_id:
1253                                 row[1] = date
1254                                 row[2] = status
1255                                 row[3] = individual_status
1256
1257                         data.append(row)
1258
1259                         for row in data:
1260                             if row[0] == selected_id:
1261                                 row[1] = date
1262                                 row[2] = status
1263                                 row[3] = individual_status
1264
1265                         data.append(row)
1266
1267                         for row in data:
1268                             if row[0] == selected_id:
1269                                 row[1] = date
1270                                 row[2] = status
1271                                 row[3] = individual_status
1272
1273                         data.append(row)
1274
1275                         for row in data:
1276                             if row[0] == selected_id:
1277                                 row[1] = date
1278                                 row[2] = status
1279                                 row[3] = individual_status
1280
1281                         data.append(row)
1282
1283                         for row in data:
1284                             if row[0] == selected_id:
1285                                 row[1] = date
1286                                 row[2] = status
1287                                 row[3] = individual_status
1288
1289                         data.append(row)
1290
1291                         for row in data:
1292                             if row[0] == selected_id:
1293                                 row[1] = date
1294                                 row[2] = status
1295                                 row[3] = individual_status
1296
1297                         data.append(row)
1298
1299                         for row in data:
1300                             if row[0] == selected_id:
1301                                 row[1] = date
1302                                 row[2] = status
1303                                 row[3] = individual_status
1304
1305                         data.append(row)
1306
1307                         for row in data:
1308                             if row[0] == selected_id:
1309                                 row[1] = date
1310                                 row[2] = status
1311                                 row[3] = individual_status
1312
1313                         data.append(row)
1314
1315                         for row in data:
1316                             if row[0] == selected_id:
1317                                 row[1] = date
1318                                 row[2] = status
1319                                 row[3] = individual_status
1320
1321                         data.append(row)
1322
1323                         for row in data:
1324                             if row[0] == selected_id:
1325                                 row[1] = date
1326                                 row[2] = status
1327                                 row[3] = individual_status
1328
1329                         data.append(row)
1330
1331                         for row in data:
1332                             if row[0] == selected_id:
1333                                 row[1] = date
1334                                 row[2] = status
1335                                 row[3] = individual_status
1336
1337                         data.append(row)
1338
1339                         for row in data:
1340                             if row[0] == selected_id:
1341                                 row[1] = date
1342                                 row[2] = status
1343                                 row[3] = individual_status
1344
1345                         data.append(row)
1346
1347                         for row in data:
1348                             if row[0] == selected_id:
1349                                 row[1] = date
1350                                 row[2] = status
1351                                 row[3] = individual_status
1352
1353                         data.append(row)
1354
1355                         for row in data:
1356                             if row[0] == selected_id:
1357                                 row[1] = date
1358                                 row[2] = status
1359                                 row[3] = individual_status
1360
1361                         data.append(row)
1362
1363                         for row in data:
1364                             if row[0] == selected_id:
1365                                 row[1] = date
1366                                 row[2] = status
1367                                 row[3] = individual_status
1368
1369                         data.append(row)
1370
1371                         for row in data:
1372                             if row[0] == selected_id:
1373                                 row[1] = date
1374                                 row[2] = status
1375                                 row[3] = individual_status
1376
1377                         data.append(row)
1378
1379                         for row in data:
1380                             if row[0] == selected_id:
1381                                 row[1] = date
1382                                 row[2] = status
1383                                 row[3] = individual_status
1384
1385                         data.append(row)
1386
1387                         for row in data:
1388                             if row[0] == selected_id:
1389                                 row[1] = date
1390                                 row[2] = status
1391                                 row[3] = individual_status
1392
1393                         data.append(row)
1394
1395                         for row in data:
1396                             if row[0] == selected_id:
1397                                 row[1] = date
1398                                 row[2] = status
1399                                 row[3] = individual_status
1400
1401                         data.append(row)
1402
1403                         for row in data:
1404                             if row[0] == selected_id:
1405                                 row[1] = date
1406                                 row[2] = status
1407                                 row[3] = individual_status
1408
1409                         data.append(row)
1410
1411                         for row in data:
1412                             if row[0] == selected_id:
1413                                 row[1] = date
1414                                 row[2] = status
1415                                 row[3] = individual_status
1416
1417                         data.append(row)
1418
1419                         for row in data:
1420                             if row[0] == selected_id:
1421                                 row[1] = date
1422                                 row[2] = status
1423                                 row[3] = individual_status
1424
1425                         data.append(row)
1426
1427                         for row in data:
1428                             if row[0] == selected_id:
1429                                 row[1] = date
1430                                 row[2] = status
1431                                 row[3] = individual_status
1432
1433                         data.append(row)
1434
1435                         for row in data:
1436                             if row[0] == selected_id:
1437                                 row[1] = date
1438                                 row[2] = status
1439                                 row[3] = individual_status
1440
1441                         data.append(row)
1442
1443                         for row in data:
1444                             if row[0] == selected_id:
1445                                 row[1] = date
1446                                 row[2] = status
1447                                 row[3] = individual_status
1448
1449                         data.append(row)
1450
1451                         for row in data:
1452                             if row[0] == selected_id:
1453                                 row[1] = date
1454                                 row[2] = status
1455                                 row[3] = individual_status
1456
1457                         data.append(row)
1458
1459                         for row in data:
1460                             if row[0] == selected_id:
1461                                 row[1] = date
14
```

C:\Users\Shubham\Desktop\projectFace\_Recognition\_Attendance\_Project\attendance.py (Face\_Recognition\_Attendance\_Project) - Sublime Text (UNREGISTERED)

**FOLDERS**

- face\_recognition\_Attendance\_Project
- \_\_pycache\_\_
- attendance
- Data
- images
- admin.py
- attendance.csv
- attendance.py

<> classifier.xml

<> face\_recognition.py

<> haarcascade\_frontalface\_default.xml

<> login.py

<> main.py

<> reparation.py

<> staff.py

<> staff\_data.py

<> student.py

<> student\_data.py

<> teacher.py

<> teacher\_data.py

<> train.py

attendance.py

```
261     writer.writerow(data)
262 
263     # Invoice Code After Successful Completion Of Operation
264     sound = gttts.gTTS("Attendance Record Has Been Updated Successfully",lang = "en")
265     sound.save("attendance.mp3")
266     playsound.playsound("attendance.mp3")
267     if os.path.exists("attendance.mp3"):
268         os.remove("attendance.mp3")
269 
270     messagebox.showinfo("Success","Attendance Record Has Been Updated Successfully",parent=self.root)
271 
272     # Populate The Treeview With Updated CSV File
273     my_data = []
274     with open((filenam)) as myfile:
275         csv_read = csv.reader(myfile,delimiter=",")
276         header = next(csv_read) # Skip the first row
277         for i in csv.read():
278             my_data.append(i)
279         self.tree.delete(*self.tree.get_children())
280         self.tree_data(my_data)
281 
282     else:
283         if not Update:
284             return
285 
286     # ***** Reset All The Fields To Default Values *****
287     def reset_data(self):
288         self.var_attendance_id.set("Select Status")
289         self.var_attendance_id.set("Status only")
290         self.var_attendance_date.set("Select Date")
291         self.var_attendance_name.set("Select Name")
292         self.var_attendance_name.set("Select Name")
293 
294     # ***** Fetch The Data From Treeview To The Input Fields *****
295     def get_data(self):
296         cursor = self.Attendance_Report_table.focus()
297         content = self.Attendance_Report_table.item(cursor['index'])
298         data = content['values']
299 
300         self.var_attendance_id.set(data[0])
301         self.var_attendance_individual_status.set(data[1])
302         self.var_attendance_name.set(data[2])
303         self.var_attendance_date.set(data[3])
304         self.var_attendance_time.set(data[4])
305         self.var_attendance_status.set(data[5])
306 
307     # ***** Back To Home Function *****
308     def back_to_main(self):
309         self.root.destroy()
310 
311     # ----- Main Class Calling -----
312     if __name__ == "__main__":
313         root = Tk()
314         obj = Attendance(root)
315         root.mainloop()
```

**face\_recognition.py**

```

1 # Required Libraries
2 # For GUI
3 import tkinter
4 from PIL import Image,ImageTk
5 import pyautogui,pyttsx3,goslate # For MongoDB
6 # For Date And Time
7 from time import strftime
8 # For Image Processing
9 import cv2 # For Face Recognition
10 import os # For OS Operations
11 # For Python GUI
12 import smtplib
13 # For Twilio Message Import EmailMessage
14 from twilio.rest import Client # For Sending Text SMS To The Phone Number
15 # For Generating Invoice
16 import playsound
17
18 # Main Class
19 class Face_Recognition:
20     def __init__(self,root):
21         self.root = root
22         self.root.title("150hd20hd40H")
23         self.root.title("Face Recognition System")
24
25         # First Top Image
26         img = Image.open("images/main\img1.jpg")
27         img1 = img.resize((510,130),Image.Resampling.LANCZOS)
28         self.photoinc1 = ImageTk.PhotoImage(img1)
29         f_lbl = Label(self.root,image=self.photoinc1)
30         f_lbl.place(x=0,y=0,width=510,height=130)
31
32         # Second Top Image
33         img2 = Image.open("images/main\img2.jpg")
34         img1 = img2.resize((550,130),Image.Resampling.LANCZOS)
35         self.photoinc2 = ImageTk.PhotoImage(img1)
36         f_lbl = Label(self.root,image=self.photoinc2)
37         f_lbl.place(x=0,y=0,width=550,height=130)
38
39         # Third Top Image
40         img3 = Image.open("images/main\img3.jpg")
41         img1 = img3.resize((520,130),Image.Resampling.LANCZOS)
42         self.photoinc3 = ImageTk.PhotoImage(img1)
43         f_lbl = Label(self.root,image=self.photoinc3)
44         f_lbl.place(x=0,y=0,width=520,height=130)
45
46         # Top Title Image
47         title_label = Label(self.root,text="FACE RECOGNITION",font=("times new roman",35,"bold"),bg="white",fg="red")
48         title_label.place(x=0,y=130,width=530,height=45)
49
50         # Back To Home Button At Label
51         back_button = Button(self.root,command=self.back_to_home,cursor="hand2",text="Home",width=10,font="times new roman",12,"bold")
52         back_button.place(x=140,y=130)
53
54         # To Display Time At Label
55         self.time()
56
57         # To Display Date At Label
58         self.date()
59
60         #----- Function Declaration -----
61
62         #----- Mark Attendance To CSV And Send Confirmation Message -----
63         def mark_attendance(self,sid,status,name,email,phone):
64             with open('attendance.csv','a+',newline='') as f: # Create File Manually For 1st Time
65                 my_data_list = []
66                 f.readline()
67                 f.readline()
68                 name_list = []
69                 for line in f.readlines():
70                     entry = line.split(',')
71                     if((sid not in entry) and (status not in name_list) and (name not in name_list)):
72                         dt = datetime.now().strftime("%d/%m/%Y")
73                         dt_string = str(dt).split("-")
74                         f.write(f'{(sid),(status),(name),(dt),(dt_string)},')
75                         message = name + " is present at the time : "+dt_string+" on the day : "+dt
76                         msg = EmailMessage()
77                         msg['Subject'] = "Attendance"
78                         msg['To'] = email
79                         msg['From'] = "karlojoseph5@gmail.com"
80                         server = smtplib.SMTP('smtp.gmail.com:587')
81                         server.starttls()
82                         server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
83                         server.send_message(msg)
84                         server.quit()
85
86                         #----- Send SMC Confirmation Message -----
87                         def send_mail(self,message,email):
88                             msg = EmailMessage()
89                             msg['Subject'] = "Attendance"
90                             msg['To'] = email
91                             msg['From'] = "karlojoseph5@gmail.com"
92                             server = smtplib.SMTP('smtp.gmail.com:587')
93                             server.starttls()
94                             server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
95                             server.send_message(msg)
96                             server.quit()
97
98                         #----- Send SMC Confirmation Message -----
99                         def send_mark(self,message,phone):
100                             account_sid = "AC51a1001a14df0d995a993273d3004" # Twilio Account SID
101                             auth_token = "f5e0a2a2a2a2a2a2a2a2a2a2a2a2a2a2" # Twilio Account Authentication Token
102                             client = Client(account_sid, auth_token) # Create A Twilio Client
103                             twilio_phone_number = "+12502864843" # Your Twilio Phone Number (You Must Have Purchased This Number On Hello)
104                             recipient_phone_number = "+919876543210" # Recipient's Phone Number (In E.164 Format Including Country Code, e.g., +919876543210)
105                             client.messages.create(body = message, from_ = twilio_phone_number, to = recipient_phone_number) # Send The SMS
106
107                         #----- Face Recognition Function -----
108                         def face_recogn(self):
109                             def recognize(img,classifier,scaleFactor,minNeighbors,color,text,cif):
110                                 gray_img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
111                                 coordinates = detectMultiScale(gray_img,scaleFactor,minNeighbors)
112                                 for(x,y,w,h) in features:
113                                     cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)
114                                     id,predict = cif.predict(gray_img[y:y+h,x:x+w])
115                                     confidence = int((100*(1-predict/300)))
116
117                         #----- Face Recognition -----
118                         if confidence > 77 and str(status) == "Current Date":
119                             cv2.putText(img,str(status),(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(100,100,0),3)
120                             cv2.putText(img,"Id : (Individual Id)",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
121                             cv2.putText(img,"Name : (Name)",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
122                             if c == 1:
123                                 collection = db1.student_details
124                                 collection.update_one({"student_id":individual_id}, {"$set": {"attendance": "present"}})
125                             elif c == 2:
126                                 collection = db1.teacher_details
127                                 collection.update_one({"teacher_id":individual_id}, {"$set": {"attendance": "present"}})
128                             elif c == 3:
129                                 collection = db1.staff_details
130                                 collection.update_one({"staff_id":individual_id}, {"$set": {"attendance": "present"}})
131                             self.mark_attendance(individual_id,status,name,email,phone)
132
133                             cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
134                             cv2.putText(img,"Unknown Face Detected", (x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
135                             sound = gts.gTTS("Unknown Face Detected",lang = "en")
136                             sound.save("recognize.mp3")
137                             os.system("recognize.mp3")
138                             coordinates = [x,y,w,h]
139                             client = pyaudio.PyAudio()
140                             playsound.playsound("recognize.mp3")
141
142                         # Recurring Individual
143                         faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
144                         clf = cv2.face.LBPHFaceRecognizer_create()
145                         id = 0
146                         video_cap = cv2.VideoCapture(0)
147                         while True:
148                             ret,video_cap.read()
149                             img = recognize(img,clf,faceCascade)
150                             cv2.imshow("Face to Face Recognition",img)
151                             if cv2.waitKey(1) == 13:
152                                 break
153                             video_cap.release()
154                             cv2.destroyAllWindows()
155
156                         #----- Back To Home Function -----
157                         def back_to_home(self):
158                             self.root.destroy()
159
160                         #----- Main Class Calling -----
161                         if __name__ == "__main__":
162                             root = Tk()
163                             obj = Face_Recognition(root)
164                             root.mainloop()
165
166                         #----- Main Class -----
167                         class Face_Recognition:
168                             def __init__(self,root):
169                                 self.root = root
170
171                                 #----- Function Declaration -----
172
173                                 #----- Mark Attendance To CSV And Send Confirmation Message -----
174                                 def mark_attendance(self,sid,status,name,email,phone):
175                                     with open('attendance.csv','a+',newline='') as f: # Create File Manually For 1st Time
176                                         my_data_list = []
177                                         f.readline()
178                                         f.readline()
179                                         name_list = []
180                                         for line in f.readlines():
181                                             entry = line.split(',')
182                                             if((sid not in entry) and (status not in name_list) and (name not in name_list)):
183                                                 dt = datetime.now().strftime("%d/%m/%Y")
184                                                 dt_string = str(dt).split("-")
185                                                 f.write(f'{(sid),(status),(name),(dt),(dt_string)},')
186                                                 message = name + " is present at the time : "+dt_string+" on the day : "+dt
187                                                 msg = EmailMessage()
188                                                 msg['Subject'] = "Attendance"
189                                                 msg['To'] = email
190                                                 msg['From'] = "karlojoseph5@gmail.com"
191                                                 server = smtplib.SMTP('smtp.gmail.com:587')
192                                                 server.starttls()
193                                                 server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
194                                                 server.send_message(msg)
195                                                 server.quit()
196
197                                                 #----- Send SMC Confirmation Message -----
198                                                 def send_mail(self,message,email):
199                                                     msg = EmailMessage()
200                                                     msg['Subject'] = "Attendance"
201                                                     msg['To'] = email
202                                                     msg['From'] = "karlojoseph5@gmail.com"
203                                                     server = smtplib.SMTP('smtp.gmail.com:587')
204                                                     server.starttls()
205                                                     server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
206                                                     server.send_message(msg)
207                                                     server.quit()
208
209                                                 #----- Send SMC Confirmation Message -----
210                                                 def send_mark(self,message,phone):
211                                                     account_sid = "AC51a1001a14df0d995a993273d3004" # Twilio Account SID
212                                                     auth_token = "f5e0a2a2a2a2a2a2a2a2a2a2a2a2a2a2" # Twilio Account Authentication Token
213                                                     client = Client(account_sid, auth_token) # Create A Twilio Client
214                                                     twilio_phone_number = "+12502864843" # Your Twilio Phone Number (You Must Have Purchased This Number On Hello)
215                                                     recipient_phone_number = "+919876543210" # Recipient's Phone Number (In E.164 Format Including Country Code, e.g., +919876543210)
216                                                     client.messages.create(body = message, from_ = twilio_phone_number, to = recipient_phone_number) # Send The SMS
217
218                                                 #----- Face Recognition -----
219                                                 if confidence > 77 and str(status) == "Current Date":
220                                                     cv2.putText(img,str(status),(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(100,100,0),3)
221                                                     cv2.putText(img,"Id : (Individual Id)",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
222                                                     cv2.putText(img,"Name : (Name)",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
223                                                     if c == 1:
224                                                         collection = db1.student_details
225                                                         collection.update_one({"student_id":individual_id}, {"$set": {"attendance": "present"}})
226                                                     elif c == 2:
227                                                         collection = db1.teacher_details
228                                                         collection.update_one {"teacher_id":individual_id}, {"$set": {"attendance": "present"}}
229                                                     elif c == 3:
230                                                         collection = db1.staff_details
231                                                         collection.update_one {"staff_id":individual_id}, {"$set": {"attendance": "present"}}
232
233                                                 cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
234                                                 cv2.putText(img,"Unknown Face Detected", (x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
235                                                 sound = gts.gTTS("Unknown Face Detected",lang = "en")
236                                                 sound.save("recognize.mp3")
237                                                 os.system("recognize.mp3")
238
239                                                 #----- Back To Home Function -----
240                                                 def back_to_home(self):
241                                                     self.root.destroy()
242
243                                                 #----- Main Class -----
244                                                 if __name__ == "__main__":
245                                                     root = Tk()
246                                                     obj = Face_Recognition(root)
247                                                     root.mainloop()
248
249                                                 #----- Main Class -----
250                                                 class Face_Recognition:
251                                                     def __init__(self,root):
252                                                         self.root = root
253
254                                                         #----- Function Declaration -----
255
256                                                         #----- Mark Attendance To CSV And Send Confirmation Message -----
257                                                         def mark_attendance(self,sid,status,name,email,phone):
258                                                             with open('attendance.csv','a+',newline='') as f: # Create File Manually For 1st Time
259                                                                 my_data_list = []
260                                                                 f.readline()
261                                                                 f.readline()
262                                                                 name_list = []
263                                                                 for line in f.readlines():
264                                                                     entry = line.split(',')
265                                                                     if((sid not in entry) and (status not in name_list) and (name not in name_list)):
266                                                                         dt = datetime.now().strftime("%d/%m/%Y")
267                                                                         dt_string = str(dt).split("-")
268                                                                         f.write(f'{(sid),(status),(name),(dt),(dt_string)},')
269                                                                         message = name + " is present at the time : "+dt_string+" on the day : "+dt
270                                                                         msg = EmailMessage()
271                                                                         msg['Subject'] = "Attendance"
272                                                                         msg['To'] = email
273                                                                         msg['From'] = "karlojoseph5@gmail.com"
274                                                                         server = smtplib.SMTP('smtp.gmail.com:587')
275                                                                         server.starttls()
276                                                                         server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
277                                                                         server.send_message(msg)
278                                                                         server.quit()
279
280                                                                         #----- Send SMC Confirmation Message -----
281                                                                         def send_mail(self,message,email):
282                                                                             msg = EmailMessage()
283                                                                             msg['Subject'] = "Attendance"
284                                                                             msg['To'] = email
285                                                                             msg['From'] = "karlojoseph5@gmail.com"
286                                                                             server = smtplib.SMTP('smtp.gmail.com:587')
287                                                                             server.starttls()
288                                                                             server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
289                                                                             server.send_message(msg)
290                                                                             server.quit()
291
292                                                                         #----- Send SMC Confirmation Message -----
293                                                                         def send_mark(self,message,phone):
294                                                                             account_sid = "AC51a1001a14df0d995a993273d3004" # Twilio Account SID
295                                                                             auth_token = "f5e0a2a2a2a2a2a2a2a2a2a2a2a2a2a2" # Twilio Account Authentication Token
296                                                                             client = Client(account_sid, auth_token) # Create A Twilio Client
297                                                                             twilio_phone_number = "+12502864843" # Your Twilio Phone Number (You Must Have Purchased This Number On Hello)
298                                                                             recipient_phone_number = "+919876543210" # Recipient's Phone Number (In E.164 Format Including Country Code, e.g., +919876543210)
299                                                                             client.messages.create(body = message, from_ = twilio_phone_number, to = recipient_phone_number) # Send The SMS
300
301                                                                         #----- Face Recognition -----
302                                                                         if confidence > 77 and str(status) == "Current Date":
303                                                                             cv2.putText(img,str(status),(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(100,100,0),3)
304                                                                             cv2.putText(img,"Id : (Individual Id)",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
305                                                                             cv2.putText(img,"Name : (Name)",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
306                                                                             if c == 1:
307                                                                                 collection = db1.student_details
308                                                                                 collection.update_one {"student_id":individual_id}, {"$set": {"attendance": "present"}}
309                                                                             elif c == 2:
310                                                                                 collection = db1.teacher_details
311                                                                                 collection.update_one {"teacher_id":individual_id}, {"$set": {"attendance": "present"}}
312                                                                             elif c == 3:
313                                                                                 collection = db1.staff_details
314                                                                                 collection.update_one {"staff_id":individual_id}, {"$set": {"attendance": "present"}}
315
316                                                                             cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
317                                                                             cv2.putText(img,"Unknown Face Detected", (x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
318                                                                             sound = gts.gTTS("Unknown Face Detected",lang = "en")
319                                                                             sound.save("recognize.mp3")
320                                                                             os.system("recognize.mp3")
321
322                                                                             #----- Back To Home Function -----
323                                                                             def back_to_home(self):
324                                                                                 self.root.destroy()
325
326                                                                             #----- Main Class -----
327                                                                             if __name__ == "__main__":
328                                                                                 root = Tk()
329                                                                                 obj = Face_Recognition(root)
330                                                                                 root.mainloop()
331
332                                                                             #----- Main Class -----
333                                                                             class Face_Recognition:
334                                                                                 def __init__(self,root):
335                                                                 self.root = root
336
337                                                                                 #----- Function Declaration -----
338
339                                                                                 #----- Mark Attendance To CSV And Send Confirmation Message -----
340                                                                                 def mark_attendance(self,sid,status,name,email,phone):
341                                                                 with open('attendance.csv','a+',newline='') as f: # Create File Manually For 1st Time
342                                                                 my_data_list = []
343                                                                 f.readline()
344                                                                 f.readline()
345                                                                 name_list = []
346                                                                 for line in f.readlines():
347                                                                     entry = line.split(',')
348                                                                     if((sid not in entry) and (status not in name_list) and (name not in name_list)):
349                                                                         dt = datetime.now().strftime("%d/%m/%Y")
350                                                                         dt_string = str(dt).split("-")
351                                                                         f.write(f'{(sid),(status),(name),(dt),(dt_string)},')
352                                                                         message = name + " is present at the time : "+dt_string+" on the day : "+dt
353                                                                         msg = EmailMessage()
354                                                                         msg['Subject'] = "Attendance"
355                                                                         msg['To'] = email
356                                                                         msg['From'] = "karlojoseph5@gmail.com"
357                                                                         server = smtplib.SMTP('smtp.gmail.com:587')
358                                                                         server.starttls()
359                                                                         server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
360                                                                         server.send_message(msg)
361                                                                         server.quit()
362
363                                                                         #----- Send SMC Confirmation Message -----
364                                                                         def send_mail(self,message,email):
365                                                                             msg = EmailMessage()
366                                                                             msg['Subject'] = "Attendance"
367                                                                             msg['To'] = email
368                                                                             msg['From'] = "karlojoseph5@gmail.com"
369                                                                             server = smtplib.SMTP('smtp.gmail.com:587')
370                                                                             server.starttls()
371                                                                             server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
372                                                                             server.send_message(msg)
373                                                                             server.quit()
374
375                                                                         #----- Send SMC Confirmation Message -----
376                                                                         def send_mark(self,message,phone):
377                                                                             account_sid = "AC51a1001a14df0d995a993273d3004" # Twilio Account SID
378                                                                             auth_token = "f5e0a2a2a2a2a2a2a2a2a2a2a2a2a2a2" # Twilio Account Authentication Token
379                                                                             client = Client(account_sid, auth_token) # Create A Twilio Client
380                                                                             twilio_phone_number = "+12502864843" # Your Twilio Phone Number (You Must Have Purchased This Number On Hello)
381                                                                             recipient_phone_number = "+919876543210" # Recipient's Phone Number (In E.164 Format Including Country Code, e.g., +919876543210)
382                                                                             client.messages.create(body = message, from_ = twilio_phone_number, to = recipient_phone_number) # Send The SMS
383
384                                                                         #----- Face Recognition -----
385                                                                         if confidence > 77 and str(status) == "Current Date":
386                                                                             cv2.putText(img,str(status),(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(100,100,0),3)
387                                                                             cv2.putText(img,"Id : (Individual Id)",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
388                                                                             cv2.putText(img,"Name : (Name)",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
389                                                                             if c == 1:
390                                                                                 collection = db1.student_details
391                                                                                 collection.update_one {"student_id":individual_id}, {"$set": {"attendance": "present"}}
392                                                                             elif c == 2:
393                                                                                 collection = db1.teacher_details
394                                                                                 collection.update_one {"teacher_id":individual_id}, {"$set": {"attendance": "present"}}
395                                                                             elif c == 3:
396                                                                                 collection = db1.staff_details
397                                                                                 collection.update_one {"staff_id":individual_id}, {"$set": {"attendance": "present"}}
398
399                                                                             cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
400                                                                             cv2.putText(img,"Unknown Face Detected", (x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
401                                                                             sound = gts.gTTS("Unknown Face Detected",lang = "en")
402                                                                             sound.save("recognize.mp3")
403                                                                             os.system("recognize.mp3")
404
405                                                                             #----- Back To Home Function -----
406                                                                             def back_to_home(self):
407                                                                                 self.root.destroy()
408
409                                                                             #----- Main Class -----
410                                                                             if __name__ == "__main__":
411                                                                                 root = Tk()
412                                                                                 obj = Face_Recognition(root)
413                                                                                 root.mainloop()
414
415                                                                             #----- Main Class -----
416                                                                             class Face_Recognition:
417                                                                                 def __init__(self,root):
418                                                                 self.root = root
419
420                                                                                 #----- Function Declaration -----
421
422                                                                                 #----- Mark Attendance To CSV And Send Confirmation Message -----
423                                                                                 def mark_attendance(self,sid,status,name,email,phone):
424                                                                 with open('attendance.csv','a+',newline='') as f: # Create File Manually For 1st Time
425                                                                 my_data_list = []
426                                                                 f.readline()
427                                                                 f.readline()
428                                                                 name_list = []
429                                                                 for line in f.readlines():
430                                                                     entry = line.split(',')
431                                                                     if((sid not in entry) and (status not in name_list) and (name not in name_list)):
432                                                                         dt = datetime.now().strftime("%d/%m/%Y")
433                                                                         dt_string = str(dt).split("-")
434                                                                         f.write(f'{(sid),(status),(name),(dt),(dt_string)},')
435                                                                         message = name + " is present at the time : "+dt_string+" on the day : "+dt
436                                                                         msg = EmailMessage()
437                                                                         msg['Subject'] = "Attendance"
438                                                                         msg['To'] = email
439                                                                         msg['From'] = "karlojoseph5@gmail.com"
440                                                                         server = smtplib.SMTP('smtp.gmail.com:587')
441                                                                         server.starttls()
442                                                                         server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
443                                                                         server.send_message(msg)
444                                                                         server.quit()
445
446                                                                         #----- Send SMC Confirmation Message -----
447                                                                         def send_mail(self,message,email):
448                                                                             msg = EmailMessage()
449                                                                             msg['Subject'] = "Attendance"
450                                                                             msg['To'] = email
451                                                                             msg['From'] = "karlojoseph5@gmail.com"
452                                                                             server = smtplib.SMTP('smtp.gmail.com:587')
453                                                                             server.starttls()
454                                                                             server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
455                                                                             server.send_message(msg)
456                                                                             server.quit()
457
458                                                                         #----- Send SMC Confirmation Message -----
459                                                                         def send_mark(self,message,phone):
460                                                                             account_sid = "AC51a1001a14df0d995a993273d3004" # Twilio Account SID
461                                                                             auth_token = "f5e0a2a2a2a2a2a2a2a2a2a2a2a2a2a2" # Twilio Account Authentication Token
462                                                                             client = Client(account_sid, auth_token) # Create A Twilio Client
463                                                                             twilio_phone_number = "+12502864843" # Your Twilio Phone Number (You Must Have Purchased This Number On Hello)
464                                                                             recipient_phone_number = "+919876543210" # Recipient's Phone Number (In E.164 Format Including Country Code, e.g., +919876543210)
465                                                                             client.messages.create(body = message, from_ = twilio_phone_number, to = recipient_phone_number) # Send The SMS
466
467                                                                         #----- Face Recognition -----
468                                                                         if confidence > 77 and str(status) == "Current Date":
469                                                                             cv2.putText(img,str(status),(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(100,100,0),3)
470                                                                             cv2.putText(img,"Id : (Individual Id)",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
471                                                                             cv2.putText(img,"Name : (Name)",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
472                                                                             if c == 1:
473                                                                                 collection = db1.student_details
474                                                                                 collection.update_one {"student_id":individual_id}, {"$set": {"attendance": "present"}}
475                                                                             elif c == 2:
476                                                                                 collection = db1.teacher_details
477                                                                                 collection.update_one {"teacher_id":individual_id}, {"$set": {"attendance": "present"}}
478                                                                             elif c == 3:
479                                                                                 collection = db1.staff_details
480                                                                                 collection.update_one {"staff_id":individual_id}, {"$set": {"attendance": "present"}}
481
482                                                                             cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
483                                                                             cv2.putText(img,"Unknown Face Detected", (x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
484                                                                             sound = gts.gTTS("Unknown Face Detected",lang = "en")
485                                                                             sound.save("recognize.mp3")
486                                                                             os.system("recognize.mp3")
487
488                                                                             #----- Back To Home Function -----
489                                                                             def back_to_home(self):
490                                                                                 self.root.destroy()
491
492                                                                             #----- Main Class -----
493                                                                             if __name__ == "__main__":
494                                                                                 root = Tk()
495                                                                                 obj = Face_Recognition(root)
496                                                                                 root.mainloop()
497
498                                                                             #----- Main Class -----
499                                                                             class Face_Recognition:
500                                                                                 def __init__(self,root):
501                                                                 self.root = root
502
503                                                                                 #----- Function Declaration -----
504
505                                                                                 #----- Mark Attendance To CSV And Send Confirmation Message -----
506                                                                                 def mark_attendance(self,sid,status,name,email,phone):
507                                                                 with open('attendance.csv','a+',newline='') as f: # Create File Manually For 1st Time
508                                                                 my_data_list = []
509                                                                 f.readline()
510                                                                 f.readline()
511                                                                 name_list = []
512                                                                 for line in f.readlines():
513                                                                     entry = line.split(',')
514                                                                     if((sid not in entry) and (status not in name_list) and (name not in name_list)):
515                                                                         dt = datetime.now().strftime("%d/%m/%Y")
516                                                                         dt_string = str(dt).split("-")
517                                                                         f.write(f'{(sid),(status),(name),(dt),(dt_string)},')
518                                                                         message = name + " is present at the time : "+dt_string+" on the day : "+dt
519                                                                         msg = EmailMessage()
520                                                                         msg['Subject'] = "Attendance"
521                                                                         msg['To'] = email
522                                                                         msg['From'] = "karlojoseph5@gmail.com"
523                                                                         server = smtplib.SMTP('smtp.gmail.com:587')
524                                                                         server.starttls()
525                                                                         server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
526                                                                         server.send_message(msg)
527                                                                         server.quit()
528
529                                                                         #----- Send SMC Confirmation Message -----
530                                                                         def send_mail(self,message,email):
531                                                                             msg = EmailMessage()
532                                                                             msg['Subject'] = "Attendance"
533                                                                             msg['To'] = email
534                                                                             msg['From'] = "karlojoseph5@gmail.com"
535                                                                             server = smtplib.SMTP('smtp.gmail.com:587')
536                                                                             server.starttls()
537                                                                             server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
538                                                                             server.send_message(msg)
539                                                                             server.quit()
540
541                                                                         #----- Send SMC Confirmation Message -----
542                                                                         def send_mark(self,message,phone):
543                                                                             account_sid = "AC51a1001a14df0d995a993273d3004" # Twilio Account SID
544                                                                             auth_token = "f5e0a2a2a2a2a2a2a2a2a2a2a2a2a2a2" # Twilio Account Authentication Token
545                                                                             client = Client(account_sid, auth_token) # Create A Twilio Client
546                                                                             twilio_phone_number = "+12502864843" # Your Twilio Phone Number (You Must Have Purchased This Number On Hello)
547                                                                             recipient_phone_number = "+919876543210" # Recipient's Phone Number (In E.164 Format Including Country Code, e.g., +919876543210)
548                                                                             client.messages.create(body = message, from_ = twilio_phone_number, to = recipient_phone_number) # Send The SMS
549
550                                                                         #----- Face Recognition -----
551                                                                         if confidence > 77 and str(status) == "Current Date":
552                                                                             cv2.putText(img,str(status),(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(100,100,0),3)
553                                                                             cv2.putText(img,"Id : (Individual Id)",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
554                                                                             cv2.putText(img,"Name : (Name)",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
555                                                                             if c == 1:
556                                                                                 collection = db1.student_details
557                                                                                 collection.update_one {"student_id":individual_id}, {"$set": {"attendance": "present"}}
558                                                                             elif c == 2:
559                                                                                 collection = db1.teacher_details
560                                                                                 collection.update_one {"teacher_id":individual_id}, {"$set": {"attendance": "present"}}
561                                                                             elif c == 3:
562                                                                                 collection = db1.staff_details
563                                                                                 collection.update_one {"staff_id":individual_id}, {"$set": {"attendance": "present"}}
564
565                                                                             cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
566                                                                             cv2.putText(img,"Unknown Face Detected", (x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
567                                                                             sound = gts.gTTS("Unknown Face Detected",lang = "en")
568                                                                             sound.save("recognize.mp3")
569                                                                             os.system("recognize.mp3")
570
571                                                                             #----- Back To Home Function -----
572                                                                             def back_to_home(self):
573                                                                                 self.root.destroy()
574
575                                                                             #----- Main Class -----
576                                                                             if __name__ == "__main__":
577                                                                                 root = Tk()
578                                                                                 obj = Face_Recognition(root)
579                                                                                 root.mainloop()
580
581                                                                             #----- Main Class -----
582                                                                             class Face_Recognition:
583                                                                                 def __init__(self,root):
584                                                                 self.root = root
585
586                                                                                 #----- Function Declaration -----
587
588                                                                                 #----- Mark Attendance To CSV And Send Confirmation Message -----
589                                                                                 def mark_attendance(self,sid,status,name,email,phone):
590                                                                 with open('attendance.csv','a+',newline='') as f: # Create File Manually For 1st Time
591                                                                 my_data_list = []
592                                                                 f.readline()
593                                                                 f.readline()
594                                                                 name_list = []
595                                                                 for line in f.readlines():
596                                                                     entry = line.split(',')
597                                                                     if((sid not in entry) and (status not in name_list) and (name not in name_list)):
598                                                                         dt = datetime.now().strftime("%d/%m/%Y")
599                                                                         dt_string = str(dt).split("-")
600                                                                         f.write(f'{(sid),(status),(name),(dt),(dt_string)},')
601                                                                         message = name + " is present at the time : "+dt_string+" on the day : "+dt
602                                                                         msg = EmailMessage()
603                                                                         msg['Subject'] = "Attendance"
604                                                                         msg['To'] = email
605                                                                         msg['From'] = "karlojoseph5@gmail.com"
606                                                                         server = smtplib.SMTP('smtp.gmail.com:587')
607                                                                         server.starttls()
608                                                                         server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
609                                                                         server.send_message(msg)
610                                                                         server.quit()
611
612                                                                         #----- Send SMC Confirmation Message -----
613                                                                         def send_mail(self,message,email):
614                                                                             msg = EmailMessage()
615                                                                             msg['Subject'] = "Attendance"
616                                                                             msg['To'] = email
617                                                                             msg['From'] = "karlojoseph5@gmail.com"
618                                                                             server = smtplib.SMTP('smtp.gmail.com:587')
619                                                                             server.starttls()
620                                                                             server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
621                                                                             server.send_message(msg)
622                                                                             server.quit()
623
624                                                                         #----- Send SMC Confirmation Message -----
625                                                                         def send_mark(self,message,phone):
626                                                                             account_sid = "AC51a1001a14df0d995a993273d3004" # Twilio Account SID
627                                                                             auth_token = "f5e0a2a2a2a2a2a2a2a2a2a2a2a2a2a2" # Twilio Account Authentication Token
628                                                                             client = Client(account_sid, auth_token) # Create A Twilio Client
629                                                                             twilio_phone_number = "+12502864843" # Your Twilio Phone Number (You Must Have Purchased This Number On Hello)
630                                                                             recipient_phone_number = "+919876543210" # Recipient's Phone Number (In E.164 Format Including Country Code, e.g., +919876543210)
631                                                                             client.messages.create(body = message, from_ = twilio_phone_number, to = recipient_phone_number) # Send The SMS
632
633                                                                         #----- Face Recognition -----
634                                                                         if confidence > 77 and str(status) == "Current Date":
635                                                                             cv2.putText(img,str(status),(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(100,100,0),3)
636                                                                             cv2.putText(img,"Id : (Individual Id)",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
637                                                                             cv2.putText(img,"Name : (Name)",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
638                                                                             if c == 1:
639                                                                                 collection = db1.student_details
640                                                                                 collection.update_one {"student_id":individual_id}, {"$set": {"attendance": "present"}}
641                                                                             elif c == 2:
642                                                                                 collection = db1.teacher_details
643                                                                                 collection.update_one {"teacher_id":individual_id}, {"$set": {"attendance": "present"}}
644                                                                             elif c == 3:
645                                                                                 collection = db1.staff_details
646                                                                                 collection.update_one {"staff_id":individual_id}, {"$set": {"attendance": "present"}}
647
648                                                                             cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
649                                                                             cv2.putText(img,"Unknown Face Detected", (x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
650                                                                             sound = gts.gTTS("Unknown Face Detected",lang = "en")
651                                                                             sound.save("recognize.mp3")
652                                                                             os.system("recognize.mp3")
653
654                                                                             #----- Back To Home Function -----
655                                                                             def back_to_home(self):
656                                                                                 self.root.destroy()
657
658                                                                             #----- Main Class -----
659                                                                             if __name__ == "__main__":
660                                                                                 root = Tk()
661                                                                                 obj = Face_Recognition(root)
662                                                                                 root.mainloop()
663
664                                                                             #----- Main Class -----
665                                                                             class Face_Recognition:
666                                                                                 def __init__(self,root):
667                                                                 self.root = root
668
669                                                                                 #----- Function Declaration -----
670
671                                                                                 #----- Mark Attendance To CSV And Send Confirmation Message -----
672                                                                                 def mark_attendance(self,sid,status,name,email,phone):
673                                                                 with open('attendance.csv','a+',newline='') as f: # Create File Manually For 1st Time
674                                                                 my_data_list = []
675                                                                 f.readline()
676                                                                 f.readline()
677                                                                 name_list = []
678                                                                 for line in f.readlines():
679                                                                     entry = line.split(',')
680                                                                     if((sid not in entry) and (status not in name_list) and (name not in name_list)):
681                                                                         dt = datetime.now().strftime("%d/%m/%Y")
682                                                                         dt_string = str(dt).split("-")
683                                                                         f.write(f'{(sid),(status),(name),(dt),(dt_string)},')
684                                                                         message = name + " is present at the time : "+dt_string+" on the day : "+dt
685                                                                         msg = EmailMessage()
686                                                                         msg['Subject'] = "Attendance"
687                                                                         msg['To'] = email
688                                                                         msg['From'] = "karlojoseph5@gmail.com"
689                                                                         server = smtplib.SMTP('smtp.gmail.com:587')
690                                                                         server.starttls()
691                                                                         server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
692                                                                         server.send_message(msg)
693                                                                         server.quit()
694
695                                                                         #----- Send SMC Confirmation Message -----
696                                                                         def send_mail(self,message,email):
697                                                                             msg = EmailMessage()
698                                                                             msg['Subject'] = "Attendance"
699                                                                             msg['To'] = email
700                                                                             msg['From'] = "karlojoseph5@gmail.com"
701                                                                             server = smtplib.SMTP('smtp.gmail.com:587')
702                                                                             server.starttls()
703                                                                             server.login("karlojoseph5@gmail.com","vaimbyujbxmpau") # App Passwords
704                                                                             server.send_message(msg)
705                                                                             server.quit()
706
707                                                                         #----- Send SMC Confirmation Message -----
708                                                                         def send_mark(self,message,phone):
709                                                                             account_sid = "AC51a1001a14df0d995a993273d3004" # Twilio Account SID
710                                                                             auth_token = "f5e0a2a2a2a2a2a2a2a2a2a2a2a2a2a2" # Twilio Account Authentication Token
711                                                                             client = Client(account_sid, auth_token) # Create A Twilio Client
712                                                                             twilio_phone_number = "+12502864843" # Your Twilio Phone Number (You Must Have Purchased This Number On Hello)
713                                                                             recipient_phone_number = "+919876543210" # Recipient's Phone Number (In E.164 Format Including Country Code, e.g., +919876543210)
714                                                                             client.messages.create(body = message, from_ = twilio_phone_number, to = recipient_phone_number) # Send The SMS
715
716                                                                         #----- Face Recognition -----
717                                                                         if confidence > 77 and str(status) == "Current Date":
718                                                                             cv2.putText(img,str(status),(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(100,100,0),3)
719                                                                             cv2.putText(img,"Id : (Individual Id)",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
720                                                                             cv2.putText(img,"Name : (Name)",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,0),3)
721                                                                             if c == 1:
722                                                                                 collection = db1.student_details
723                                                                                 collection.update_one {"student_id":individual_id}, {"$set": {"attendance": "present"}}
724                                                                             elif c == 2:
725                                                                                 collection = db1.teacher_details
726                                                                                 collection.update_one {"teacher_id":individual_id}, {"$set": {"attendance": "present"}}
727                                                                             elif c == 3:
728                                                                                 collection = db1.staff_details
729                                                                                 collection.update_one {"staff_id":individual_id}, {"$set": {"attendance": "present"}}
730
731                                                                             cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
732                                                                             cv2.putText(img,"Unknown Face Detected", (x,y-55),cv2.FONT_HERSHEY_COMPLEX
```



Three screenshots of the Sublime Text editor showing the student.py file from the Face\_Recognition\_Attendance\_Project. The code is a Python script with several sections of logic:

- Function Declaration:** A large block of code starting with `def validate\_fields(self):` which handles validation for phone number, email, and course selection.
- Add Data to Database:** A section starting with `def add\_data(self):` which adds student details to a MongoDB database. It includes logic for creating unique indices and handling successful insertion.
- Fetch Data From Database:** A section starting with `def fetch\_data(self):` which retrieves data from the MongoDB database.
- Face Recognition:** A section starting with `def face\_recognition(self):` which uses OpenCV's Haar Cascade classifier to detect faces in captured images. It includes image cropping, scaling, and saving to a specified path.
- Back Home Function:** A section starting with `def back\_to\_home(self):` which returns the user to the main menu.
- Main Class Calling:** The final section at the bottom of the code, starting with `if \_\_name\_\_ == '\_\_main\_\_':` which initializes the application.

**train.py**

```

1 # Required Libraries
2 # for GUI
3 import Tkinter
4 from PIL import Image,ImageTk
5 import cv2 # For Training Model With Photo Samples
6 from time import strftime # For Time Stamp
7 import os # For Creating Folders
8 import gtts # For Generating Invoice
9 import gTTS
10 import playsound
11 import numpy as np # It Gives The Performance Of More Than 88% While Converting In Array
12
13 # Main class
14 class Train:
15     def __init__(self,root):
16         self.root = root
17         self.root.title("Face Recognition System")
18
19         # First Top Image
20         img = Image.open(r"images\main\img1.jpg")
21         img_top = img.resize((150,150),Image.Resampling.LANCZOS)
22         f_lbl = Label(self.root,image=img_top)
23         f_lbl.place(x=0,y=0,width=150,height=150)
24
25         # Second Top Image
26         img1 = Image.open(r"images\main\img2.jpg")
27         img1 = img1.resize((150,150),Image.Resampling.LANCZOS)
28         f_lbl1 = Label(self.root,image=img1)
29         f_lbl1.place(x=0,y=0,width=150,height=150)
30
31         # Third Top Image
32         img2 = Image.open(r"images\main\img3.jpg")
33         img2 = img2.resize((150,150),Image.Resampling.LANCZOS)
34         f_lbl2 = Label(self.root,image=img2)
35         f_lbl2.place(x=0,y=0,width=150,height=150)
36
37         # Top Title Label
38         title_label = Label(self.root,text="TRAIN DATA SET",font="times new roman",35,"bold",bg="white",fg="red")
39         title_label.place(x=0,y=150,width=150,height=45)
40
41         # Back Home Function
42         back_button = Button(self.root,cursor="hand2",command=self.back_to_home,text="Home",width=10,font="times new roman",12,"bold",bg="green",fg="white",activebackground="black",activeforeground="white")
43         back_button.place(x=1400,y=150)
44
45         # To Display Time At Label
46         def time():
47             strTime = strftime("%H:%M %S %p")
48             lbl.config(text = strTime)
49             lbl.after(200,time)
50
51             #lbl.config(text = strTime)
52             #lbl.after(200,time)
53             #lbl.config(title_label,font="times new roman",12,"bold",background="white",foreground="black")
54             #lbl.config(text="Time",font="times new roman",12,"bold",background="white",foreground="black")
55
56         # ----- Function Declaration -----
57
58         # Background Image
59         img_top = Image.open(r"images\train\img1.jpg")
60         img_top_resized_top = img_top.resize((150,150),Image.Resampling.LANCZOS)
61         f_lbl_top = Label(self.root,image=img_top_resized_top)
62         f_lbl_top.place(x=0,y=175,width=150,height=150)
63
64         # Button For Train Data
65         train_data = Button(self.root,cursor="hand2",text="Train Data",width=10,font="times new roman",30,"bold",bg="blue",fg="white",activebackground="blue",activeforeground="white")
66         train_data.place(x=700,y=650,width=200,height=60)
67
68         # ----- Train The Classifiers -----
69
70         # Train The Classifiers
71         def train_classifier(self):
72             data_dir = "data"
73             path = [os.path.join(data_dir,file) for file in os.listdir(data_dir) ]
74             rpath = []
75             rids = []
76             for image in path:
77                 for id in image:
78                     rpath.append(image)
79                     rids.append(id)
80
81             faces = []
82             ids = []
83             for image in rpath:
84                 img = Image.open(image).convert('L') # Convert Into Grayscale
85                 image_np = np.array(img,uint8) # uint8 is Data Type
86                 id = int(os.path.split(image)[-1].split('_')[1])
87
88                 faces.append(image_np)
89                 ids.append(id)
90
91             # Train The Classifier And Save The Data
92             classifier = cv2.face.LBPHFaceRecognizer_create() # For This Error Try This Command 'pip install opencv-contrib-python' And If Open CV Is Not There Try 'pip install opencv'
93             classifier.train(faces,ids)
94             classifier.write("classifier.xml")
95             cv2.destroyAllWindows()
96             sound = gTTS.gTTs("Training Data set completed successfully",lang = "en")
97             sound.save("train.mp3")
98             os.system("mpg321 train.mp3")
99             if os.path.exists("train.mp3"):
100                 os.remove("train.mp3")
101             messagebox.showinfo("Result","Training Data set completed successfully",parent=self.root)
102
103             # ----- Back To Home Function -----
104             def back_to_home(self):
105                 self.root.destroy()
106
107             # ----- Main Class Calling -----
108             if __name__ == "__main__":
109                 root = Tk()
110                 obj = Train(root)
111                 root.mainloop()

```

**login.py**

```

199 if not bool(re.search("[A-Z]", password)): # Uppercase
200     message.append("Password Should Contain At Least One Uppercase Letter.")
201 if not bool(re.search("[0-9]", password)): # Digits
202     message.append("Password Should Contain At Least One Digit.")
203 if not bool(re.search("[!@#$%^&*()_+=-]", password)): # Special Characters
204     message.append("Password Should Contain At Least One Special Character.")
205
206 # To Check Password Strength Dynamically
207 if message:
208     pass_strength_label.config(text="Weak", fg="red")
209     elif len(password) < 12:
210         pass_strength_label.config(text="Moderate", fg="orange")
211     else:
212         pass_strength_label.config(text="Strong", fg="green")
213
214 # ----- Reset Password -----
215 def reset_pass(self):
216     if self.var_sec_ques.get() == "Select Question" or self.var_sec_ans.get() == "" or self.var_reset_pass.get() == "":
217         messagebox.showerror("Error","All Fields Are Required",parent=self.root)
218     else:
219         client = MongoClient('mongodb://localhost:27017/')
220         collection = client['face_recognition_system']['registration_details']
221         result = collection.find_one({'email':self.var_email.get(), 'security_question':self.var_sec_ques.get(), 'security_answer':self.var_sec_ans.get()}, {'_id': 0})
222         if not result:
223             messagebox.showerror("Error","Please Enter Valid Details",parent=self.root)
224         else:
225             messagebox.showerror("Error", "Email Already Exist",parent=self.root)
226
227         else:
228             collection.update_one({'email':self.var_email.get()}, {"$set": {"password":self.var_reset_pass.get()}})
229             sound = gTTS.gTTs("Password Reset Done Successfully",lang = "en")
230             sound.save("login.mp3")
231             playsound("login.mp3")
232             if os.path.exists("login.mp3"):
233                 os.remove("login.mp3")
234             messagebox.showinfo("Success","Password Reset Done Successfully",parent=self.root)
235             self.root.destroy()
236             client.close()
237
238 # ----- Open Register Window -----
239 def registration(self):
240     self.new_window = Toplevel(self.root)
241     self.app_Register_Window(self.new_window)
242
243 # ----- Open Admin Window -----
244 def admin(self):
245     self.new_window = Toplevel(self.root)
246     self.app_Admin(self.new_window)
247
248 # ----- Main Class Calling -----
249 if __name__ == "__main__":
250     root = Tk()
251     obj = login_window(root)
252     root.mainloop()

```

C:\Users\Shubham\Desktop\project\Face\_Recognition\_Attendance\_Project\student.py (Face\_Recognition\_Attendance\_Project) - Sublime Text (UNREGISTERED)

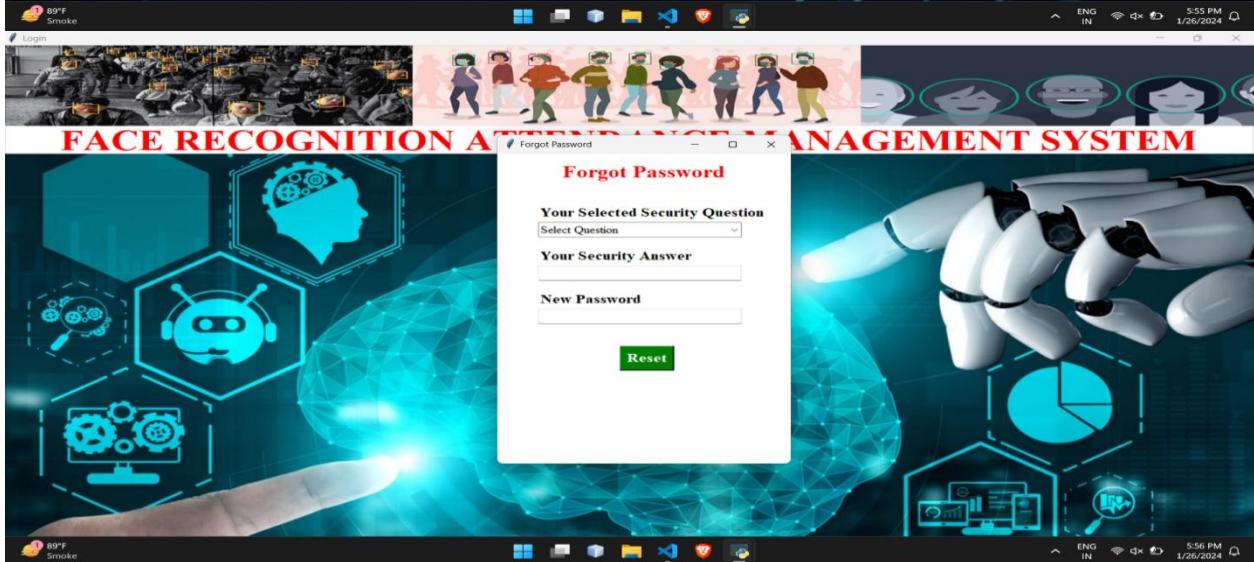
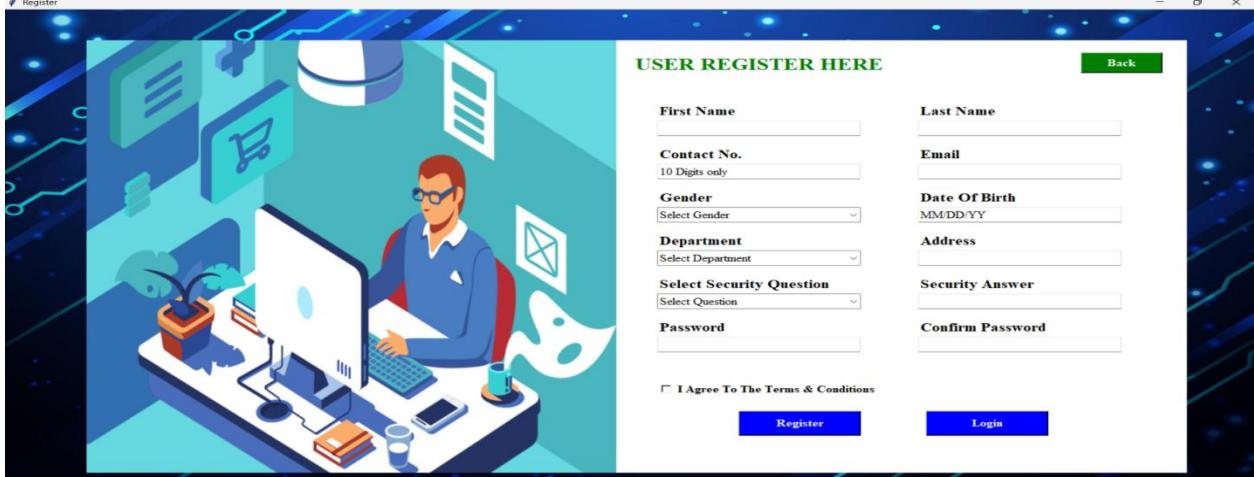
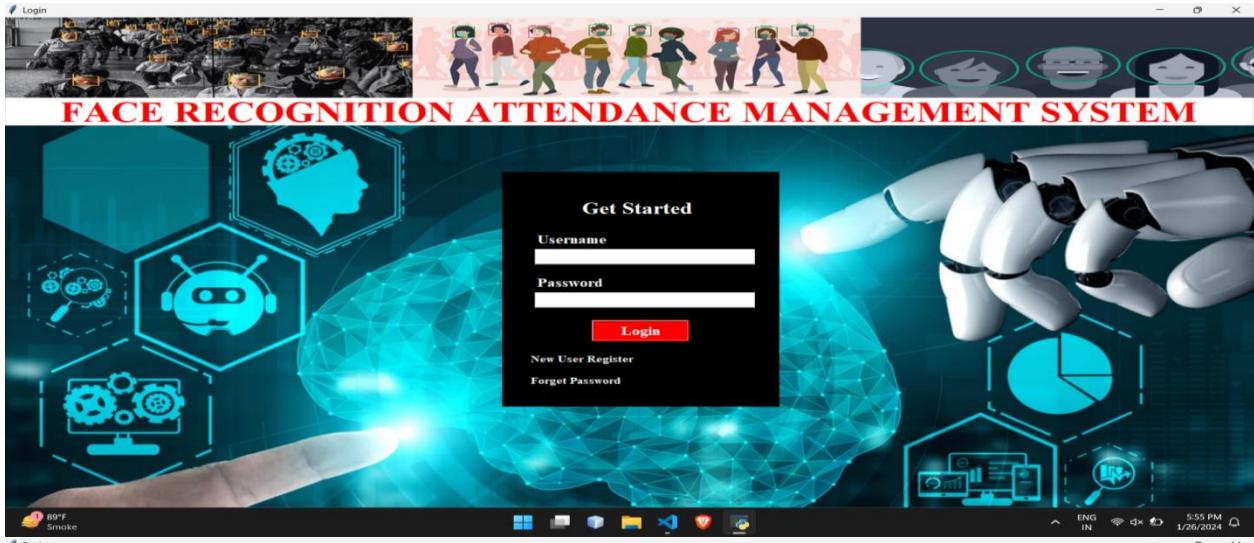
FOLDERS

```
1 Required Libraries
2
3 from Tkinter import *
4 from tkcalendar import *
5 from PIL import ImageTk
6 from tkinter import messagebox
7 from tkcalendar import DateEntry # For Date Of Birth To Open Calendar
8 from tkcalendar import DateEntry # For Date To Open For Customize The Selected Date
9 import re # To Match The Email
10 from time import strftime, localtime
11 import cv2 # For Taking Image Samples
12 import os # For OS Operations
13 import time
14 import gits
15 # For pip install pip install playsound==1.2.2
16 # For pip install pip install playsound==1.2.2
17 import pymongo
18 from pymongo import MongoClient
19
20 # Main Class
21 class Student:
22     def __init__(self,root):
23         self.root = root
24         self.root.geometry("1560x800+0+0")
25         self.root.title("Face Recognition System")
26
27         # Variables
28         self.var_name = StringVar()
29         self.var_course = StringVar()
30         self.var_year = StringVar()
31         self.var_dep = StringVar()
32         self.var_studentid = StringVar()
33         self.var_studentname = StringVar()
34         self.var_email = StringVar()
35         self.var_roll = StringVar()
36         self.var_gender = StringVar()
37         self.var_address = StringVar()
38         self.var_email = StringVar()
39         self.var_address = StringVar()
40         self.var_duration = StringVar()
41         self.var_search_combo = StringVar()
42         self.var_search_combo = StringVar()
43
44         # First Top Image
45         img = Image.open("images/mininimg1.jpg")
46         img = img.resize((550, 130), Image.ANTIALIAS)
47         self.photoing = ImageTk.PhotoImage(img)
48         r1lb = Label(self.root,image=self.photoing)
49         r1lb.place(x=10,y=50,width=500,height=130)
50
51         # Second Top Image
52         img1 = Image.open("images/mininimg2.jpg")
53         img1 = img1.resize((550, 130), Image.ANTIALIAS)
54         self.photoing1 = ImageTk.PhotoImage(img1)
55         r2lb = Label(self.root,image=self.photoing1)
56         r2lb.place(x=550,y=50,width=500,height=130)
57
58         # Function Declaration
59         # ***** Validation On Data Fields *****
60         def validate_fields(self):
61             if self.var_name.get() == "" or self.var_email.get() == "" or self.var_roll.get() == "" or self.var_dep.get() == "" or self.var_year.get() == "" or self.var_course.get() == "" or self.var_studentid.get() == "":
62                 return False
63             else:
64                 return True
65
66         # ***** Add The Data To Database *****
67         def add_data(self):
68             if self.validate_fields():
69                 if self.var_dep.get() == "Select Department" or self.var_course.get() == "Select Course" or self.var_year.get() == "Select Year" or self.var_semester.get() == "Select Semester" or self.var_email.get() == "Enter A Valid Email Address":
70                     messagebox.showerror("Error", "All Fields Are Required", parent=self.root)
71                 else:
72                     if self.validate_fields():
73                         try:
74                             client = MongoClient('mongodb://localhost:27017/')
75                             collection = client['face_recognition_system'][self.student_details]
76                             collection.insert_one({'student_id':self.var_studentid.get(), 'dept':self.var_dep.get(), 'course':self.var_course.get(), 'year':self.var_year.get(), 'semester':self.var_semester.get()})
77                             sound = gits.gits("Student details has been added successfully",lang = "en")
78                             playsound.playsound("student.mp3")
79                             if os.path.exists("student.mp3"):
80                                 os.remove("student.mp3")
81                             messagebox.showinfo("Success", "Student Details Has Been Added Successfully", parent=self.root)
82                             self.fetch_data()
83                         except Exception as es:
84                             messagebox.showerror("Error", f"Due To :{str(es)}", parent=self.root)
85
86             # ***** Fetch The Data From Database *****
87             def fetch_data(self):
88                 self.var_search.set("")
89                 self.var_search_combo.set("Select")
90                 client = MongoClient('mongodb://localhost:27017/')
91                 collection = client['face_recognition_system'][self.student_details]
92                 collection.update_many({}, {"$set": {"student_id":self.var_studentid.get(), "dept":self.var_dep.get(), "course":self.var_course.get(), "year":self.var_year.get(), "semester":self.var_semester.get()}})
93                 client.close()
94                 self.fetch_data()
95
96                 # Face Classifier
97                 face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml') # Load Predefined Data On Face Frontals From OpenCV For Object Detection
98
99                 def face_cropped(img): # Cropping The Image In Gray Scale / Resize The Image
100                     gray = cv2.cvtColor(img, cv2.COLOR_BGRGRAY) # Converting The Img To Grey Scale
101                     gray = cv2.resize(gray,(500,500)) # Resizing The Image To Specific Resolution(500,500) # Scaling Factor = 1.3(0.9 Default) Minimum Neighbour = 5
102                     faces = face_classifier.detectMultiScale(gray, 1.3, 5)
103                     for(x,y,w,h) in faces:
104                         cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
105                         cropped_face = img[y:y+h,x:x+w]
106                         return img[y:y+h,x:x+w]
107
108                 cap = cv2.VideoCapture(0) # Camera Captured
109                 img = ImageTk.PhotoImage(Image.new("RGB", (640, 480), "white"))
110                 while True:
111                     ret,my_frame=cap.read()
112                     if my_frame==None:
113                         print("my frame==None")
114                         break
115                     img_id=ImageTk.PhotoImage(my_frame)
116                     if img_id!=None:
117                         Face = cv2.resize(face_cropped(my_frame), (450, 450))
118                         file_name = "Data\\Student_id_{}_{}_photosample_{}.jpg".format(self.var_studentid.get(),strftime("%d-%m-%Y"),img_id)
119                         cv2.imwrite(file_name,Face, (50, 50),cv2.FONT_HERSHEY_COMPLEX, 2, (255, 255, 255), 2)
120                         cv2.imshow("Cropped Face",Face)
121                         if cv2.waitKey(1) == 13 or int(img_id) == 100: # 13 Represents The User Has Hit The Enter Then The Capture Will Stop
122                             break
123
124                 cap.release()
125                 cv2.destroyAllWindows()
126
127                 # Invoice Code After Successful Completion Of Operation
128                 sound = gits.gits("Generation of data set is completed successfully",lang = "en")
129                 sound.get("student.mp3")
130                 playsound.playsound("student.mp3")
131                 if os.path.exists("student.mp3"):
132                     os.remove("student.mp3")
133
134                 messagebox.showinfo("Success", "Data Set Is Completed Successfully", parent=self.root)
135                 self.fetch_data()
136                 messagebox.showinfo("Success", "Data Set Is Completed Successfully", parent=self.root)
137
138             # ***** Back To Home Function *****
139             def back_to_home(self):
140                 self.root.destroy()
141
142             if __name__ == "__main__":
143                 root = Tk()
144                 root.mainloop()
```

C:\Users\Shubham\Desktop\project\Face\_Recognition\_Attendance\_Project\teacher.py (Face\_Recognition\_Attendance\_Project) - Sublime Text (UNREGISTERED)

```
Required Libraries
1   | Required Libraries
2   | For Tkinter
3   | from tkinter import*
4   | from tkinter import ttk
5   | from tkinter import messagebox
6   | from tkcalendar import*
7   | from tkcalendar import DateEntry # For Date of Birth To Open Calendar
8   | from tkcalendar import DateEntry # For Duration To Customize The Selected Date
9   | import re # To Match The Email
10  | import os
11  | import pyaudio
12  | from pymongo import MongoClient
13  | import cv2, numpy, ImageTk
14  | import os # For OS Operations
15  | from time import strftime # For Time Stamp
16  | if __name__ == "__main__":
17      import gtt
18      import playsound
19
20  # Main Class
21  class teacherDetails:
22      def __init__(self,root):
23          self.root = root
24          self.root.iconbitmap(r"1560x824+0+0")
25          self.root.title("Face Recognition System")
26
27  # Variables
28  self.var_teacherid = StringVar()
29  self.var_teachermane = StringVar()
30  self.var_qualification = StringVar()
31  self.var_address = StringVar()
32  self.var_duration = StringVar()
33  self.var_gender = StringVar()
34  self.var_email = StringVar()
35  self.var_phone = StringVar()
36  self.var_addresss = StringVar()
37  self.var_experience = StringVar()
38  self.var_poto = StringVar()
39  self.var_search = StringVar()
40  self.var_search_combo = StringVar()
41
42
43  # First Top Image
44  img = Image.open("images/mainimg1.jpg")
45  img = img.resize((510,180),Image.Resampling.LANCZOS)
46  self.photolink = ImageTk.PhotoImage(img)
47  f_l1=Label(self.root,image=self.photolink)
48  f_l1.place(x=0,y=0,width=510,height=180)
49
50
51  # Second Top Image
52  img1 = Image.open("images/mainimg2.jpg")
53  img1 = img1.resize((550,180),Image.Resampling.LANCZOS)
54  self.photolink1 = ImageTk.PhotoImage(img1)
55  f_l2=Label(self.root,image=self.photolink1)
56  f_l2.place(x=0,y=0,width=550,height=180)
57
58
59  # Function Declaration
60  def validate_data(self):
61      # ***** Validation On Data Fields *****
62      if self.validate_fields():
63          def validate_phone_number():
64              return len(phone_number) == 10 and phone_number.isdigit()
65
66          def validate_email(email):
67              return re.match(r'^[a-zA-Z0-9_.-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$', email) # Basic Email Validation Using Regular Expression
68
69          if not validate_phone_number(self.var_phone.get()):
70              messagebox.showerror("Error", "Phone Number Must Be Of 10 Digits",parent=self.root) # Display An Error Message For Invalid Phone Number
71              return 0
72
73          if not validate_email(self.var_email.get()):
74              messagebox.showerror("Error", "Enter A Valid Email Address",parent=self.root) # Display An Error Message For Invalid Email
75              return 0
76
77          return 1
78
79
80  # ***** Add The Data To Database *****
81  def add_data(self):
82      if self.var_teacherid.get() == "Digits only" or self.var_teachermane.get() == "" or self.var_duration.get() == "MMYY - MMYY" or self.var_qualification.get() == "Select Qualification":
83          messagebox.showerror("Error", "All Fields Are Required",parent=self.root)
84      else:
85          if self.validate_fields():
86              try:
87                  client = MongoClient("mongodb://localhost:27017/")
88                  collection = client['face_recognition_system']['teacher_details']
89                  collection.create_index([('teacher_id', pymongo.ASCENDING)], unique=True) # Create a Unique Index On a Field
90                  collection.update_one({'teacher_id':self.var_teacherid.get(), 'teacher_name':self.var_teachermane.get(), 'qualification':self.var_qualification.get(), 'department':self.var_address.get()})
91
92                  # Invoice code After Successful Completion of Operation
93                  total = len(collection.find())
94                  sound.saver("teacher.mp3")
95                  playsound("teacher.mp3")
96                  os.remove("teacher.mp3")
97                  message.showinfo("Success", "Teacher Details Has Been Added Successfully",parent=self.root)
98                  self.fetch_data()
99                  self.reset_data()
100                 except Exception as e:
101                     messagebox.showerror("Error", "Due To :{0},parent={1}",parent=self.root)
102
103
104  # ***** Fetch The Data From Database *****
105  def fetch_data(self):
106      if self.var_search.get() == "Select":
107          self.var_search_combo.set("Select")
108          client = MongoClient("mongodb://localhost:27017/")
109          collection = client['face_recognition_system']['teacher_details']
110          result = collection.find({}).sort('_id', 0)
111          values = []
112
113          for document in result:
114              values.append(document)
115
116          self.teacher_table.delete(*self.teacher_table.get_children())
117          for key in document:
118              values.list.append(document[key])
119          self.teacher_table.insert(-1,values=values.list)
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
857
858
859
859
860
861
862
863
864
865
866
867
867
868
869
869
870
871
872
873
873
874
875
875
876
877
877
878
879
879
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
```

# System Screenshots



**Face Recognition System**

17:57:03 PM FACE RECOGNITION ATTENDANCE SYSTEM SOFTWARE

89°F Smoke

17:58:50 PM STUDENT DETAILS

**Student Academic Details**

<b>Current Course Information</b>	
Department : Select Department	Course : Select Course
Year : Select Year	Semester : Select Semester
<b>Class Student Information</b>	
Student ID : Digits only	Student Name :
Class Division : Select Division	Roll Number :
Gender : Select Gender	Date Of Birth : MM/DD/YY
Email :	Phone Number : 10 Digits only
Address :	Duration : MMYY - MMYY
<input type="radio"/> Take Photo Sample <input type="radio"/> No Photo Sample	
<a href="#">Save</a> <a href="#">Update</a> <a href="#">Delete</a> <a href="#">Reset</a>	
<a href="#">Take/Update Photo Sample</a>	

**Student Details**

**Search System**

Select     Search    Refresh

StudentID	Department	Course	Year	Semester	Name	Division

89°F Smoke

17:59:58 PM TEACHER DETAILS

**Teacher Details**

**Teacher Personal Information**

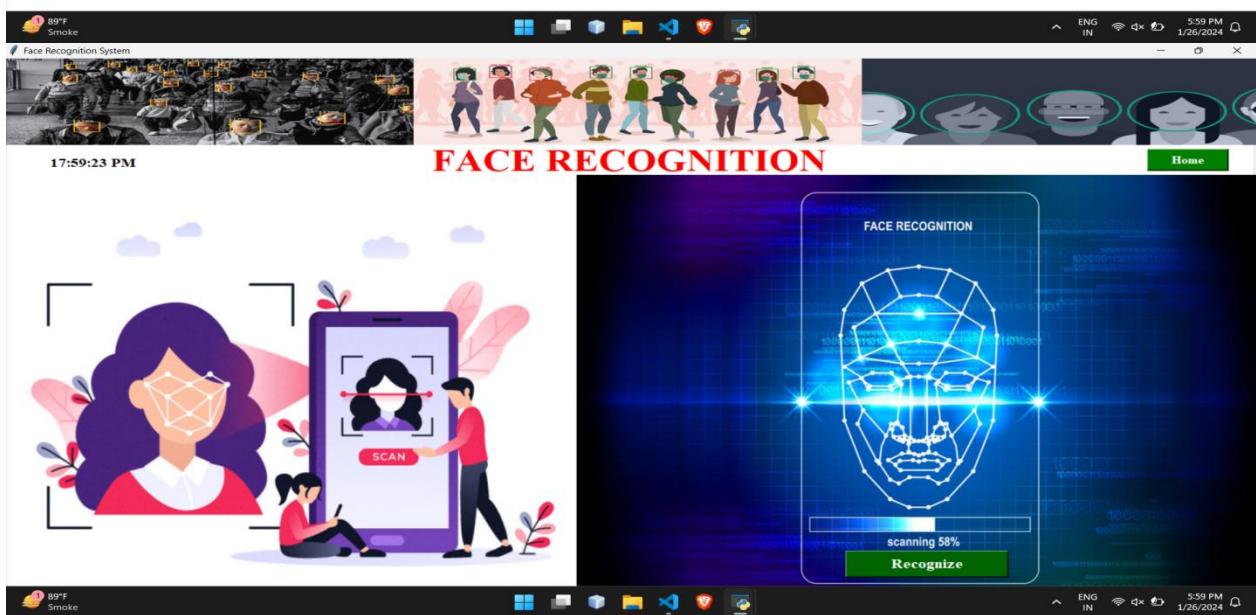
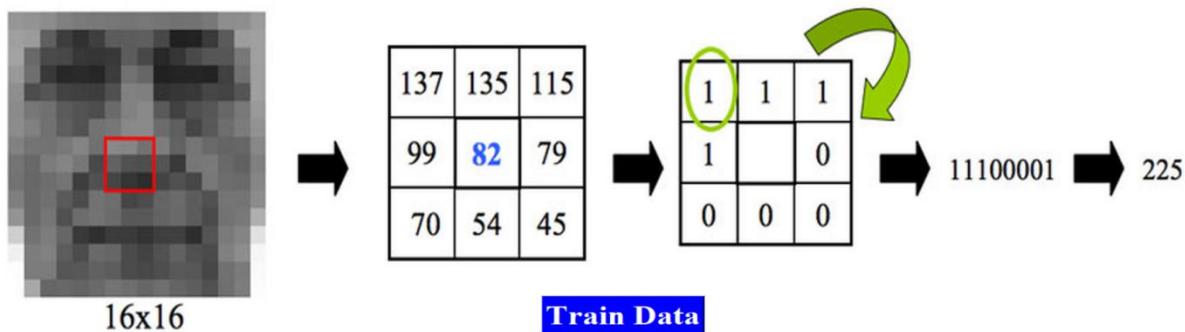
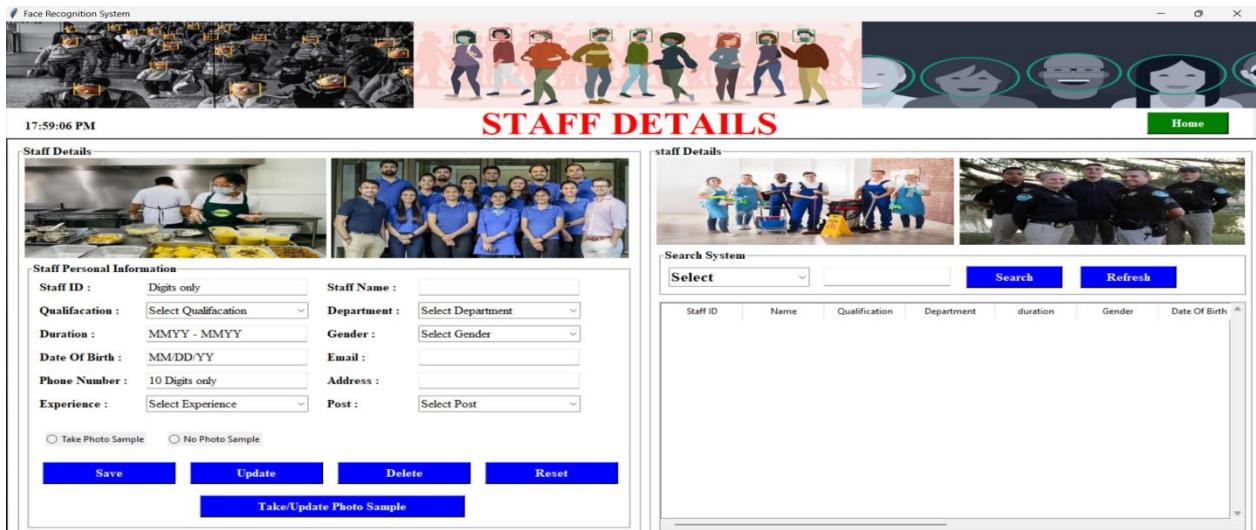
Teacher ID : Digits only	Teacher Name :
Qualification : Select Qualification	Department : Select Department
Duration : MMYY - MMYY	Gender : Select Gender
Date Of Birth : MM/DD/YY	Email :
Phone Number : 10 Digits only	Address :
Experience : Select Experience	Post : Select Post
<input type="radio"/> Take Photo Sample <input type="radio"/> No Photo Sample	
<a href="#">Save</a> <a href="#">Update</a> <a href="#">Delete</a> <a href="#">Reset</a>	
<a href="#">Take/Update Photo Sample</a>	

**Teacher Details**

**Search System**

Select     Search    Refresh

TeacherID	Name	Qualification	Department	duration	Gender	Date Of Birth



**ATTENDANCE MANAGEMENT SYSTEM**

Attendance Details

Attendance ID	Status	Name	Date	Time	Attendance

Attendance ID : Digits only  
Select Status : Status  
Name :  
Date :  
Time :  
Attendance : Select Status

Import CSV  
Export CSV  
Update  
Reset

**ADMIN PANEL**

User Details

First Name	
Last Name	
Contact No.	10 Digits only
Email	
Gender	Select Gender
Date Of Birth	MM/DD/YY
Address	
Department	Select Department
Security Question	Select Security Question
Security Answer	
Password	

Save      Update  
Delete      Reset

User Data

Search System

Select		Search	Refresh	Absent Message	Reset Data				
First_Name	Last_Name	Contact_Number	Department	Gender	DateOfBirth	Email	Address	Security_Question	Security_An

**STUDENT DATA**

StudentID	Department	Course	Year	Semester	Name	Division	Roll No	Gender	Date Of Birth	Email	Phone	Address	Duration
12	Computer Science	BSC	2020-21	Semester-1	shubam	A	aa	Male	4/13/23	bhaleraoshubham2371@gmail.com	7039718959	sscds	Mar 2024 - Sep 20 Nc

# Scope for Future Enhancement

I plan to improve the model by incorporating advancements in machine learning and data analysis. Certainly! FaceTrek Attendance - Face recognition Attendance System have a wide range of applications and continuous enhancements can lead to further improvements in performance, accuracy, and usability. I will attempt to incorporate the following future scope areas for improving FaceTrek Attendance.

- **Improved Accuracy and Reliability:** My goal is to refine the algorithm to make face recognition more accurate in difficult situations including dim lighting, occlusion, changing expressions on the face, and changing postures.
- **Privacy and Security:** To guarantee that facial data is safely maintained and utilized only for approved purposes, I will create strong privacy-preserving methods. Incorporate cutting-edge encryption techniques to safeguard facial data both in transit and storage.
- **Real-time Processing:** My goal is to enhance hardware and algorithms for real-time face identification in high-definition video streams. Creating effective parallel processing strategies to manage complex, real-time face recognition workloads.
- **Ethical Considerations:** I will discuss the moral issues of bias, equity, and transparency in FaceTrek Attendance, with a focus on potential abuse and demographic inequalities. Establish auditing and accountability procedures to guarantee that FaceTrek Attendance is utilized in an ethical and responsible manner.
- **Adaptability and Robustness:** I'll create robust algorithms that are able to learn and adjust over time, accounting for things like changing hairstyles, aging-related changes in face appearance, and more. By utilizing adversarial training and anti-spoofing strategies, I can improve resilience against spoofing attacks and manipulations.
- **User Experience:** By creating user-friendly interfaces for face enrollment, authentication, and management, I will enhance the user experience. Improve accessibility features so that people with a range of abilities and preferences can use FaceTrek Attendance.
- **Scalability and Deployment:** To manage workloads and resource needs, I will investigate cloud-based options for distributed FaceTrek Attendance processing.
- **Continuous Learning and Feedback:** In order to gather user input and performance information for continuous system improvement, I will put feedback systems in place. Allowing FaceTrek Attendance to adjust and improve its behavior in response to feedback and interactions from the real world by incorporating reinforcement learning techniques.

My FaceTrek Attendance - Face recognition Attendance System can be made more accurate, dependable, secure, and morally sound by concentrating on these areas of improvement, which will create new opportunities for the broad application.

# Bibliography

**References:**

- <https://stackoverflow.com/>
- <https://www.w3schools.com/>
- <https://opencv.org/>
- <https://docs.python.org/3/library/tkinter.html>
- <https://www.twilio.com/docs>
- <https://docs.python.org/3/library/csv.html>
- <https://gtts.readthedocs.io/en/latest/index.html>
- <https://www.mongodb.com/docs>
- <https://docs.python.org/3/library/smtplib.html>
- <https://docs.python.org/3/library/shutil.html>
- <https://pythonbasics.org/python-play-sound/>
- <https://dutypar.com/blogs/how-does-face-recognition-attendance-system-work/>
- <https://www.jibble.io/article/face-recognition-attendance>