COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

COCHIN UNIVERSITY COLLEGE OF ENGINEERING KUTTANAD

A MINI-PROJECT REPORT ON

“ Attendance System Using Face Recognition ”

Submitted by :-

EKLAVYA KUMAR TRIPATHI (20221563)

Department of Computer Science & Engineering

2023-2024

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

COCHIN UNIVERSITY COLLEGE OF ENGINEERING KUTTANAD

Department of Computer Science & Engineering

CERTIFICATE

Certified that the mini project work entitled Attendance System Using Face Recognition is a bonafide work carried out by EKLAVYA KUMAR TRIPATHI (20221563).

The report has been approved as it satisfies the academic requirements in respect to mini-project work prescribed for the course.

Mrs. Bindu P. K
Mini-Project Coordinator

Mrs. Alice Joseph
Mini-Project Coordinator

Dr. Preetha Mathew
Head of the Department

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to Mrs. Bindu P K and Mrs. Alice Joseph ma’am for their guidance, constant supervision, and encouragement. Efforts have been taken by using this project however it would not have been possible without the kind of support and help of many individuals and organizations.

Thanks to our respected principal, Dr. Joseph Kutty sir, for the facilities provided by him during the preparation of this report. We also express our gratitude towards all the staff members of the Computer Science and Engineering department and faculties of CUCEK for their guidance and assistance. Their expertise and insights have greatly enriched our project.

We would also like to extend our sincere gratitude to Dr. Preetha Mathew, Head of the Department for giving us innovative suggestions, timely advice, correction, and suggestions during this endeavor.

Abstract:

Face Recognition Attendance System (FRAS) is an innovative solution aimed at automating attendance tracking in educational institutions and workplaces. This paper presents the design and implementation of FRAS, leveraging cutting-edge deep learning techniques in computer vision. The system employs advanced algorithms for face detection and recognition, allowing it to accurately identify individuals in real-time from images or live video streams.

Key features of FRAS include robustness to varying environmental conditions, such as lighting and facial expressions, ensuring reliable performance in diverse settings. Upon enrollment, individuals' facial features are extracted and stored securely in a database for subsequent recognition during attendance marking.

Through the integration of FRAS into existing attendance management systems, organizations can streamline the attendance tracking process, eliminate manual errors, and prevent fraudulent practices like buddy punching. Experimental results demonstrate the system's effectiveness in achieving high accuracy rates and scalability to accommodate large user populations.

Overall, FRAS offers a reliable, efficient, and secure solution for attendance management, contributing to improved productivity and accountability in educational and professional environments.

CONTENTS:

LIST OF FIGURES:

CHAPTER :1

INTRODUCTION

1. Introduction

This project involves building an attendance system which utilizes facial recognition to mark the presence, time-in, and time-out of Students. It covers areas such as facial detection, alignment, and recognition, along with the development of a web application to cater to various use cases of the system such as registration of new Students, addition of photos to the training dataset, viewing attendance reports, etc. This project intends to serve as an efficient substitute for traditional manual attendance systems. It can be used in corporate offices, schools, and organizations where security is essential.

1.1 Purpose

The purpose of this SRS document is to specify software requirements of the Attendance Management System Using Face Recognition. It is intended to be a complete specification of what functionality the Attendance Management System provides.
This project aims to automate the traditional attendance system where the attendance is marked manually. It also enables an organization to maintain its records like in-time, out time, break time and attendance digitally. Digitalization of the system would also help in better visualization of the data using graphs to display the no. of Students present today, total work hours of each Student and their break time. Its added features serve as an efficient upgrade and replacement over the traditional attendance system.

1.2 Document Conventions

No specific user document conventions are used this time. All the sections and points are written in simpler words with utmost clarity.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, marketing stuff users, testers and documentation writers of the system.
Preference to read the document is in the sequence of table of contents only. Document is organized in a manner to understand the need and implementation details of the system.

1.4 Product Scope

Facial recognition is becoming more prominent in out society. It has made major progress in the field of security. It is a very effective tool that can help low enforcers to recognize criminals and software companies are leveraging the technology to help users access the technology. This technology can be further developed to be used in other avenues such as ATMs, accessing confidential files, or other sensitive materials.

This project serves as a foundation for future projects based on facial detection and recognition. This project also convers web development and database management with a user-friendly UI. Using this system any corporate offices, school and organization can replace their traditional way of maintaining attendance of the Students and can also generate their availability(presence) report throughout the month.

CHAPTER :2

REQUIREMENT ANALYSIS

2. REQUIREMENT ANALYSIS

This section contains all of the functional requirements, non-functional requirements and quality requirements of the system.

2.1 FUNCTIONAL REQUIREMENTS

A functional requirement defines function of a system or its component where a function is described as a specification of behaviour between outputs and inputs.

2.1.1 Handling Files

The system should be capable enough to handle multiple file Orders and multiple students also for different subjects.

2.1.2 Extracting Data

The system should be capable enough to extract the Data from student profile.

2.2 External Interface Requirements:

The following sections will introduce the numerous requirements of the system from the point of view of different users and will introduce a number of decisions that have been made regarding implementation.

2.2.1 User Interface

The user interface for this system will be simple and clear. Most importantly, the ages must be easy to read, easy to understand and accessible. The color scheme should be appropriate to provide familiarity with the university and there should be no contrast issues.

2.2.2 Hardware Interfaces

I3 Processor-based computer or higher

Memory: 3GB RAM

Hard drive

Working Web camera with clear image

2.3.3 Software Interfaces

Windows or Linux Operating System

Client-side Browser Support

2.3 Communications Interfaces

2.3.1Communication Standard: HTTPS
2.3.2Network Server: Localhost
2.3.3 Chrome

2.4 NON-FUNCTIONAL REQUIREMENTS

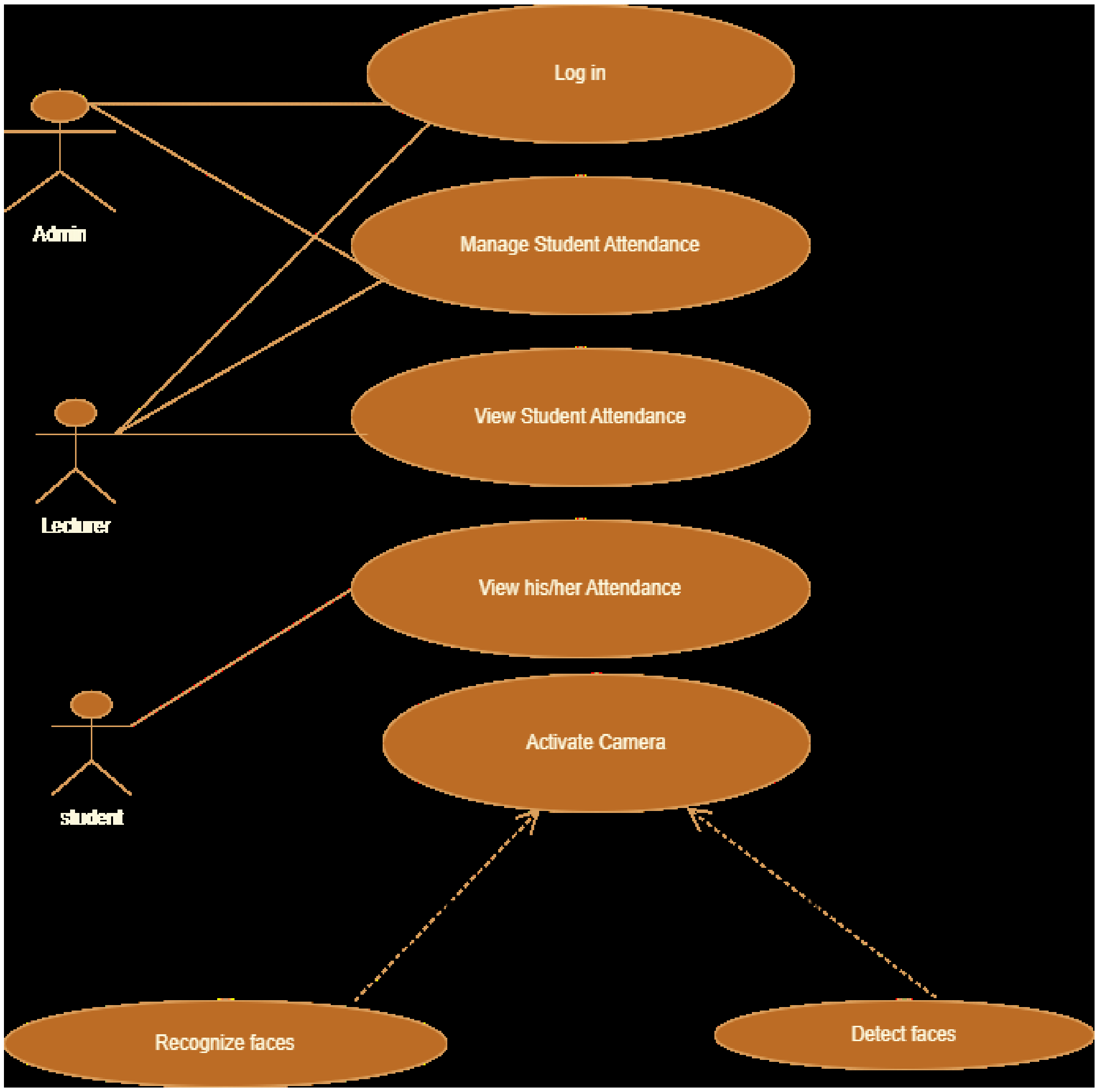
2.4.1 Performance

The performance of the system must be fast and accurate. The app must be light weight and must be able to send invitations and announcements to multiple students via mail without any lag or delay. Accessing and updating data through database must be fast and reliable. The system should
handle expected and unexpected errors in ways that prevent loss in information and long delay in acquiring information.

2.4.2 Safety and Security

The database may get crashed at any point in time due to virus or OS errors. Therefore it is required to take
backup of the database so that all information is not lost. The user should keep their username and password safe and should not share it with anyone. The app should use a secured database.

CHAPTER 3
SYSTEM DESIGN

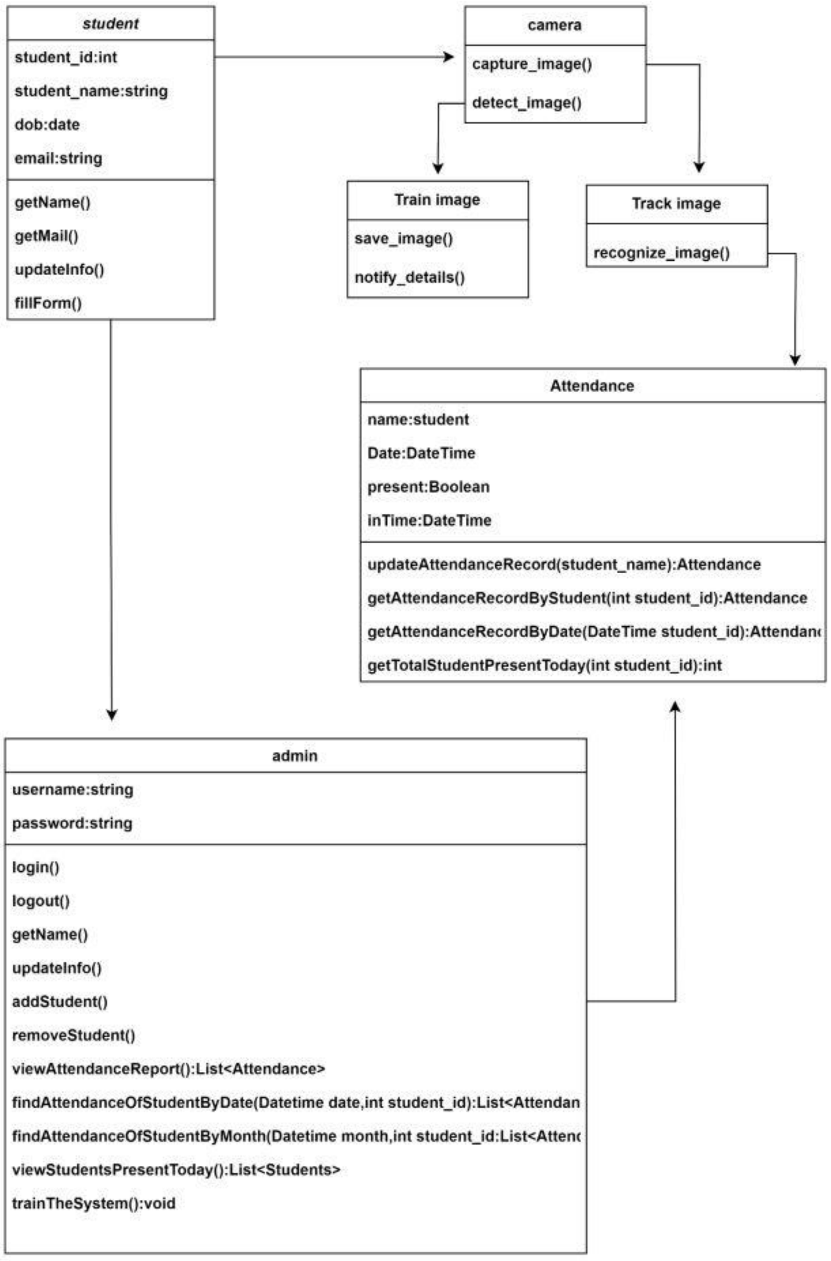


Use case diagram

1) A user (student, professor or admin) should be able to log in using their college credentials.
2) A student should be able to view his attendance for any class of any course he has taken.
3) A student should be able to view his attendance (0 or 1) for a class as soon as it ends.
4) A student should be able to see how many ”blocks” in a class he attended (as explained in the previous section) and whether this is above the threshold set by the professor, hence justifying the attendance obtained for that class.

5) A student should be able to view the total number of classes he has attended in a course so far, and the number of classes he is allowed to miss before his total attendance for that course drops below the course’s requirements.
6) A professor should be able to view the total attendance of a class as soon as it ends.
7) A professor should be able to change the threshold determining for how many blocks a student must be present in the class to get attendance. This can be changed for a specific class or for all classes in the course. This gives the professor the power to make attendance lenient or optional on a specific day, without having to go through the college administration.
8) A professor should be able to change the room number of the class before it starts, in case of any sudden events to ensure that attendance will still be taken by activating the camera in the new
9) An administrator should be able to directly change the attendance of a student in a particular class if required, overriding the automated attendance.
10) An administrator should be able to change the room number of the entire course, in case of any clashes with other classes or events.

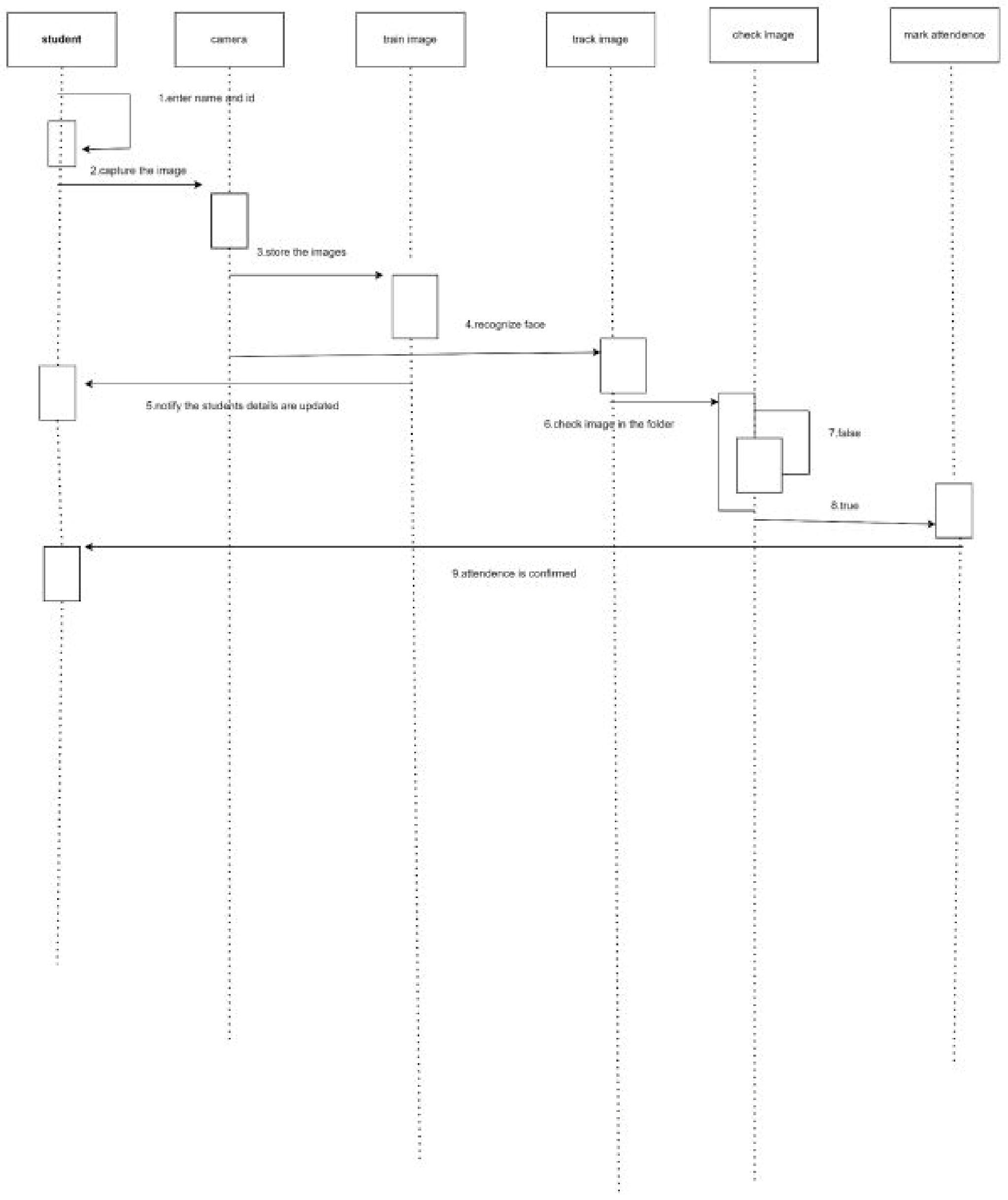
CLASSDIAGRAM



student
student_id:int
student_name:string
dob:date
email:string
getName()
getMail()
updateInfo()
fillForm()
camera
capture_image()
detect_image()
Train image
save_image()
notify_details()
Track image
recognize_image()
Attendance
name:student
Date:DateTime
present:Boolean
inTime:DateTime
updateAttendanceRecord(student_name):Attendance
getAttendanceRecordByStudent(int student_id):Attendance
getAttendanceRecordByDate(DateTime student_id):Attendanc
getTotalStudentPresentToday(int student_id):int
admin
username:string
password:string
login()
logout()
getName()
updateInfo()
addStudent()
removeStudent()
viewAttendanceReport():List<Attendance>
findAttendanceOfStudentByDate(Datetime date,int student_id):List<Attendan
findAttendanceOfStudentByMonth(Datetime month,int student_id:List<Attenc
viewStudentsPresentToday():List<Students>
trainTheSystem():void

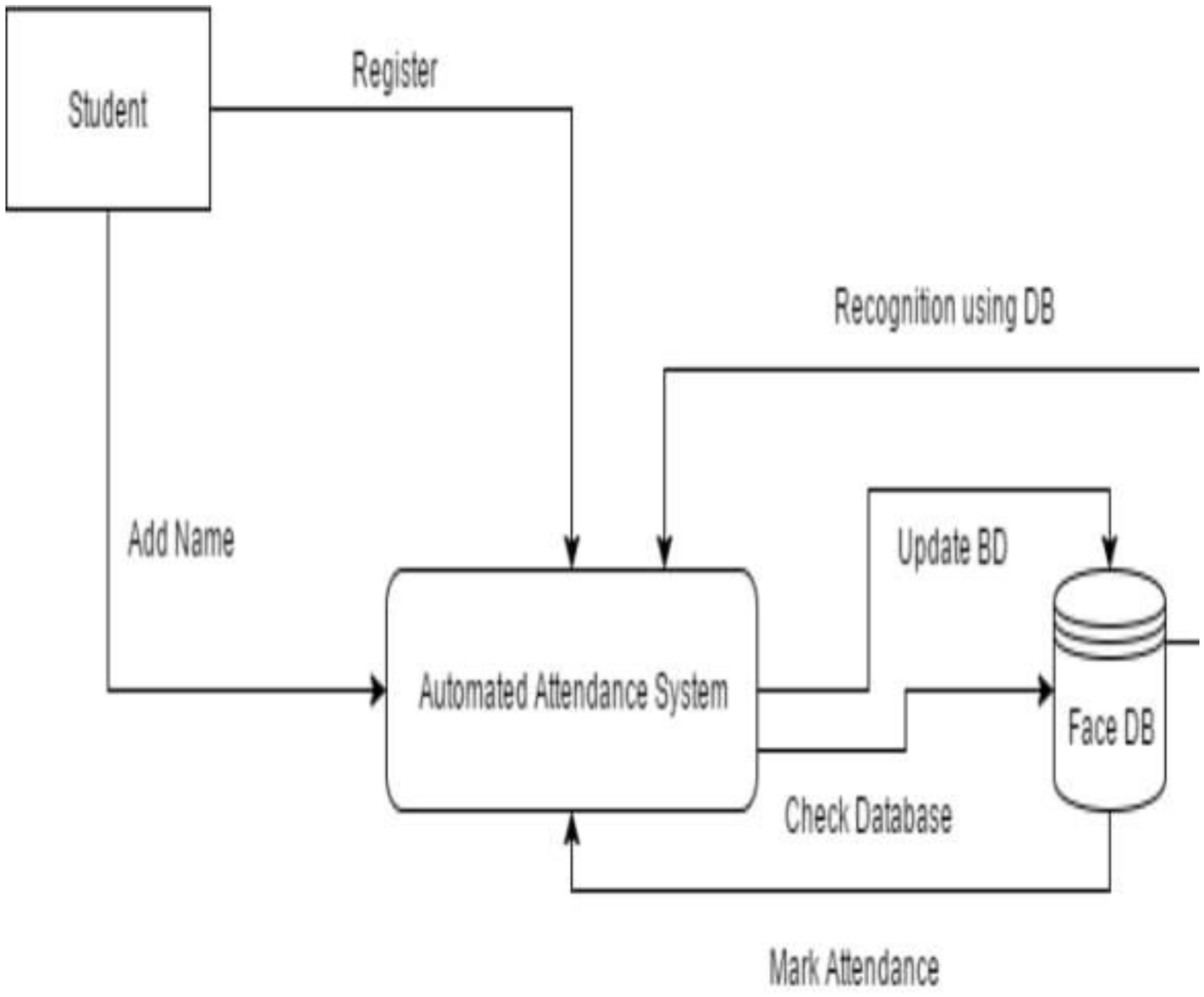
The class diagram describes the structure of the system by presenting classes, attributes, operations, and relationships between objects. The class diagram is the main block of object-oriented modeling and is mainly used for general conceptual modeling of application structure and for detailed model translation modeling in coding. The class diagram can also be used for data modeling.

SEQUENCE DIAGRAM

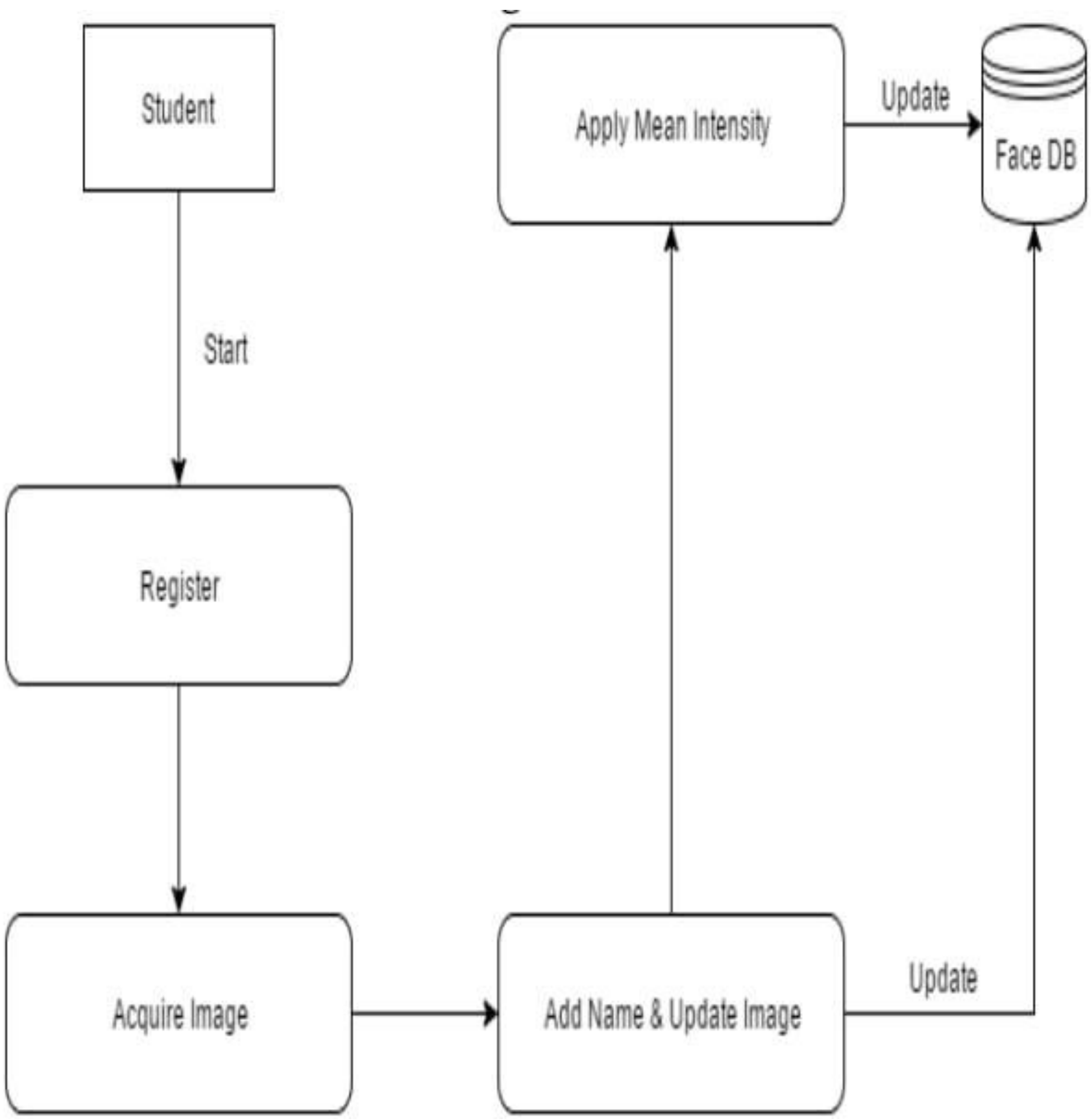


A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that Shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

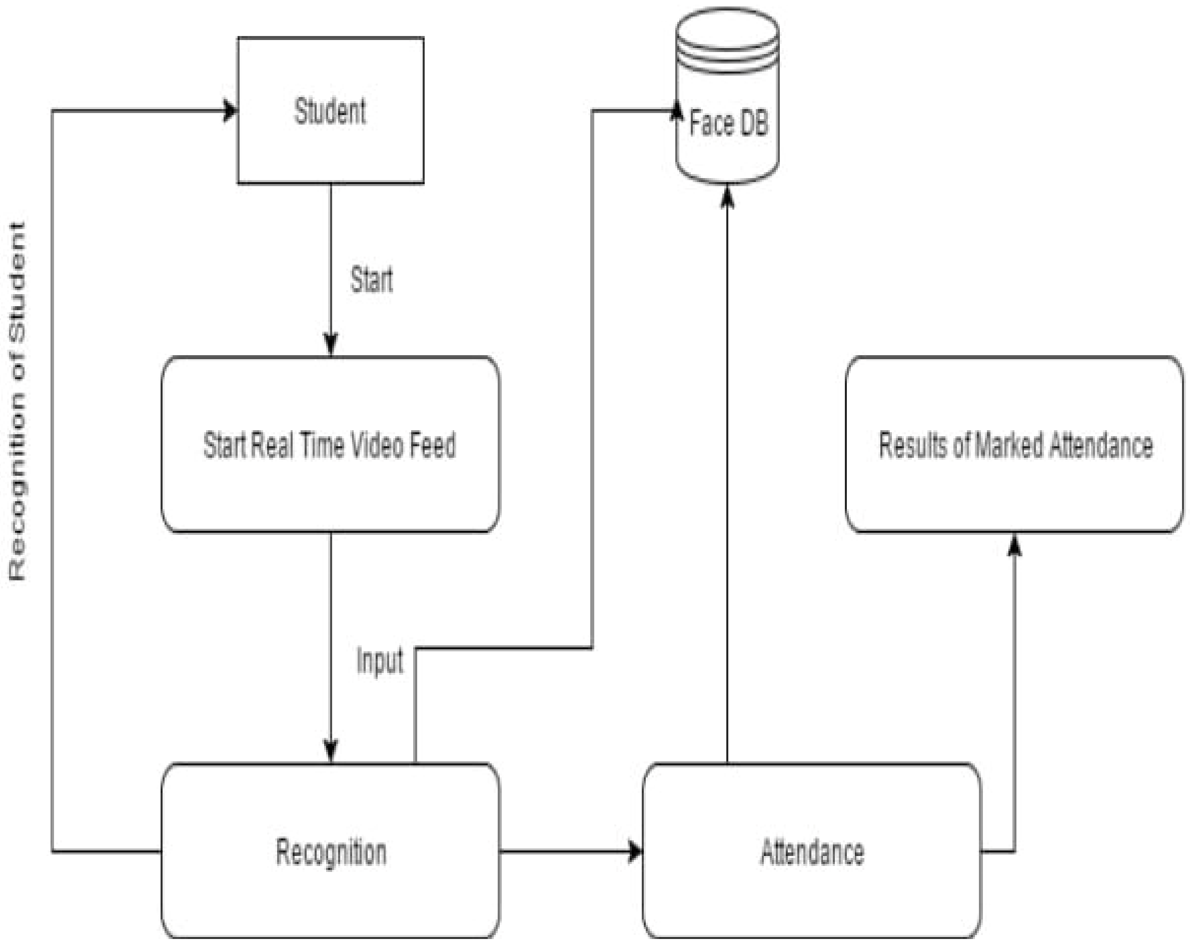
DATA FLOW DIAGRAM



DFD LEVEL-0

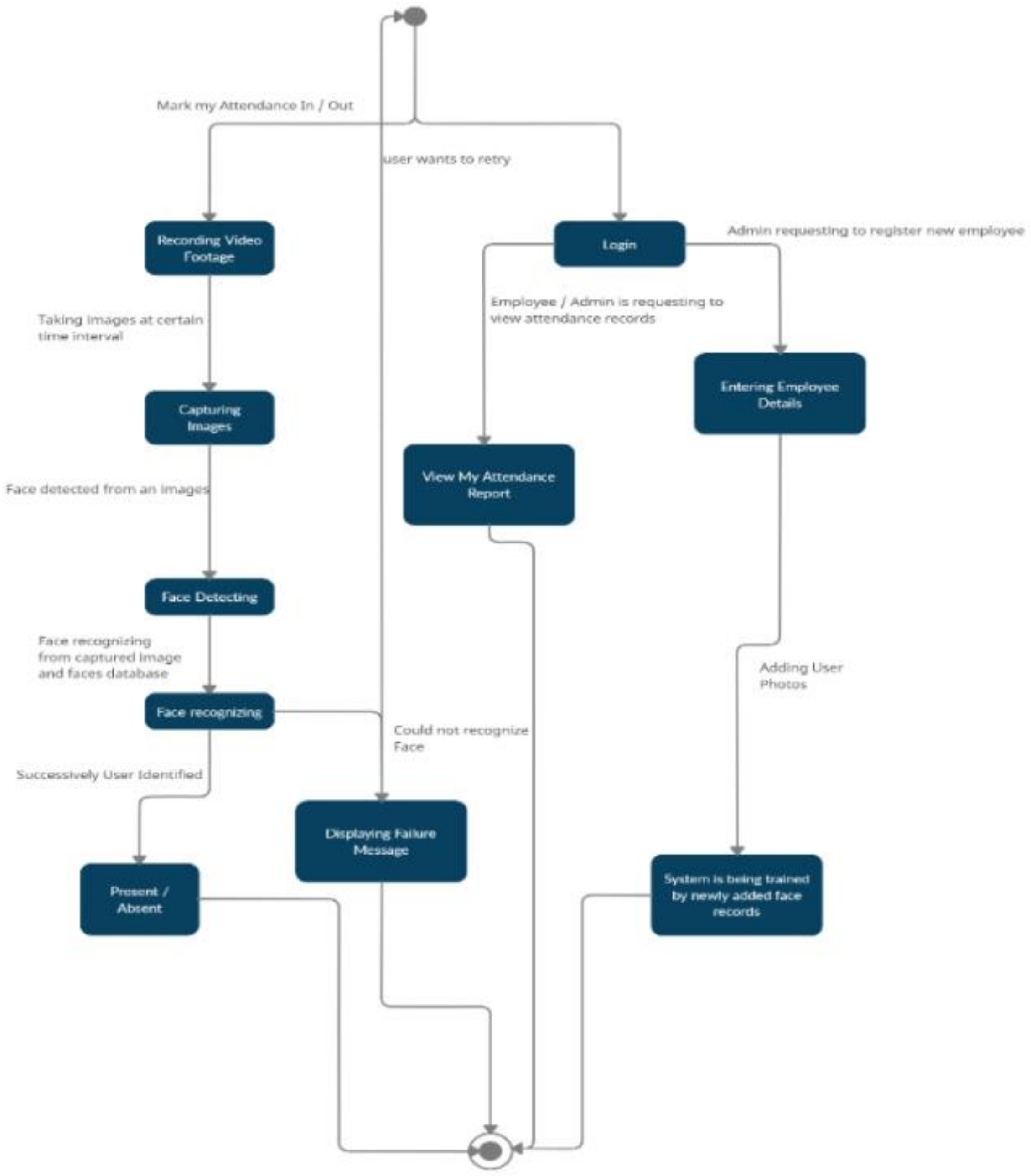

Student
Start
Register
Acquire Image
Add Name & Update Image
Apply Mean Intensity
Update
Face DB
Update

DFD LEVEL-1


Student
Start
Start Real Time Video Feed
Input
Recognition
Recognition of Student
Face DB
Attendance
Results of Marked Attendance

DFD LEVEL-2

STATE DIAGRAM


Mark my Attendance In / Out
user wants to retry
Recording Video Footage
Login
Admin requesting to register new employee
Taking images at certain time interval
Employee / Admin is requesting to view attendance records
Capturing Images
Entering Employee Details
View My Attendance Report
Face detected from an images
Face Detecting
Face recognizing from captured image and faces database
Adding User Photos
Face recognizing
Could not recognize Face
Successively User Identified
Displaying Failure Message
Present / Absent
System is being trained by newly added face records

The state diagram helps us to better understand the simple functions as well as the complex ones that the system offers. By using the state diagram, we identify the dynamic behavior of the system as a whole or even of the subsystems. Exactly with the help of the state diagram, we determine the different states of the participants during the operation with the system.

System Development

The application is built using Visual Studio Code software as well as the Python programming language. Special emphasis during the development of the system has been the design of the interface always seems to be "user friendly" and as easy to use, so that it can be easily used even by people who do not have a computer background.

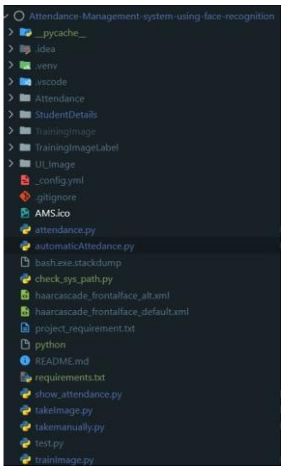
The structure of the project from the software side which includes the necessary coding parts which are arranged in VS code, as well as other assets which are used in the development of the system are shown in the following picture:

Such ranking of directories is made based on my selection, making it easier to manage coding components as well as easier to access them.

The main project file is "attendance.py" in which the connection is made between all other coding parts, as well as the execution of the basic functions of the system. The execution of the system is done exactly with this file by executing the command "py attendance.py"

In "show_attendance.py" is the part which enables the lecturer to publish the reports of previously received presences. "TakeImage.py" accepts inputs or images during the registration process of new students. "TrainImage.py" is one of the most important parts of the system with which after taking the images the system deals with the training so that in the future it can do the detection without making mistakes in recognizing the faces that it has previously registered.

The execution of the system is done by executing the command "py attendance.py" in the terminal, and this command then initiates the start of the system and the screen should appear the part that looks like displayed in figure 11.



```
Attendance-Management-system-using-face-recognition
  > __pycache__
  > .idea
  > .venv
  > .vscode
  > Attendance
  > StudentDetails
  > TrainingImage
  > TrainingImageLabel
  > UI_Image
    _config.yml
    .gitignore
    AMS.ico
    attendance.py
    automaticAttedance.py
    bash.exe.stackdump
    check_sys_path.py
    haarcascade_frontalface_alt.xml
    haarcascade_frontalface_default.xml
    project_requirement.txt
    python
    README.md
    requirements.txt
    show_attendance.py
    takeImage.py
    takemanually.py
    test.py
    trainImage.py
```

Figure 10. Project Files

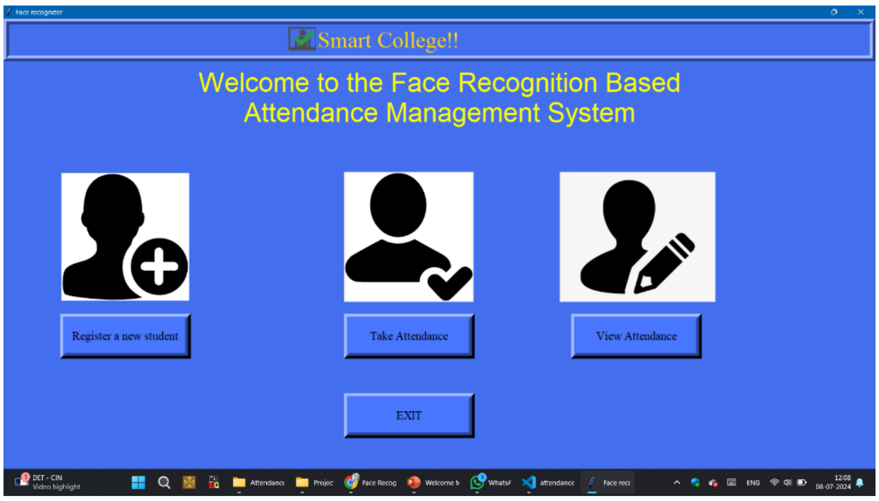
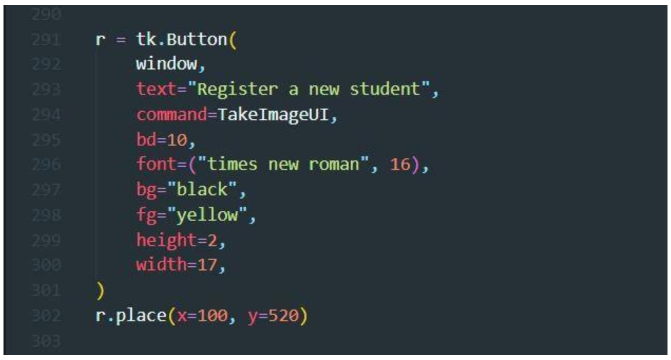


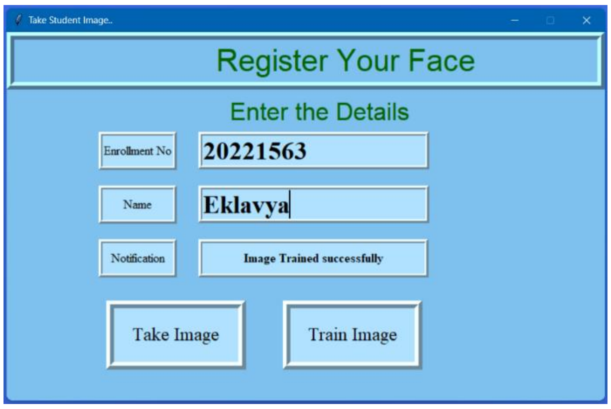
Figure 11. System

The opening interface of the system is built from the ribbons that name it, and below is divided into 3 parts which are the main functions of the system, the first part enables the registration of new students in case you click on the button, the second part enables the start of the present acquisition process, while the third part provides reports from previously received presences. As well as at the end we have the exit button that enables system shutdown. The coding part which has enabled the construction of this interface will be displayed in figure 12. Whereas by clicking the "Register new student" button, a new interface has been designed which will accept inputs that have previously been defined as necessary for the specification of new students, where each student in addition to the name will also contain an Id which is unique to all. The design of this part is shown below in figure 13.



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

Figure 12. Code Extract



Take Student Image..

Register Your Face

Enter the Details

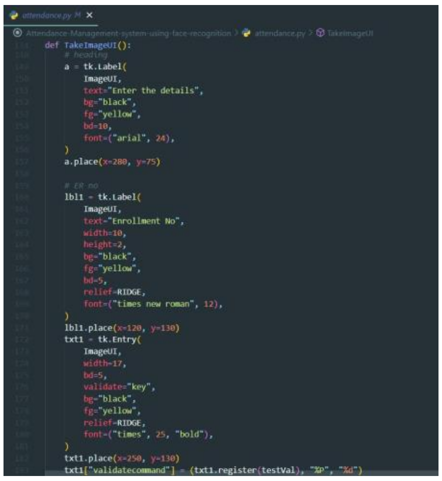
Enrollment No: 20221563

Name: Eklavya

Notification: Image Trained successfully

Figure Register new students UI

After filling in the required data in the above fields then it is necessary to click the Take Image button in which case the system opens the camera and then captures images and finally gives the message if the images were taken successfully. After taking the images the same process happens with the Train Image button which in the background trains the captured images but the users do not interact with it, except at the end when the system notifies if the training was successful, or if it failed. The coding part for building this interface is displayed below in figure 14.



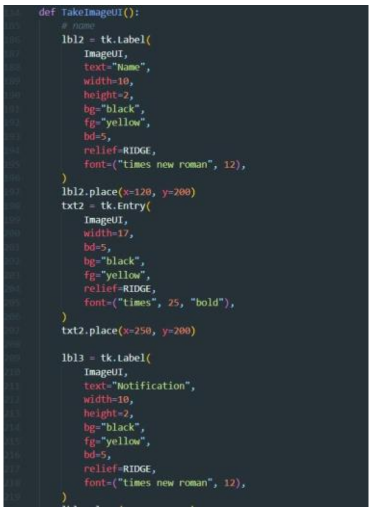
attendance.py M

Attendance-Management-system-using-face-recognition > attendance.py > TakeImageUI

```
def TakeImageUI():
    # heading
    a = tk.Label(
        ImageUI,
        text="Enter the details",
        bg="black",
        fg="yellow",
        bd=10,
        font=("arial", 24),
    )
    a.place(x=280, y=75)

    # ER no
    lbl1 = tk.Label(
        ImageUI,
        text="Enrollment No",
        width=10,
        height=2,
        bg="black",
        fg="yellow",
        bd=5,
        relief=RIDGE,
        font=("times new roman", 12),
    )
    lbl1.place(x=120, y=130)
    txt1 = tk.Entry(
        ImageUI,
        width=17,
        bd=5,
        validate="key",
        bg="black",
        fg="yellow",
        relief=RIDGE,
        font=("times", 25, "bold"),
    )
    txt1.place(x=250, y=130)
    txt1["validatecommand"] = (txt1.register(testVal), "%P", "%d")
```

Figure 14. Input field design

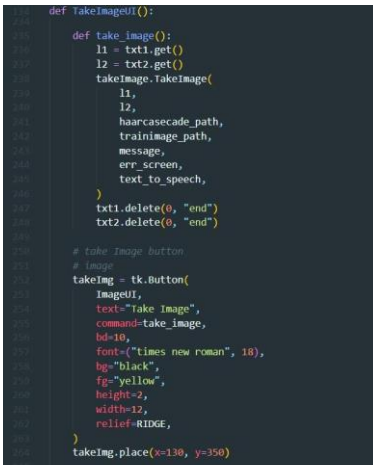


```
def TakeImageUI():
    # name
    lbl2 = tk.Label(
        ImageUI,
        text="Name",
        width=10,
        height=2,
        bg="black",
        fg="yellow",
        bd=5,
        relief=RIDGE,
        font=("times new roman", 12),
    )
    lbl2.place(x=120, y=200)
    txt2 = tk.Entry(
        ImageUI,
        width=17,
        bd=5,
        bg="black",
        fg="yellow",
        relief=RIDGE,
        font=("times", 25, "bold"),
    )
    txt2.place(x=250, y=200)

    lbl3 = tk.Label(
        ImageUI,
        text="Notification",
        width=10,
        height=2,
        bg="black",
        fg="yellow",
        bd=5,
        relief=RIDGE,
        font=("times new roman", 12),
    )
```

Figure 15. Input field

The function that is called on the take image button enables the camera to be opened, and is defined as follows:

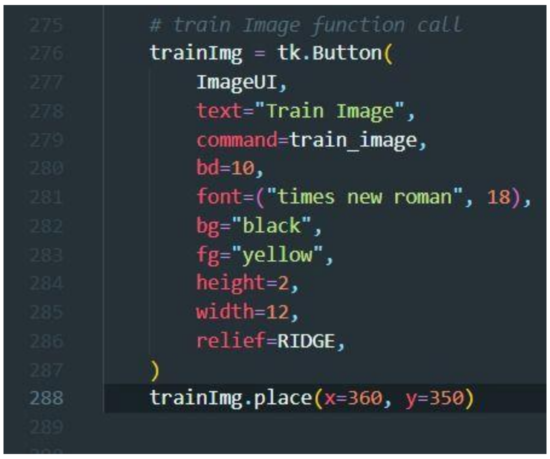


```
def TakeImageUI():

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

    # take Image button
    # image
    takeImg = tk.Button(
        ImageUI,
        text="Take Image",
        command=take_image,
        bd=10,
        font=("times new roman", 18),
        bg="black",
        fg="yellow",
        height=2,
        width=12,
        relief=RIDGE,
    )
    takeImg.place(x=130, y=350)
```

Figure 16. Function that opens camera



```
    # train Image function call
    trainImg = tk.Button(
        ImageUI,
        text="Train Image",
        command=train_image,
        bd=10,
        font=("times new roman", 18),
        bg="black",
        fg="yellow",
        height=2,
        width=12,
        relief=RIDGE,
    )
    trainImg.place(x=360, y=350)
```

Figure 17. Train image

After capturing the images, the system training is required, a function which is defined in the file "trainImage.py" and which is called in the Train Image function in the interface that was presented above.

Take attendance" is the function that enables the lecturer to start directly with the process of taking attendance but first by specifying the subject in question and then has the opportunity to start the process, but also to see the reports from past lectures, the interface of this process as well as the coding part and function are displayed below.

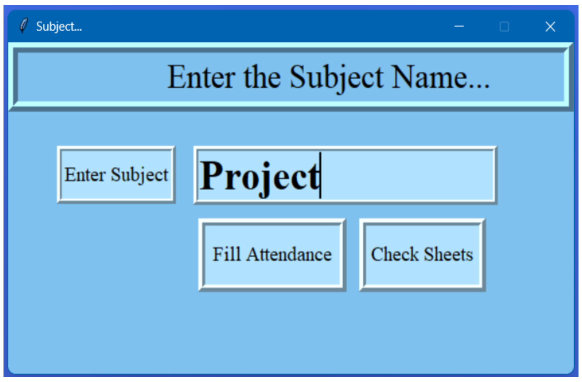
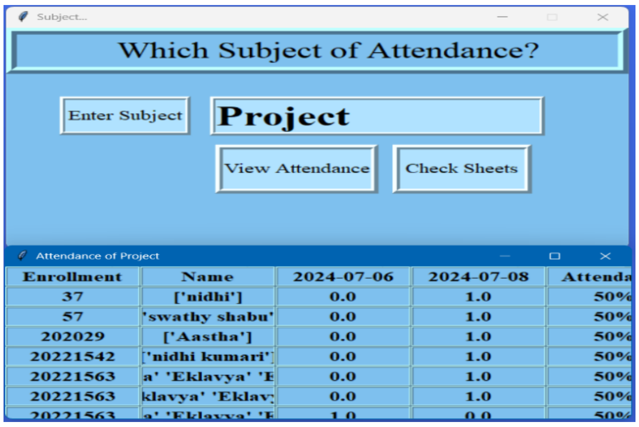


Figure 18. Take attendance UI

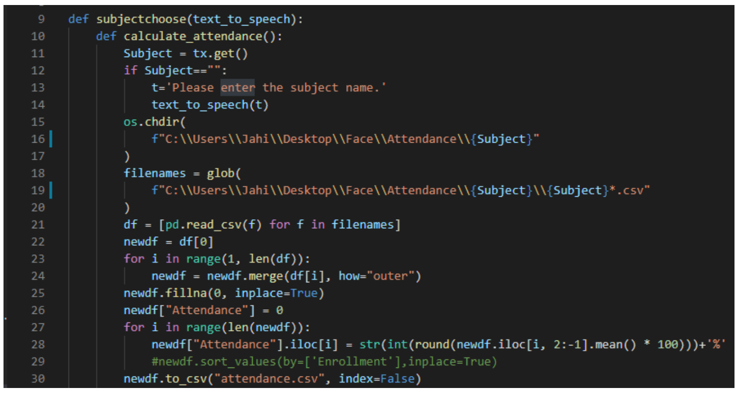


Enrollment	Name	2024-07-06	2024-07-08	Attenda
37	['nidhi']	0.0	1.0	50%
57	'swathy shabu'	0.0	1.0	50%
202029	['Aastha']	0.0	1.0	50%
20221542	['nidhi kumari']	0.0	1.0	50%
20221563	a' 'Eklavya' 'E	0.0	1.0	50%
20221563	klavya' 'Eklav	0.0	1.0	50%
20221563	a' 'Eklavya' 'E	1.0	0.0	50%

Figure 19. Showing attendance UI

The "View Attendance" function enables lecturers to view reports from previously received presences, in tables that are generated within the system, but also in excel files which are created automatically. The following figures show the graphical representation of the function, but also the coding part with which the construction of this function has been achieved.

The coding part includes two functions that are related to the buttons shown in the figure, by clicking the "view attendance" button displays the table which is shown in the figure, while by clicking the "check sheets" button then the system directs us to excel documents in which the registered presence is also stored.



```
def subjectchoose(text_to_speech):
    def calculate_attendance():
        Subject = tx.get()
        if Subject=="":
            t='Please enter the subject name.'
            text_to_speech(t)
        os.chdir(
            f"C:\\Users\\Jahi\\Desktop\\Face\\Attendance\\{Subject}"
        )
        filenames = glob(
            f"C:\\Users\\Jahi\\Desktop\\Face\\Attendance\\{Subject}\\{Subject}*.csv"
        )
        df = [pd.read_csv(f) for f in filenames]
        newdf = df[0]
        for i in range(1, len(df)):
            newdf = newdf.merge(df[i], how="outer")
        newdf.fillna(0, inplace=True)
        newdf["Attendance"] = 0
        for i in range(len(newdf)):
            newdf["Attendance"].iloc[i] = str(int(round(newdf.iloc[i, 2:-1].mean() * 100)))+'%'
            #newdf.sort_values(by=['Enrollment'],inplace=True)
        newdf.to_csv("attendance.csv", index=False)
```

Figure 20. View attendance coding part

IMPLEMENTATION

To conduct studies and analyses of an operational and technological nature, and to promote the exchange and development of methods and tools for operational analysis as applied to defence problems.

1.1 HARR Cascade Face Detection

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

The algorithm has four stages:

- Haar Feature Selection
- Creating Integral Images
- Adaboost Training
- Cascading Classifiers

It is well known for being able to detect faces and body parts in an image, but can be trained to identify almost any object.

1.2 LBPH Face Reorganization:

Local Binary Patterns Histogram algorithm was proposed in 2006. It is based on local binary operator. It is widely used in facial recognition due to its computational simplicity and discriminative power.

The steps involved to achieve this are:

- creating dataset
- face acquisition
- feature extraction
- classification

The LBPH algorithm is a part of opencv.

CONCLUSION AND FUTURE ENHANCEMENT

A machine learning methods were used to evaluate the student's observable actions in the classroom teaching system. The evaluation was created right after the live feed review. Several models have been produced. Such models were tested using map to decide which model is appropriate for object detection. The map (mean average accuracy) is a common measure used to determine the precision of the artifacts being observed. The experimental testing shows that model accuracy is 88.606%. Tests indicate that this method offers reasonable pace of identification and positive outcomes for the detecting student faces dependent on observable student actions in classroom instruction. The suggested approach is often versatile and responsive to different situations, since more students would be interested in greater room sizes, utilizing a higher form of camera with certain enhancements such as IP camera for continuously capturing images of the students, detect the faces in images and compare the detected faces with the database. It may be used such as greater input picture measurements, anchor box dimensions ideal for different situations and further training details.

The Facial recognition database can be improved by adding more images and variable poses of the students to make recognition full proof. The Web based architecture can be utilized further by maintaining the databases on a remote server and the application will be accessible via the Internet.