MINOR PROJECT REPORT

For

ATTENDANCE MONITORING SYSTEM USING FACE DETECTION TECHNIQUES

Submitted By

Specialization	SAP ID	Name
CSF	500083471	Alok Dixit
CSF	500083489	Dharun S.R.
AIML	500087339	Divyanshu Bansal
AIML	500088053	Parag Anil Bedase

Department of Informatics

School of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

DEHRADUN- 248007. Uttarakhand

Project Guide: Dr. Akashdeep Bhardwaj

ABSTRACT

In this digital era, face recognition system plays a vital role in almost every sector. Face recognition is one the mostly used biometrics. It can used for security, authentication, identification, and has got many more advantages. It is being widely used due to contactless and non-invasion process. Face recognition system can also be used for attendance marking in schools, colleges, offices, etc. This system aims to build a class attendance system which uses the concept of face recognition as existing manual attendance system is time consuming and cumbersome to maintain. And there may be chances of proxy attendance.

This system consists of three phases- face detection, face recognition, attendance updating. Faces are detected and recognized from live streaming video of the classroom. Attendance will be generated to the respective faculty at the end of the session. Face detection and recognition is performed using Haar-Cascade classifier and Cascade Classifier algorithm respectively. Faces are detected and recognized from live streaming video of the classroom.

TABLE OF CONTENTS

SI.NO CONTENTS

1. INTRODUCTION

This project is entitled as Attendance Monitoring using Face Recognition. It is developedin Tkinter python GUI. The project mainly deals with desktop application that will provides attendance for student. This is very helpful for the college faculty they can be able to view the attendance of particular session in CSV file. The purpose of this system is to build a attendance system which is based on face recognition techniques. Here the face of an individual will be considered for marking attendance his desktop application which provides a clean, informative, and interactive user interface to see and register for attendance. It contains option to add student, view attendance and manage the overall attendance.

Face recognition has set an important biometric feature, which can be easily acquirable and is non-intrusive. Face recognition-based systems are relatively oblivious to various facial expression. Face recognition system consists of two categories: verification and face identification. Face verification is a 1:1 matching process, it compares face image against the template face images

Here face of an individual will be considered for marking attendance. Nowadays, face recognition is gaining more popularity and has been widely used. In this paper, we proposed a system which detects the faces of students from live streaming video face is found in the database. This new system will consume less time than compared to traditional methods.

1.1 Existing System

In the existing Attendance Monitoring system, faculty are not able to get proper attendance about the students while taking seminar or session. The faculty needs to spend the time to get the attendance of the student. The faculty enters the classes to get attendance for the specific class which takes a lot of time.

Nowadays, most of the faculty members is having problems like risk of human Error marking attendance, Incorrect entry of times, too much paperwork, ineffective and outdated etc. To overcome all these problems, we are proposing the system Attendance monitoring using Face Recognition with utilities like different face recognition, Spot registration, automatic or manually marking attendance, statics for attendance report, maintaining log.

1.2 Proposed System

The Attendance Monitoring using Face Recognition is built in Python tKinter for frontend GUI and Python for middleware (Backend).

The users can interact with the system using a GUI. Here users will be mainly provided with three different options such as, student registration, faculty registration, and mark attendance.

The first procedure is the student registration. Here the admin or faculty members enters the details of a student. All these information will be stored in the local system. Next isthe entry of the student, And the system marks the attendance based on the face registered

2. REQUIREMENTS

2.1 Hardware Requirements

• **Processor :** Intel core i5/ AMD Ryzen 5 and above. • **RAM :** 4

GB and Above.

• **Hard Disk :** 500 GB

2.2 Software Requirements

• **Front End :** Python GUI(Tkinter)

• **Back End :** Python

• **Operating System :** Windows 7/ Windows 10/Windows 11

3. MODULE

This project will help the teaching faculties to mark attendance. Through this interface the teaching faculties will be able to view the attendance on certain time and date. It has a simple user interface with two types of new registration for new users and login for an existing user. The teaching faculty shall beable to view marked attendance at the system for future reference. And admin can generate a whole report of attendance.

In this face detection system, we used the below four steps to implement the requirements,

- MODULE 1: DATASET CREATION
- MODULE 2: FACE DETECTION
- MODULE 3: FACE RECOGNITION
- MODULE 4: ATTENDANCE UPDATION

Module 1

Images of students are captured using a web cam. Multiple images of single student will be acquired with varied gestures and angles. These images undergo pre-processing. The images are cropped to obtain the Region of Interest (ROI) which will be further used in recognition process. Next step is to resize the cropped images to particular pixel position. Then these images will be converted from RGB to Gray scale images. And then these images will be saved as the names of respective student in a folder.

Module 2

Face detection here is performed using Haar-Cascade Classifier with OpenCV. Haar Cascade algorithm needs to be trained to detect human faces before it can be used for face detection. This is called feature extraction. The haar cascade training data used is an xml file haarcascade_ frontalface_default. Here we are using detect Multi Scale module from OpenCV. This is required to create a rectangle around the faces in an image. It has got three parameters to consider- scale Factor, min Neighbours, min Size.

Module 3

Face recognition process can be divided into three steps prepare training data, train face recognizer, prediction. Here training data will be the images present in the dataset. They will be assigned with an integer label of the student it belongs to. These images are then used for face recognition.

Module 4

After face recognition process, the recognized faces will be marked as present in the excel sheet and the rest will be marked as absent and the list of absentees will be mailed to the respective faculties. Faculties will be updated with monthly attendance sheet at the end of every month.

4. DESIGN

SAMPLE UNIFIED MODELING LANGUAGE

The heart of the object-oriented problem solving is the connection of a model. The model abstracts the eventual details of the underlying problem from its detail of the underlying usually complicated real world. Several modelling tools are wrapped under the heading of the UML, which stands for unified modelling language. The purpose of the course is to present important highlights of the UML.

• Use case diagram

• Class diagram

- Object diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Activity diagram
- Component diagram
- Deployment diagram

Some of the section of this course contains links to pages with more detailed information. And every section has start question. Use them for understanding of the section topic.

An introduction to UML diagram

The unified modeling language for specifying, constructing, visualizing and documenting the certificate of the software intensive system. Analogue to use of architectural blueprint in the construction industry.

UML provides a common language for describing software model and it can be use in conjunction with a wide range of software life cycle and development process.

Why UML is important?

Let’s look at these questions from the point of view of the construction trade. Architect design building, builders of the design to create buildings. Builders use the design to create more complicated buildings the more critical the communication between architect and builders. Blue print is the standard graphical language that both architects building must learn as part of this trade.

Writing software is not unlike constructing a building, the more complicated underlying system, the more critical the communication among everyone involved in creating and developing the software. In the past decade the UML has emerged as the software blueprint language for analyst, designers and programmers alike. It is new part of software trade. The UML gives everyone from business analyst designer to program a common vocabulary to task about software design.

The UML is applicable to object-oriented program solving. Anyone interested in learning UML must be familiar with the underlying text of object-oriented problem. It all begins with construction of a model. A model is an abstraction of the underlying problem. The domain is the actual word from which the problem comes. Models consist of object that interact by sending each other message think of an object as "alive" object have things, they know and things can be (behavior or operation). The value of the object attribute determines its state.

Classes are the blueprint for object. A class wraps attributes (data) and behavior (method or function) into a single distinct entity object are instances of classes.

Sample Use case diagram

Use case diagram identify the function provided by the system. The user who interacts with the system and the allocation between the user and the functionality. Use case is used and the analysis phase of software development to Particulate the high level requirements of the system. The primary goals of use case diagram include.

- Providing a high-level view of what the system does.
- Identify the user ("actor") of the system.
- Determining areas needed human computer interfaces.

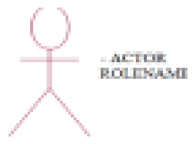
Use case extended beyond pictorial diagram. In fact, text bound use case description as often used to supplement diagrams and the explore use case functionality in more detail.

Graphical Notation

The basic component of use case diagram are actor. The use case and association.

Actor

An actor, as mentioned is a user of the system and is depicted using a stick figure. The role of use is written beneath the icon. Actors are not limited to humans. If a system communication with other application and except input or delivers output then that application can also be considered as actor.

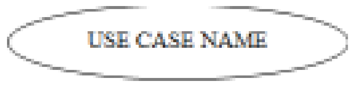


Use case

A use case is functionality provided by the system typically described as verb + object.

Example

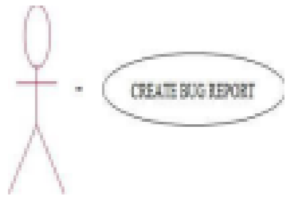
Register (car, delete user) use case is depicted with a ellipse. The name of the use case is written within the ellipse.



Association

It is used to link actor with the use cases and indicate that an actor participates inthe some form. Association is depicted by the link connecting the actor and the user case.

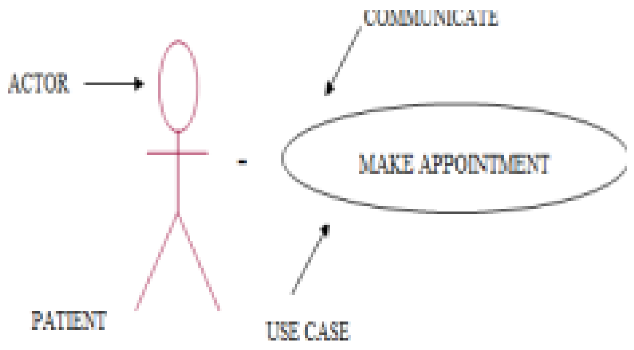
The following image show how there three basic elements work together to form ause case diagram.



Use case diagram describe what a system forms the standard stand point of an external observer. The emphasis is what system does rather than how. Use case diagram are usually clearly and closely connected to scenarios. A scenario is an example of what happen, when someone interact with the system. There is a scenario for a medical clinic.

“A patient calls the clinic to make an appointment for a yearly checkup. The receptionist finds the nearest empty time slot in the appointment books and schedule the appointment for that time slot.

A use case is a summary of scenario for a single task or goals. An actor is who or what initiate the events involved are simply role that people object play. The picture below is the make appointment use case for the medical clinic. The actor is a patient. The connection between actors and use case is a communicationassociation.



Actor are stick figure user case are over all communicate and line that a link actor to use case.

A use case diagram is a collection of actor use case and their communication; we must put make appointment as a part of diagram with 4 actor of 4 use cases. Notice that a single use case has multiple actors.

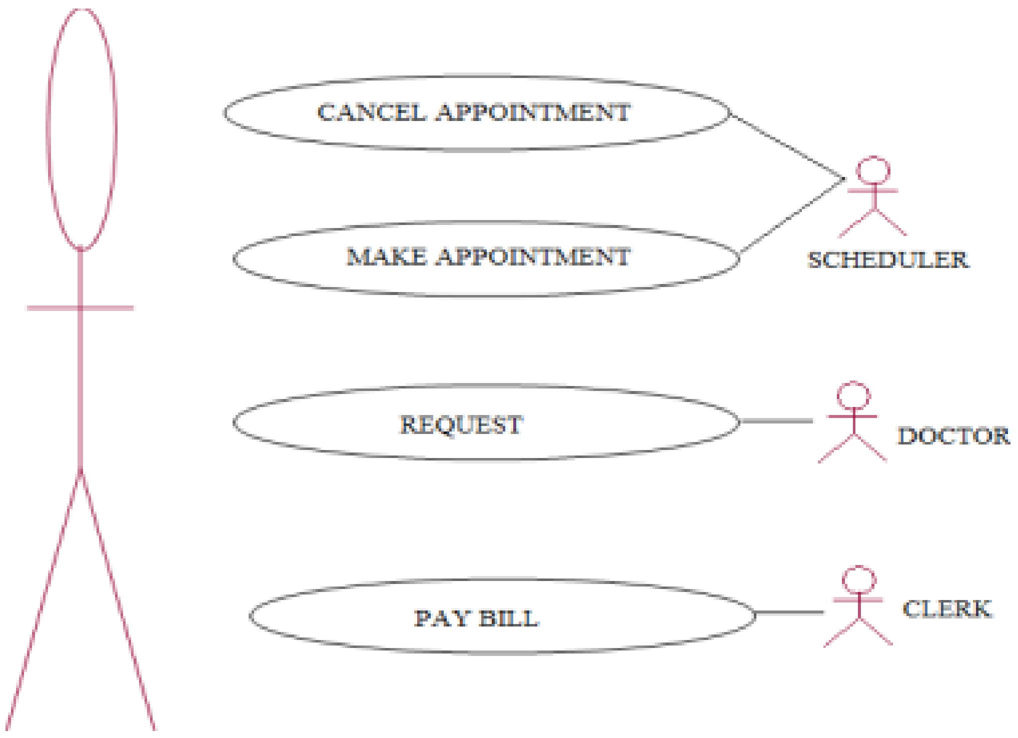
Use case diagram are helpful in three areas.

Deterministic Features

New use case often generates a new request as the system is analyzed and the design that sequence.

Communicating with client

Their national simplicity makes use case diagram a good way of those scenarios.



The collection of scenario for a use case may suggest a suite of test cases for those scenarios.

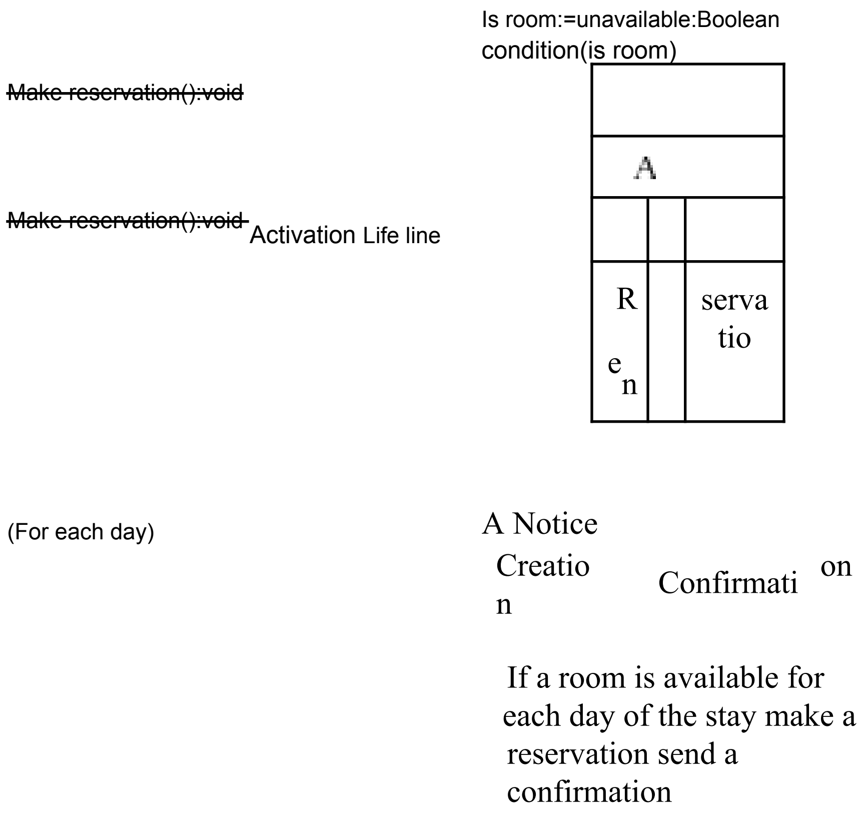
Sample Sequence diagram

Sequence diagram document that interact between classes to achieve a result such a use case. Because UML is designated for OOP that communicate between classes are known as message.

The sequence diagram lists object horizontally and time vertically and modally their message overtime.

Make
Reservation():void
A Hotel:Hotel

Window:User Interface


Is room:=unavailable:Boolean
condition(is room)
~~Make reservation():void~~
A
~~Make reservation():void~~
Activation Life line
R
e
n
serva
tio
(For each day)
A Notice
Creatio
n
Confirmati
on
If a room is available for each day of the stay make a reservation send a confirmation

Sample Sequence Diagram

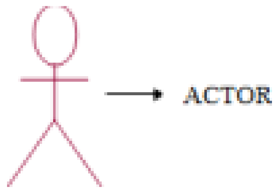
Object

Objects are instance of classes are arranged horizontally. The pictorial representation for an object is a class (a rectangle) with name preferred by the object name (optional) and a semicolon.

: object

Actor

Actor can also communicate with object so two can be listed as a column. An actor is modeled using the unambiguous symbol the stick figure.



Life time

The lifetime indicates the existences of the object overtime. The notation for the lifetime is a vertical dotted line extending from an object.

Activation

Activation modeled as a rectangular box on the lifetime, indicate when the object is performing an action.

Message

Message modeled as horizontal area between activation indicate the

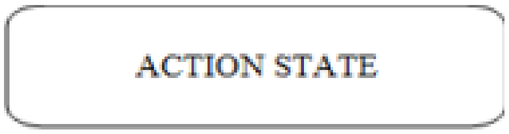
communication between the object.

Sample Activity diagram

It is used to document workflow in a system, from the business level to operational level. When looking at an activity diagram, in fact the activity diagram is a variation of the state diagram. When the “states” represent operation and the transition represent the activities that happen when the operation is complete. The general purpose of a activity diagram is to flow on flow drive by internal processing Vs external extends.

Activity stages

Activity stages names as action by an object. The notations for these states are rounded rectangle, the same notation is found in state chart diagram.



Transition

When an activity state is completed processing, moves to another state. Transitions are used to move this movement. Transitions are modeled using arrows. **Transition**

Swim lane

It divides activities according to object by arranging object in column object arelinked at the top of the column and vertical bars separate the column to form swim lanes.

Initial state

If marks the entry point and the initial activity state. The notation for the initial states in same as state chart diagrams a solid circle there an only be one initial state.

Final state

Final state mark the end of the modeled work-flow. These can be multiple final state on a diagram and those states are modeled using a said circle surrounded by another circle.

Synchronization bar

Activity of ten can be done is parallel to split processing(“bork”) or to resume processing multiple activities has been completed (“john”) synchronization bars are used these moduled as solids rectangles with multiple transition going in and or row.

Sample Component diagrams

Component diagram fall under the category of an implementation and diagram a kind of diagram that modules the implementation and development of the system. A component diagram is particular is used to describe the dependencies between exe files and source file. This information is similar to that within make files, which describes source code dependencies and can be used to properly compile an application.

Component

A component represents a software entity in a system. Eg. include source code files, program document and resource files. A component is represented using a rectangular box with two rectangle potential from the left side, as seeing the image to the right.

Dependency

Dependency is used to model the relationship between a component. The notation for the dependency relationship is a dotted arrow. Pointing from a component to

depend on.

Sample Class diagram

A class diagram gives the over view of the system by showing its clauses and the relationships among them. Class diagram are static they display what interacts but not what happens when they do interact.

The class diagram below models a customer order from a detail category. The control class in the order. Associated with it are customer making in the purchase and payment is one of 3 kinds, cash, check or credit. The order contains the order detail (line, terms) each with its associated item.

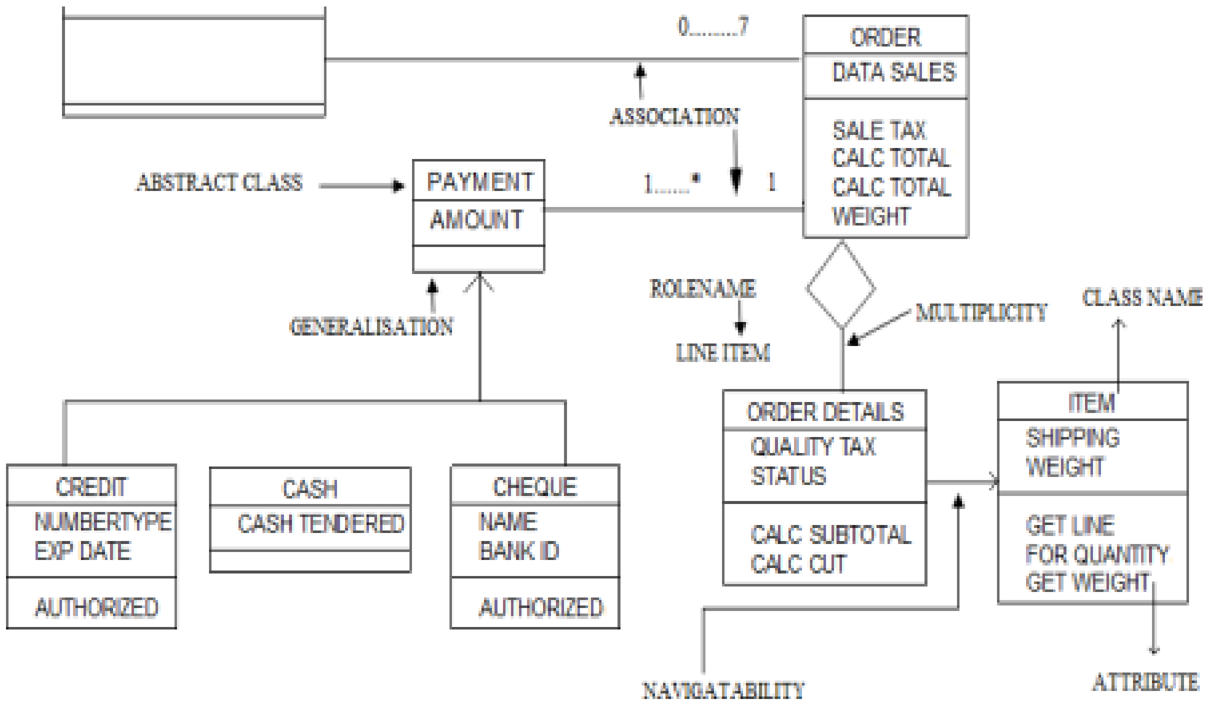
UML class notation is a rectangle divided into three parts. Class name, attribute and operations.

Name of abstract clauses such as payment are in italics. Relationship between clauses are the connecting links.

Our class diagram has 3 kinds of relationship

- ASSOCIATION

- AGGREGATION
- GENERALIZATION



Sample class diagram

Association

A relationship between instances of a 2 clause. There is an association between two clauses. If and instance of one class must know about the other in order to perform the work. The diagram an association is a link connecting two clause.

Aggregation

An association in which one clause belong to a collection. An aggregation has a diamond end pointing to other part containing the whole. In our diagram, order as a collection of order detail.

Generalizations

An inheritance link indicating one clause is a super class of its other. A generalization has a pointing to the super class. Payment is a super class of cash, check and credit.

An association has two ends. An end may have role name to clarity the nature of association. For resemble an order detail is a line item of each other

A navigability arrow on a association end is the no of possible instance of the class associated with a single instance of the class. Multiplicities are single number or range of number. In our resemble, there can be only one customer for each order but a customer can have any number of orders.

The table gives the most common multiplicities.

Multiplication	Meaning
0………1	Zero or one instances. The notation n……n indicates n to m instance.
0*… .....or*	No limit on the number by instances(including name)
1*	Exactly one instances
1…..*	Atleast one instance

Every table class diagram has clauses, association and multiplicities. Navigability of roles are optional items placed in a diagram to provide a clarity.

Packages and object diagrams

To simplify the complexity classes diagram you can group class into package. A package is a collection of logically related UML element. The diagram below is abusiness model in which the clauses are grouped into packages.

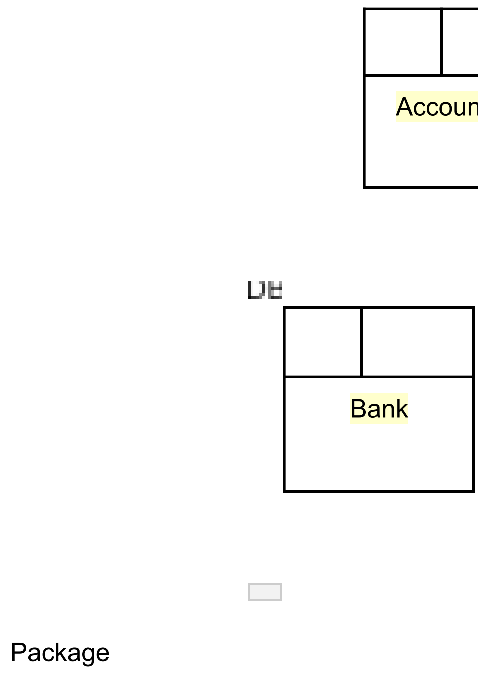
Applied Math:Department

Sample package and object diagram

The hotel chain the send a make reservation message to the hotel. If the hotel has available rooms, then it makes a reservation and a confirmation.

Each rectangle in the object diagram correctness to a single instance. Instance name are underlined in UML diagrams.

Class or instance names may be omitted from object diagrams as long as the diagram meaning is still clear.



Package appears as rectangle with small tasks at the top. The packages names is on the tab at inside the rectangle. The dotted arrows are dependent once package depends on one another if the changes.

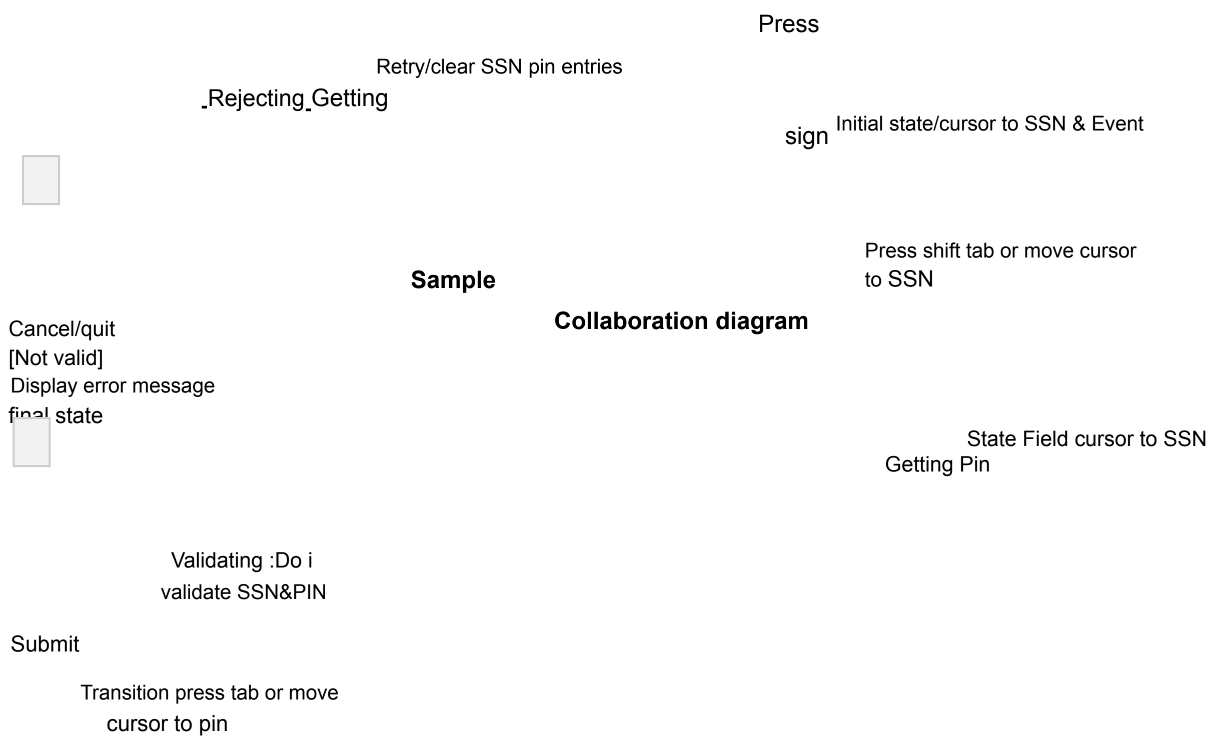
This small class diagram show that a university department an contain lot of the other department.

The object diagram below instantiates the class diagram replacing it by a accurate example.

Sample Collaboration diagram

Collaboration diagram are also interaction diagram. They convey the same information as sequence diagram, but they forces an object roles instead of the timer, that message are send. In sequence diagram, object role are the vertices and message are connecting lines.

The object role rectangles are labeled with either class or object name (or both) class names are preceded by colons(:).


Press
Retry/clear SSN pin entries
Rejecting Getting
sign
Initial state/cursor to SSN & Event
Press shift tab or move cursor to SSN
Sample
Collaboration diagram
Cancel/quit
[Not valid]
Display error message
final state
State Field cursor to SSN
Getting Pin
Validating :Do i validate SSN&PIN
Submit
Transition press tab or move cursor to pin

Each message in a collaboration diagram has a sequence number. The top-level message is numbered.

Message at the same level (send during the same cell) have the same decimal prefix but suffix of 1,2 etc according to when they occur.

Sample State chart diagram

Object has behavior and state. The state of the object depends on its current activity or condition. The state chart diagram shows the possible states of the object and the transactions that cause a change of state.

Our example diagram model the login part of an online banking system. Logging an consists of entering a valid solid security number and personal id number. Then submitting the information of validation.

Logging in can be factored into four non overlapping state. Getting SSN, getting PIN validating and rejecting, from each comes a complete set of transactions that determine the set sequence state.

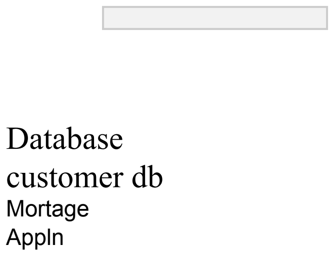
Our diagram has 2 self-transitions, one on getting SSN, and other a getting PIN. The initial state is a dummy to start the action. Final state are also dummy state that terminate the action. The activation bar represents the duration of execution of the message. In our diagram, the hotel issues a self all to determine if a room is available. If so then the hotel creates a reservation and a confirmation. The expression is square bracket[] is an condition.

The action that occurs as a result of an event or condition is expressed is the formulation, while in its validating state the object does not wait for an outside event to trigger a transition. Information it performs an activity. The result of that activity determines its subsequent state.

Sample Component and deployment diagrams

A component is a code module component diagram are physically analog of class diagram. Deployment diagram shows the physical configuration of s/w and hardware. The following deployment diagram show the relationship among software and hardware components in real estate transition.

The physical hardware is made up of nodes. Each component belongs on a node. Components are show as rectangle with two table at the upper left.



deployment diagram

Sample component and

Feasibility

Feasibility study is conducted once the problem is clearly understood. Feasibility study is a high-level capsule version of the entire system analysis and design process. The objective is to determine quickly at a minimum expense how to solve a problem. The purpose of feasibility is not to solve the problem but to determine if the problem is worth solving.

The system has been tested for feasibility in the following

points. • Technical Feasibility

• Economic Feasibility

• Operational Feasibility.

Technical Feasibility

The project entitles "ATTENDANCE MONITORING USING FACE RECOGNITION" is technically feasibility because of the below mentioned feature. The project was developed in Python which Graphical User Interface. It provides the high level of reliability, availability and compatibility. All these make Python an appropriate language for this project. Thus the existing software Python is a powerful language.

Economical Feasibility

The computerized system will help in automate the selection leading the profits and details of the organization. With this software, the machine and manpower utilization are expected to go up by 80-90% approximately.

Operational Feasibility

In this project, the management will know the details of each project where he may be presented and the data will be maintained as decentralized and if any inquires for that particular contract can be known as per their requirements and necessaries.

4.1 UML DIAGRAMS FOR PROPOSED SYSTEM

4.1.1. Class Diagram for Attendance System

Figure 4.1.1

The class diagram is an illustration of Unified Modelling Language. It was provide with its attributes with matching methods to show the structure of each class. It resembles a chart in which classes that will be included in project with three rectangles. Their relationships also plotted to show connection between classes and their multiplicity.

4.1.2 Use Case Diagram for Attendance System

We have two use case diagrams here, One for New Register of students and another one for Taking attendance in classes or session. That has explained below

New Registration

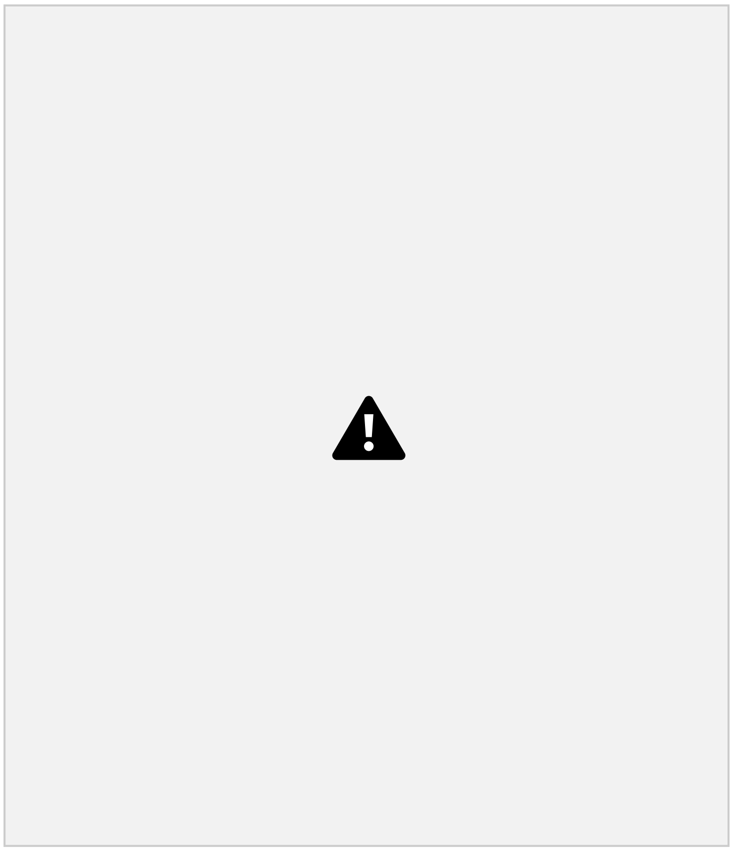


Figure 4.1.2 Use case diagram for New Registration

The admin can open the application to register new student, with his/her details. Then the admin opens the camera to register the student's face. After save the profile with password provided. Then the system prompts the message as profile.

Taking Attendance

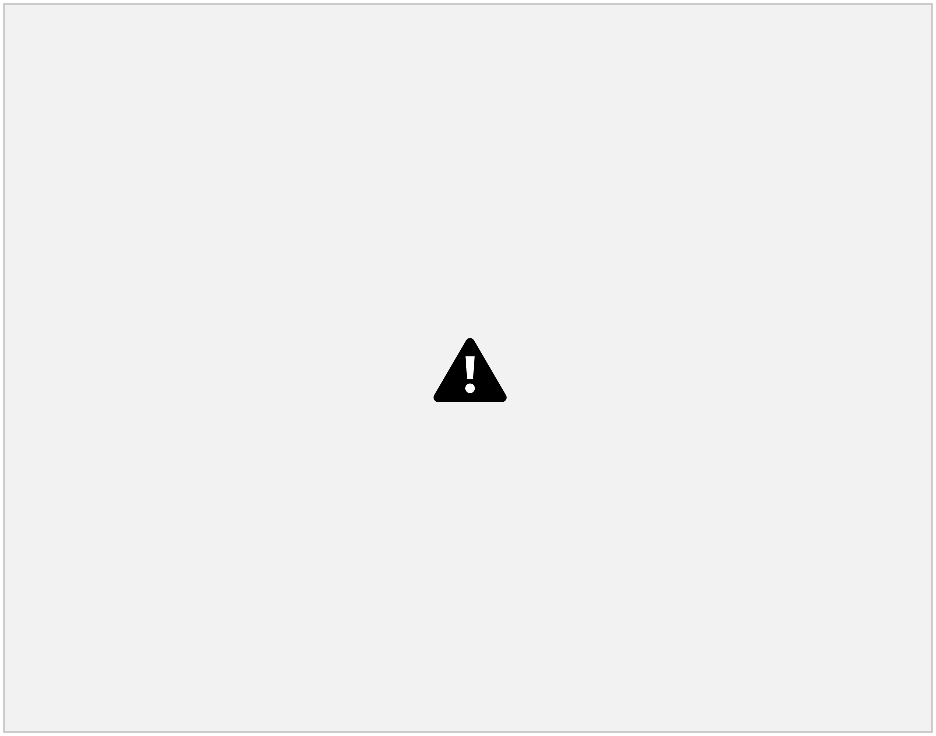


Figure 4.1.2.1 Use case diagram for Taking Attendance

The admin or faculty have access to take attendance that triggers the system to opens up the camera and the face detection system invokes to detect the faces. Already registered students name, id and in time marked into document. Then the attendance sheet is generated in respective day.

4.1.3 State Diagram for Attendance System

Figure 4.1.3 State diagram for Attendance System

The class teacher or admin have access to view the attendance or who have entered the class. The state diagram follows from student registration to Attendance generation. The unknow person who entered the class marked as unknow.

4.1.4 Sequence Diagram for Attendance System

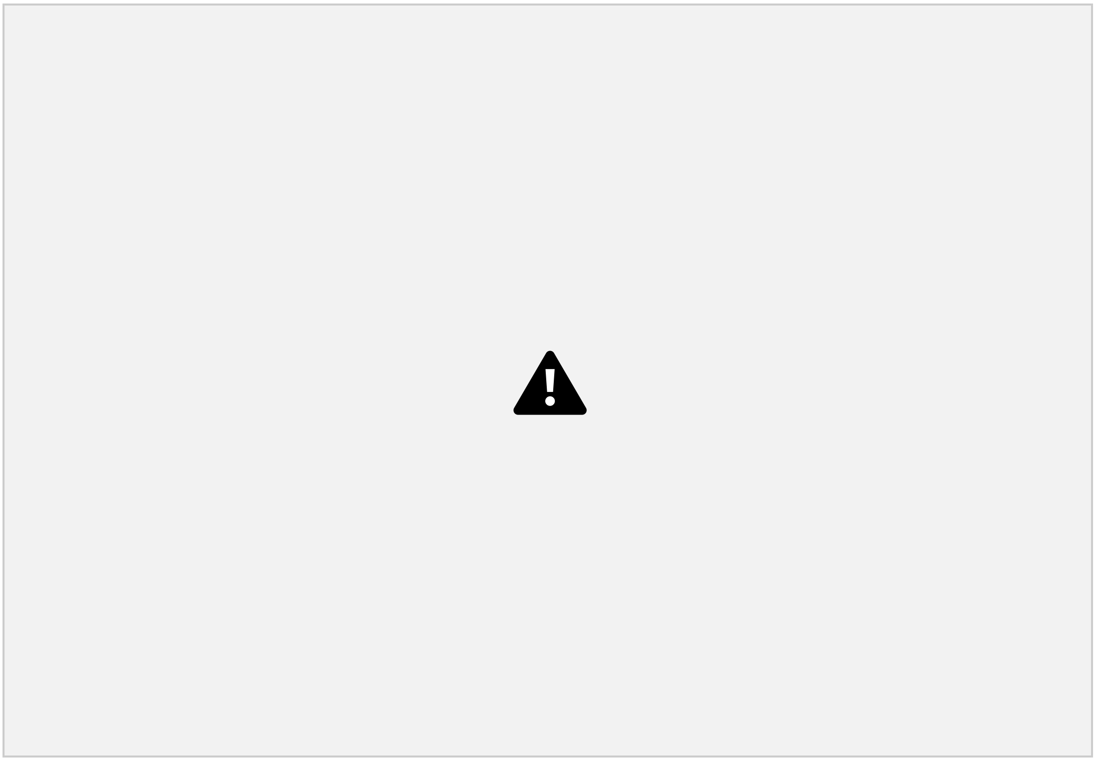


Figure 4.1.4 Sequence Diagram for Attendance System

This sequence diagram shows the operation performs in the activity. The admin can view the time and date of the upcoming student, and do marking of attendance for the desired classes, Operations are listed from left to right occurring to when they take part in the message sequence.

4.1.5. Activity Diagram for Attendance System

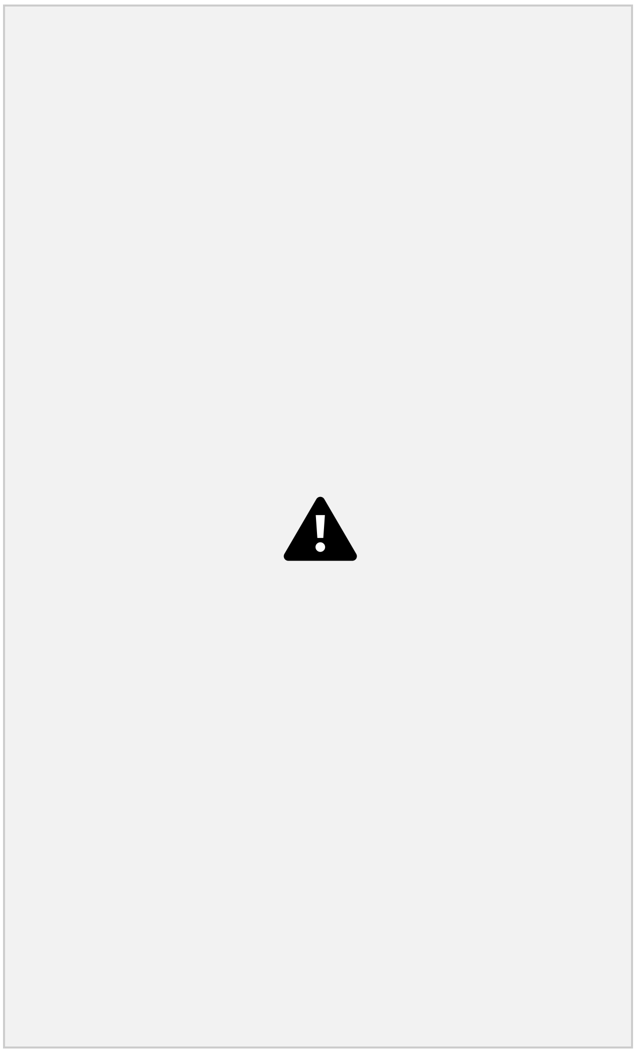


Figure 4.1.5 Activity Diagram for Attendance System

The Activity Diagram for Face Recognition System is a designed illustration that provides its behavioral aspect. This Activity Diagram shows the flow of activities and decision paths that exist in the system.The Face Recognition System Activity Diagram is built up of activities, decisions, and paths. It uses symbols to define the overall workflow of the activity diagram.

4.1.6 Collaboration Diagram of Attendance System

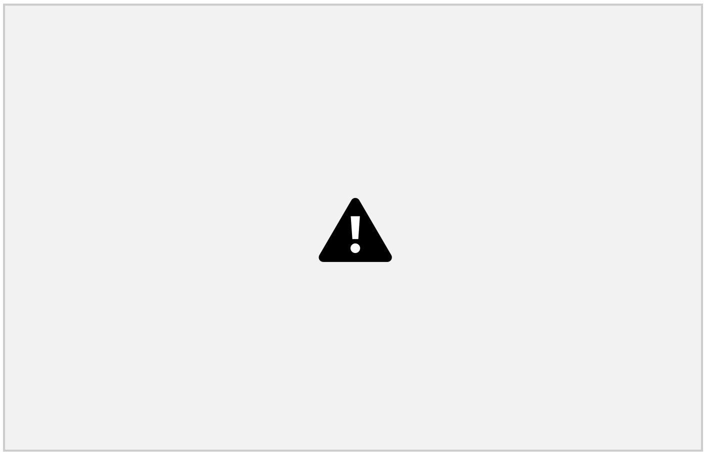


Figure 4.1.6 Collaboration Diagram for Attendance system

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object. The collaboration diagrams are best suited for analyzing use cases.

5. IMPLEMENTATION OF PROJECT WROKS

Libraries

- import tkinter as tk
- from tkinter import ttk
- from tkinter import messagebox as mess
- import tkinter.simpledialog as tsd
- import cv2,os
- import csv
- import numpy as np
- from PIL import Image

Description of import tkinter

Tkinter Label widget is used to display a text or image on the screen. To use a Label widget, you use the following general syntax: label = ttk.Label. appearance

Description of from tkinter import ttk

button = ttk.Button(container, text, command) In this syntax: The container is the parent component on which you place the button. The text is the label of the button. The command specifies a callback function that will be called automatically when the button clicked.

Description of from tkinter import messagebox

Python Tkinter – MessageBox Widget is used to display the message boxes in the python applications. This module is used to display a message using provides a number of functions.

Description of import tkinter.simple dialog

The Simple Dialog module is used to create dialog boxes to take input from the user in a variety of ways. Simple Dialog allows us to take input of varying datatypes from the user, such as float, string and integer. The below Tkinter Simple Dialog functions work together with tkinter.

Description of import cv2,os

cv2 is the module import name for opencv-python, “Unofficial pre-built CPU only OpenCV packages for Python”. The traditional OpenCV has many complicated steps involving building the module from scratch, which is unnecessary. I would recommend remaining with the opencv-python library.

Description of import csv

You can import data from a text file into an existing worksheet. On the Data tab, in the Get &amp; Transform Data group, click From Text/CSV. In the Import Data dialog box, locate and double-click the text file that you want to import, and click Import.

Description of import numpy as np

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software.

Description of from PIL import Image

To load the image, we simply import the image module from the pillow and call the Image. open(), passing the image filename. Instead of calling the Pillow module, we will call the PIL module as to make it backward compatible with an older module called Python Imaging Library (PIL).

Description of import pandas as pd

The import pandas portion of the code tells Python to bring the pandas data analysis library into your current environment. The as pd portion of the code then tells Python to give pandas the alias of pd. This allows you to use pandas functions by simply typing pd.

Description of import datetime

In the Python programming language, datetime is a single module. This means that it is not two separate data types. You can import this datetime module in such a way that they work with dates and times. datetime is a built-in Python module.

Description of import time

time() The time() function returns the number of seconds passed since epoch. For Unix system, January 1, 1970, 00:00:00 at UTC is epoch (the point where time begins). import time seconds = time.time() print(&quot;Seconds since epoch =&quot;, seconds)

Frontend GUI

```
window = tk.Tk()
window.geometry("1280x720")
window.resizable(True,False)
window.title("Attendance System")
window.configure(background='#262523')
```

```
frame1 = tk.Frame(window, bg="#00aeff")
frame1.place(relx=0.11, rely=0.17, relwidth=0.39, relheight=0.80)

frame2 = tk.Frame(window, bg="#00aeff")
frame2.place(relx=0.51, rely=0.17, relwidth=0.38, relheight=0.80)

message3 = tk.Label(window, text="Face Recognition Based Attendance System"
,fg="white",bg="#262523" ,width=55 ,height=1,font=('times', 29, ' bold '))
message3.place(x=10, y=10)

frame3 = tk.Frame(window, bg="#c4c6ce")
frame3.place(relx=0.52, rely=0.09, relwidth=0.09, relheight=0.07)

frame4 = tk.Frame(window, bg="#c4c6ce")
frame4.place(relx=0.36, rely=0.09, relwidth=0.16, relheight=0.07)

datef = tk.Label(frame4, text = day+"-"+mont[month]+"-"+year+" |",
fg="orange",bg="#262523" ,width=55 ,height=1,font=('times', 22, ' bold '))
datef.pack(fill='both',expand=1)

clock = tk.Label(frame3,fg="orange",bg="#262523" ,width=55 ,height=1,font=('times', 22, '
bold '))
clock.pack(fill='both',expand=1)
tick()

head2 = tk.Label(frame2, text="For New Registrations ", fg="black",bg="#3ece48"
,font=('times', 17, ' bold ') )
head2.grid(row=0,column=0)

head1 = tk.Label(frame1, text="For Already Registered", fg="black",bg="#3ece48"
,font=('times', 17, ' bold ') )
head1.place(x=0,y=0)
```

```
lbl = tk.Label(frame2, text="Enter ID",width=20 ,height=1 ,fg="black" ,bg="#00aeff"
,font=('times', 17, ' bold ') )
lbl.place(x=80, y=55)

txt = tk.Entry(frame2,width=32 ,fg="black",font=('times', 15, ' bold '))
txt.place(x=30, y=88)
```

```
lbl2 = tk.Label(frame2, text="Enter Name", width=20 , fg="black" , bg="#00aeff", font=('times',
17, ' bold '))
lbl2.place(x=80, y=140)

txt2 = tk.Entry(frame2,width=32 , fg="black", font=('times', 15, ' bold ') )
txt2.place(x=30, y=173)

message1 = tk.Label(frame2, text="1)Take Images >>> 2)Save Profile" ,bg="#00aeff"
,fg="black" ,width=39 ,height=1, activebackground = "yellow" ,font=('times', 15, ' bold '))
message1.place(x=7, y=230)

message=tk.Label(frame2,text="",bg="#00aeff",fg="black",width=39,height=
1, activebackground = "yellow" ,font=('times', 16, ' bold '))
message.place(x=7, y=450)

lbl3 = tk.Label(frame1, text="Attendance",width=20, fg="black", bg="#00aeff", height=1
,font=('times', 17, ' bold '))
lbl3.place(x=100, y=115)

res=0
exists = os.path.isfile("StudentDetails\StudentDetails.csv")
if exists:
   with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:
      reader1 = csv.reader(csvFile1)
      for l in reader1:
         res = res + 1
   res = (res // 2) - 1
   csvFile1.close()
else:
   res = 0
message.configure(text='Total Registrations till now : '+str(res))
```

Face Detection

```
def TrackImages():
  check_haarcascadefile()
  assure_path_exists("Attendance/")
  assure_path_exists("StudentDetails/")
```

```
    for k in tv.get_children():
        tv.delete(k)
    msg = ''
    i = 0
    j = 0
    recognizer = cv2.face.LBPHFaceRecognizer_create() # cv2.createLBPHFaceRecognizer()
    exists3 = os.path.isfile("TrainingImageLabel\Trainner.yml")
    if exists3:
        recognizer.read("TrainingImageLabel\Trainner.yml")
    else:
        mess._show(title='Data Missing', message='Please click on Save Profile to reset
        data!!') return
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath);

    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names = ['Id', '', 'Name', '', 'Date', '', 'Time']
    exists1 = os.path.isfile("StudentDetails\StudentDetails.csv")
    if exists1:
        df = pd.read_csv("StudentDetails\StudentDetails.csv")
    else:
        mess._show(title='Details Missing', message='Students details are missing, please
check!')
        cam.release()
        cv2.destroyAllWindows()
        window.destroy()
    while True:
        ret, im = cam.read()
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
```

```
        faces = faceCascade.detectMultiScale(gray, 1.2, 5)
        for (x, y, w, h) in faces:
            cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)
            serial, conf = recognizer.predict(gray[y:y + h, x:x + w])
            if (conf < 50):
                ts = time.time()
                date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
                timeStamp =
            datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S') aa =
            df.loc[df['SERIAL NO.'] == serial]['NAME'].values
            ID = df.loc[df['SERIAL NO.'] == serial]['ID'].values
            ID = str(ID)
            ID = ID[1:-1]
            bb = str(aa)
            bb = bb[2:-2]
            attendance = [str(ID), '', bb, '', str(date), '', str(timeStamp)]

        else:
            Id = 'Unknown'
            bb = str(Id)
        cv2.putText(im, str(bb), (x, y + h), font, 1, (255, 255, 255),
    2) cv2.imshow('Taking Attendance', im)
    if (cv2.waitKey(1) == ord('q')):
        break
ts = time.time()
date =
datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y') exists
= os.path.isfile("Attendance\Attendance_" + date + ".csv") if
exists:
    with open("Attendance\Attendance_" + date + ".csv", 'a+') as
        csvFile1: writer = csv.writer(csvFile1)
        writer.writerow(attendance)
    csvFile1.close()
else:
    with open("Attendance\Attendance_" + date + ".csv", 'a+') as
        csvFile1: writer = csv.writer(csvFile1)
        writer.writerow(col_names)
        writer.writerow(attendance)
    csvFile1.close()
with open("Attendance\Attendance_" + date + ".csv", 'r') as csvFile1:
```

```
    reader1 = csv.reader(csvFile1)
    for lines in reader1:
        i = i + 1
        if (i > 1):
            if (i % 2 != 0):
                iidd = str(lines[0]) + ' '
                tv.insert('', 0, text=iidd, values=(str(lines[2]), str(lines[4]),
    str(lines[6]))) csvFile1.close()
    cam.release()
    cv2.destroyAllWindows()
```

Function

```
def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)
```

```
##########################################################################

def tick():
    time_string = time.strftime('%H:%M:%S')
    clock.config(text=time_string)
    clock.after(200,tick)

##########################################################################

def contact():
    mess._show(title='Contact us', message="Please contact us on :
'praveenk.kv2pondy@gmail.com' ")

##########################################################################

def check_haarcascadefile():
    exists = os.path.isfile("haarcascade_frontalface_default.xml")
    if exists:
        pass
    else:
        mess._show(title='Some file missing', message='Please contact us for
        help') window.destroy()

##########################################################################

def save_pass():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile("TrainingImageLabel\psd.txt")
    if exists1:
        tf = open("TrainingImageLabel\psd.txt", "r")
```

```
        key = tf.read()
    else:
        master.destroy()
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new password below',
show='*')
        if new_pas == None:
            mess._show(title='No Password Entered', message='Password not set!! Please try
again')
        else:
            tf = open("TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was registered
successfully!!')
            return
    op = (old.get())
    newp= (new.get())
    nnewp = (nnew.get())
    if (op == key):
        if(newp == nnewp):
            txf = open("TrainingImageLabel\psd.txt", "w")
            txf.write(newp)
        else:
            mess._show(title='Error', message='Confirm new password again!!!')
            return
    else:
        mess._show(title='Wrong Password', message='Please enter correct old
        password.') return
    mess._show(title='Password Changed', message='Password changed
    successfully!!') master.destroy()

##################################################################################

def change_pass():
    global master
    master = tk.Tk()
    master.geometry("400x160")
    master.resizable(False,False)
    master.title("Change Password")
```

```
    master.configure(background="white")
    lbl4 = tk.Label(master,text=' Enter Old Password',bg='white',font=('times', 12, ' bold '))
    lbl4.place(x=10,y=10)
    global old
    old=tk.Entry(master,width=25 ,fg="black",relief='solid',font=('times', 12, ' bold
    '),show='*') old.place(x=180,y=10)
    lbl5 = tk.Label(master, text=' Enter New Password', bg='white', font=('times', 12, ' bold
    ')) lbl5.place(x=10, y=45)
    global new
    new = tk.Entry(master, width=25, fg="black",relief='solid', font=('times', 12, ' bold
'),show='*')
    new.place(x=180, y=45)
    lbl6 = tk.Label(master, text='Confirm New Password', bg='white', font=('times', 12, ' bold
    ')) lbl6.place(x=10, y=80)
    global nnew
    nnew = tk.Entry(master, width=25, fg="black", relief='solid',font=('times', 12, ' bold
'),show='*')
    nnew.place(x=180, y=80)
    cancel=tk.Button(master,text="Cancel", command=master.destroy ,fg="black" ,bg="red"
,height=1,width=25 , activebackground = "white" ,font=('times', 10, ' bold '))
cancel.place(x=200, y=120)
    save1 = tk.Button(master, text="Save", command=save_pass, fg="black", bg="#3ece48",
height = 1,width=25, activebackground="white", font=('times', 10, ' bold '))
save1.place(x=10, y=120)
    master.mainloop()

#################################################################################

def psw():
    assure_path_exists("TrainingImageLabel/")
    exists1 = os.path.isfile("TrainingImageLabel\psd.txt")
    if exists1:
        tf = open("TrainingImageLabel\psd.txt", "r")
        key = tf.read()
    else:
        new_pas = tsd.askstring('Old Password not found', 'Please enter a new password below',
show='*')
        if new_pas == None:
                mess._show(title='No Password Entered', message='Password not set!! Please try
```

```
again')
        else:
            tf = open("TrainingImageLabel\psd.txt", "w")
            tf.write(new_pas)
            mess._show(title='Password Registered', message='New password was registered
successfully!!')
```

```
            return
        password = tsd.askstring('Password', 'Enter Password', show='*')
        if (password == key):
            TrainImages()
        elif (password == None):
            pass
        else:
            mess._show(title='Wrong Password', message='You have entered wrong password')

###########################################################################

#

def clear():
    txt.delete(0, 'end')
    res = "1)Take Images  >>> 2)Save Profile"
    message1.configure(text=res)


def clear2():
    txt2.delete(0, 'end')
    res = "1)Take Images  >>> 2)Save Profile"
    message1.configure(text=res)

###########################################################################

def TakeImages():
    check_haarcascadefile()
    columns = ['SERIAL NO.', '', 'ID', '', 'NAME']
    assure_path_exists("StudentDetails/")
    assure_path_exists("TrainingImage/")
    serial = 0
    exists = os.path.isfile("StudentDetails\StudentDetails.csv")
    if exists:
```

```
        with open("StudentDetails\StudentDetails.csv", 'r') as csvFile1:
            reader1 = csv.reader(csvFile1)
            for l in reader1:
```

```
            serial = serial + 1
        serial = (serial // 2)
        csvFile1.close()
    else:
        with open("StudentDetails\StudentDetails.csv", 'a+') as
            csvFile1: writer = csv.writer(csvFile1)
            writer.writerow(columns)
            serial = 1
        csvFile1.close()
    Id = (txt.get())
    name = (txt2.get())
    if ((name.isalpha()) or (' ' in name)):
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector = cv2.CascadeClassifier(harcascadePath)
        sampleNum = 0
        while (True):
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.3, 5)
            for (x, y, w, h) in faces:
                cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
                # incrementing sample number
                sampleNum = sampleNum + 1
                # saving the captured face in the dataset folder TrainingImage
                cv2.imwrite("TrainingImage\ " + name + "." + str(serial) + "." + Id + '.' +
str(sampleNum) + ".jpg",
                            gray[y:y + h, x:x + w])
                # display the frame
                cv2.imshow('Taking Images', img)
            # wait for 100 miliseconds
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
            # break if the sample number is morethan 100
            elif sampleNum > 100:
```

```
                break
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Taken for ID : " + Id
        row = [serial, '', Id, '', name]
        with open('StudentDetails\StudentDetails.csv', 'a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
        message1.configure(text=res)
    else:
        if (name.isalpha() == False):
            res = "Enter Correct name"
            message.configure(text=res)

###########################################################################

def TrainImages():
    check_haarcascadefile()
    assure_path_exists("TrainingImageLabel/")
    recognizer = cv2.face_LBPHFaceRecognizer.create()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    faces, ID = getImagesAndLabels("TrainingImage")
    try:
        recognizer.train(faces, np.array(ID))
    except:
        mess._show(title='No Registrations', message='Please Register someone
        first!!!') return
    recognizer.save("TrainingImageLabel\Trainner.yml")
    res = "Profile Saved Successfully"
    message1.configure(text=res)
    message.configure(text='Total Registrations till now : ' + str(ID[0]))

###########################################################################

def getImagesAndLabels(path):
    # get the path of all the files in the folder
```

```
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    # create empth face list
    faces = []
    # create empty ID list
    Ids = []
    # now looping through all the image paths and loading the Ids and the
    images for imagePath in imagePaths:
        # loading the image and converting it to gray scale
        pilImage = Image.open(imagePath).convert('L')
        # Now we are converting the PIL image into numpy array
        imageNp = np.array(pilImage, 'uint8')
        # getting the Id from the image
        ID = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(ID)
    return faces, Ids

###########################################################################################

def TrackImages():
    check_haarcascadefile()
    assure_path_exists("Attendance/")
    assure_path_exists("StudentDetails/")
    for k in tv.get_children():
        tv.delete(k)
    msg = ''
    i = 0
    j = 0
    recognizer = cv2.face.LBPHFaceRecognizer_create() # cv2.createLBPHFaceRecognizer()
    exists3 = os.path.isfile("TrainingImageLabel\Trainner.yml")
    if exists3:
        recognizer.read("TrainingImageLabel\Trainner.yml")
    else:
        mess._show(title='Data Missing', message='Please click on Save Profile to reset
        data!!') return
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath);
```

```
    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names = ['Id', '', 'Name', '', 'Date', '', 'Time']
    exists1 = os.path.isfile("StudentDetails\StudentDetails.csv")
    if exists1:
        df = pd.read_csv("StudentDetails\StudentDetails.csv")
    else:
        mess._show(title='Details Missing', message='Students details are missing, please
check!')
        cam.release()
        cv2.destroyAllWindows()
        window.destroy()
    while True:
        ret, im = cam.read()
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(gray, 1.2, 5)
        for (x, y, w, h) in faces:
            cv2.rectangle(im, (x, y), (x + w, y + h), (225, 0, 0), 2)
            serial, conf = recognizer.predict(gray[y:y + h, x:x + w])
            if (conf < 50):
                ts = time.time()
                date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
                timeStamp =
                datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S') aa =
                df.loc[df['SERIAL NO.'] == serial]['NAME'].values
                ID = df.loc[df['SERIAL NO.'] == serial]['ID'].values
                ID = str(ID)
                ID = ID[1:-1]
                bb = str(aa)
                bb = bb[2:-2]
                attendance = [str(ID), '', bb, '', str(date), '', str(timeStamp)]

            else:
                Id = 'Unknown'
                bb = str(Id)
            cv2.putText(im, str(bb), (x, y + h), font, 1, (255, 255, 255), 2)
        cv2.imshow('Taking Attendance', im)
        if (cv2.waitKey(1) == ord('q')):
```

```
            break
    ts = time.time()
```

```
date =
datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y') exists
= os.path.isfile("Attendance\Attendance_" + date + ".csv") if
exists:
   with open("Attendance\Attendance_" + date + ".csv", 'a+') as
      csvFile1: writer = csv.writer(csvFile1)
      writer.writerow(attendance)
   csvFile1.close()
else:
   with open("Attendance\Attendance_" + date + ".csv", 'a+') as
      csvFile1: writer = csv.writer(csvFile1)
      writer.writerow(col_names)
      writer.writerow(attendance)
   csvFile1.close()
with open("Attendance\Attendance_" + date + ".csv", 'r') as
   csvFile1: reader1 = csv.reader(csvFile1)
   for lines in reader1:
      i = i + 1
      if (i > 1):
         if (i % 2 != 0):
            iidd = str(lines[0]) + ' '
            tv.insert('', 0, text=iidd, values=(str(lines[2]), str(lines[4]), str(lines[6])))
csvFile1.close()
cam.release()
cv2.destroyAllWindows()
```

Attendance Table

```
tv= ttk.Treeview(frame1,height =13,columns =
('name','date','time')) tv.column('#0',width=82)
tv.column('name',width=130)
tv.column('date',width=133)
tv.column('time',width=133)
tv.grid(row=2,column=0,padx=(0,0),pady=(150,0),columnspan=4
) tv.heading('#0',text ='ID')
tv.heading('name',text ='NAME')
tv.heading('date',text ='DATE')
tv.heading('time',text ='TIME')
```

6. TESTING

Unit Testing

During the implementation of the system each module of the system was tested separately to uncover errors within its boundaries. User interface was used as a guide in the process.

Module Testing

A module is composed of various programs related to that module. Module testing is done to check the module functionality and interaction between units within a module. It checks the functionality of each program with relation to other programs within the same module. It then tests the overall functionality of each module.

Integration Testing

Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. The objective is to take unit-tested module and build a program structure that has been dictated by design.

System Testing

Testing is a set of activities that can be planned in advance and conducted systematically. The proposed system is tested in parallel with the software that consists ofits own phases of analysis, implementation, testing and maintenance. Following are the tests conducted on the system.

Acceptance Testing

The software has been tested with the realistic data given by the client and produced fruitful results. The client satisfying all the requirements specified by them has also developed the software within the time limitation specified. A demonstration has been given to the client and the end-user giving all the operational features.

Implementation Phase

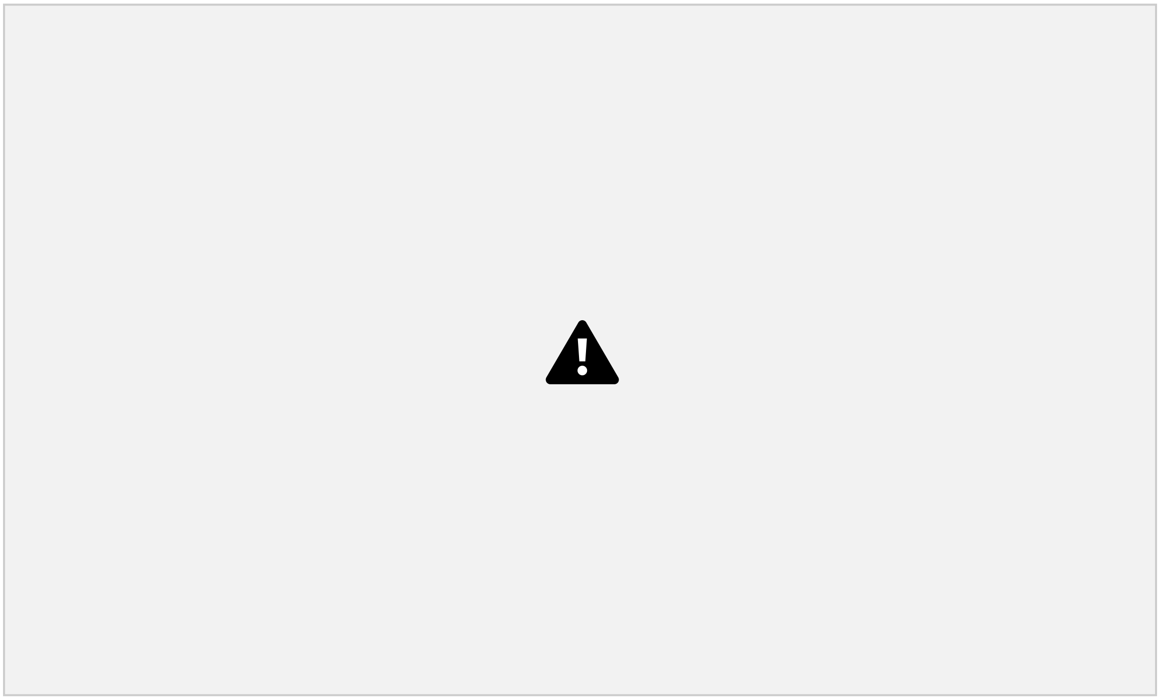
The implementation is the final and important phase. It involves user training, system testing and successful running of the developed system. The users test the developed system when changes are made according to the needs. The testing phase involves the testing of the developed system using various kinds of data. An elaborate testing of data is prepared and system is tested using the tests data.

Implementation is the stage where theoretical design turned into a working system. Implementation is planed carefully to propose system to avoid unanticipated problems. Many preparations involved before and during the implementation of proposed system. The system needed to be plugged in to the organization's network then it could be accessed from anywhere, after a user logins into the portal. The tasks that had to be done to implement the system were to create the database tables in the organization database domain. Then the administrator was granted his role so that the system could be accessed.

The next phase in the implementation was to educate the system. A demonstration of all the functions that can be carried out by the system was given to examination department person, who will make extensive use of the system.

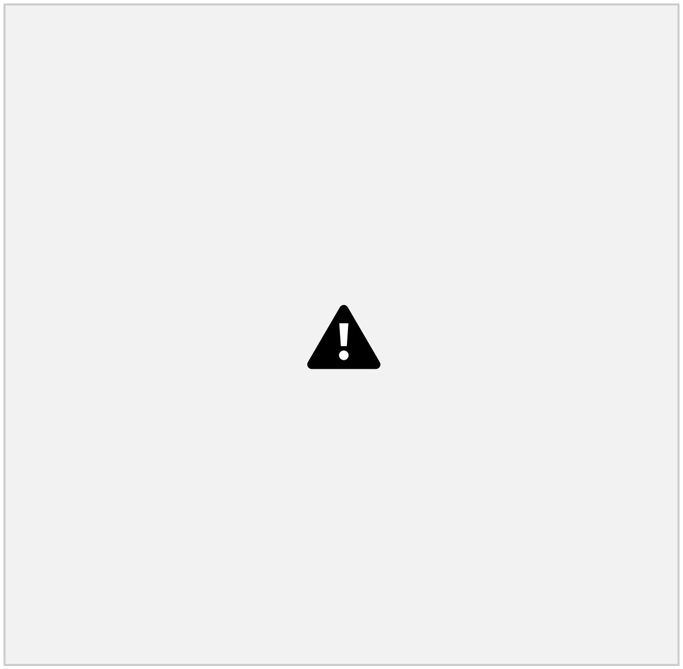
7. OUTPUT SCREEN

Front Page



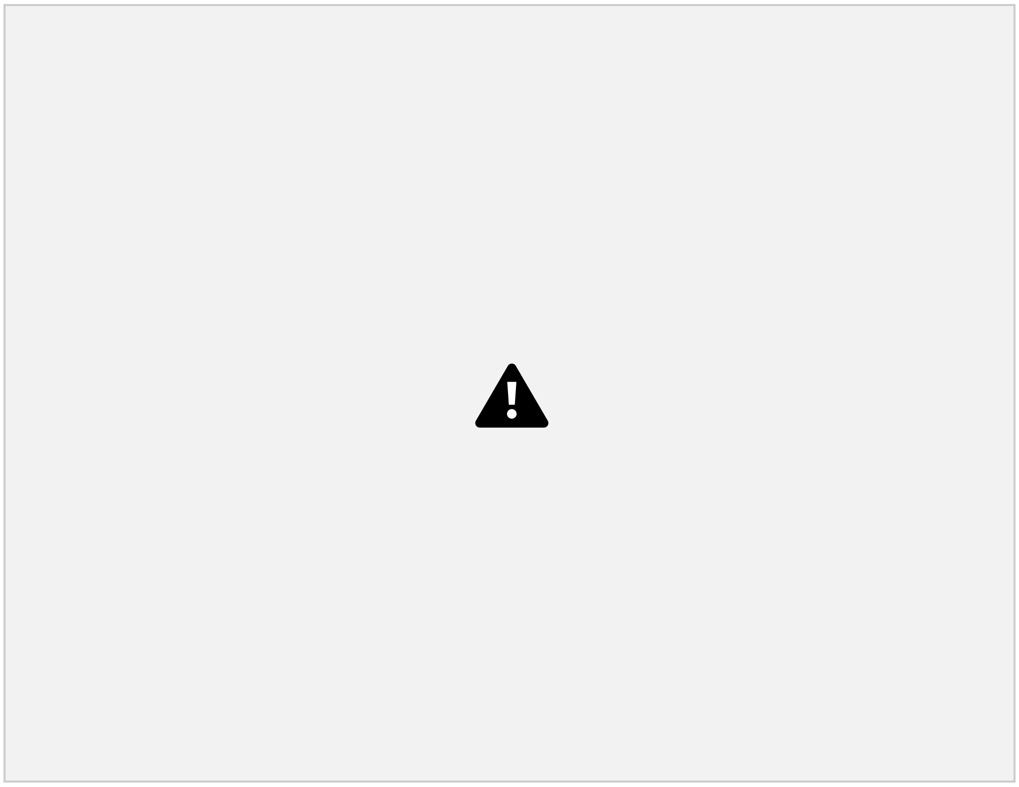
It is front end of out Application. It consists of two section, One for new registration and for already registered. Total number of registered student is shown below. It provide an clean and clear user interface

New Registration



This is our UI for New Registration of student, It has text box for Student's ID and Name. Next process is to take images and save profile of student. The Total registration till now of student is shown below for future references

Already Registered

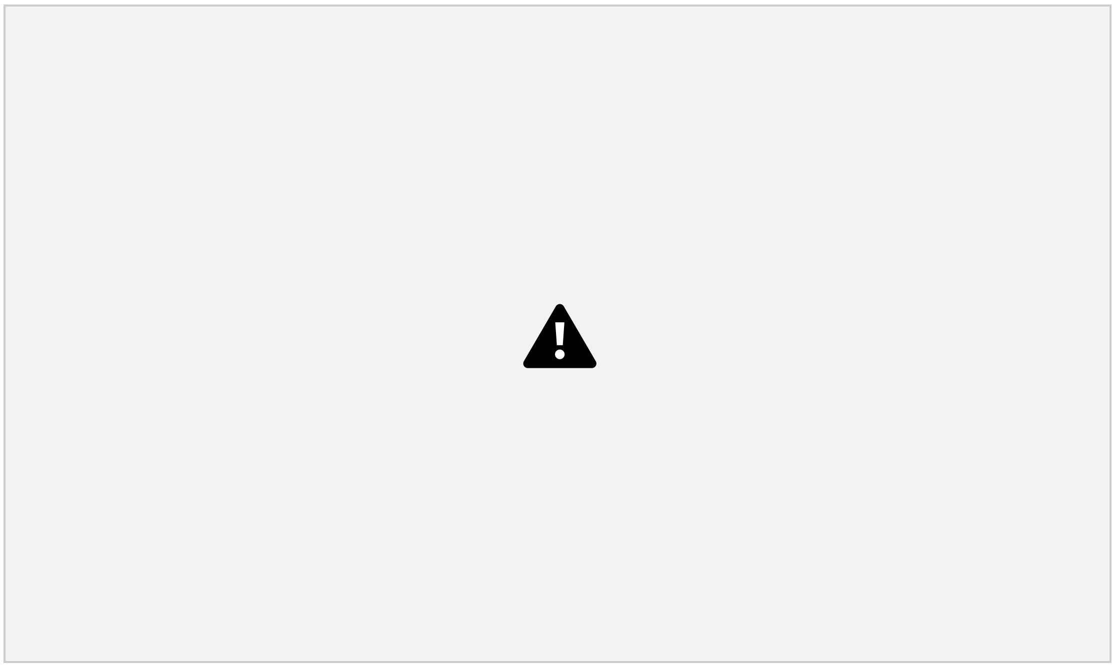


The next part of application is to take attendance. This part consists of two buttons one for Take attendance which triggeres the camera to track student’s faces and mark attendance for them. If needed we can close the camera by using quit button

Face Registration

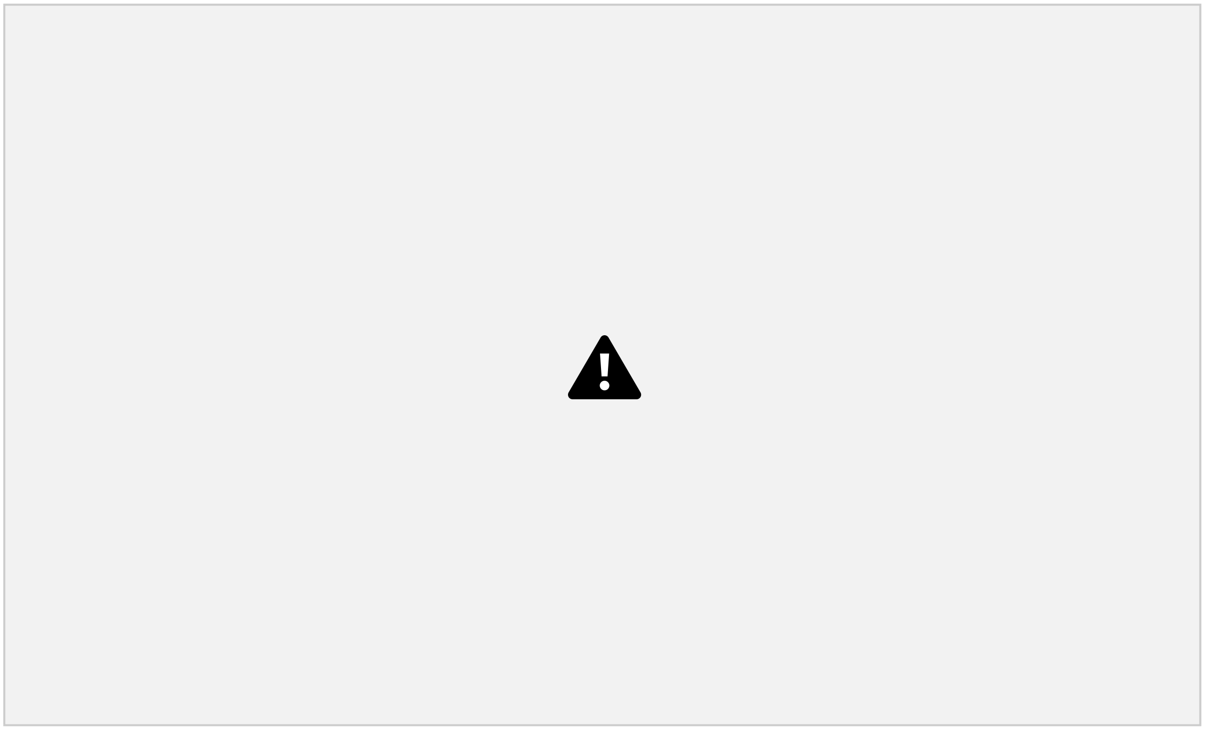
This screenshot shows the first process of our application, that is to register the student face. Before that we face to enter the student details such as student ID and Name. Then after the face detected the profile saved is prompted in the UI

Attendance Marking



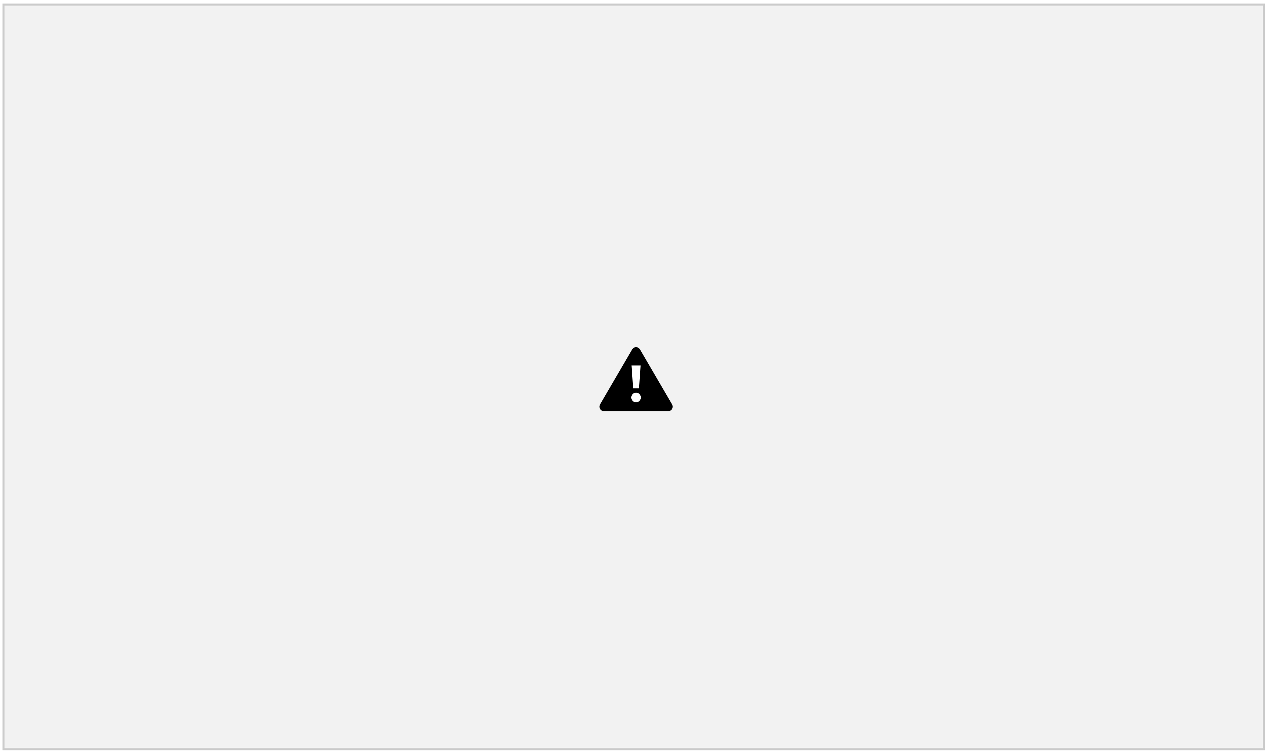
Here is the actual part of our application, That is to mark the attendace basedd on their faces. The face is tracked and labed with their name for regfernce to admin

View Attendance



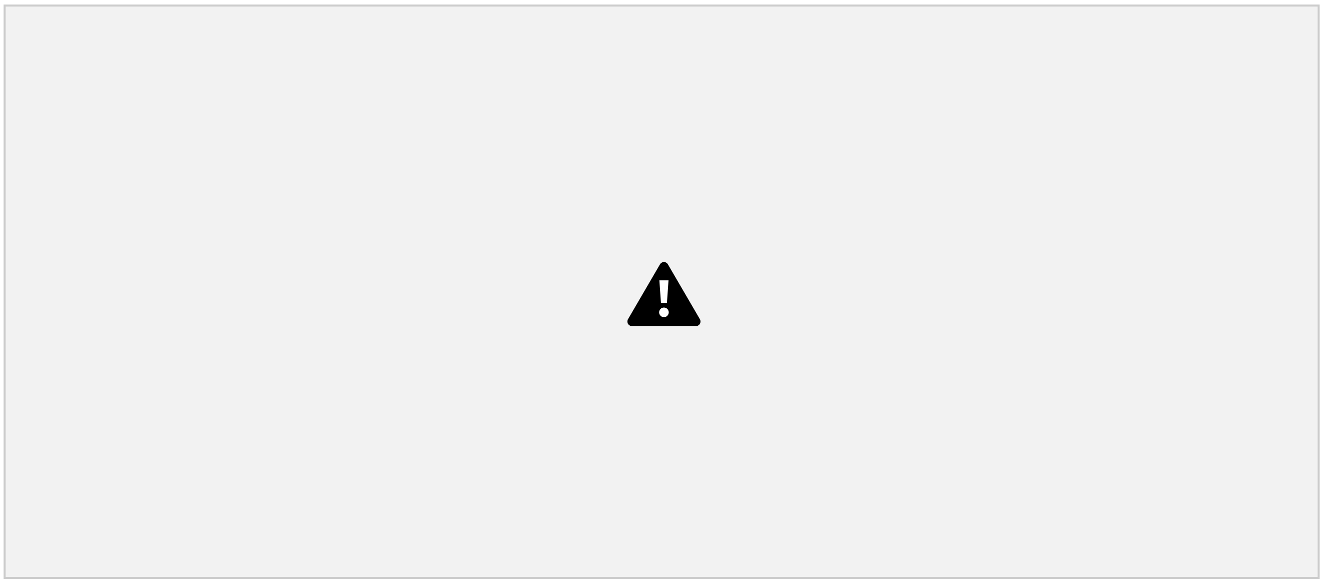
After all process the attendance of the student is marked and shown in the UI with that student in time. It list out all the student who entered into class

Password



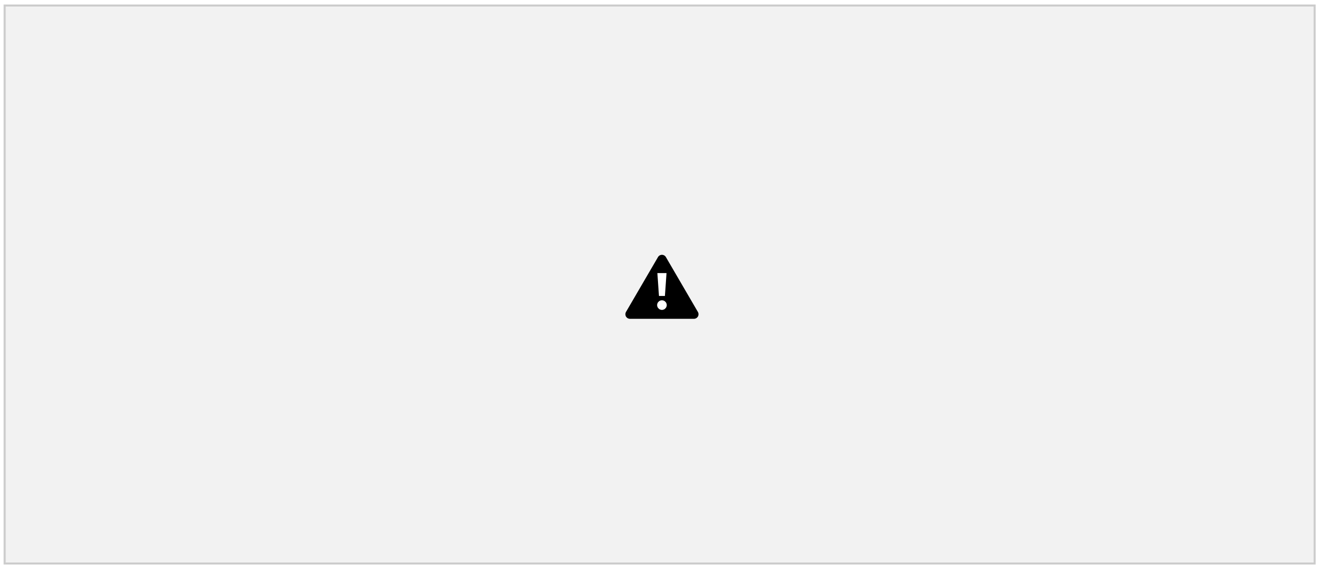
In today’s world security is top priority so our application provide an decent level of security. That protects student from entering invalid data to system or student may change the details of them. This can be avoided using a password protection that is shown in above screenshot

Student Details Csv File



The details of student entered in previous part is stored in an CSV file. From here we can see the overall registered student in that application. For now the basic details of student is mentioned here. It can also be exported to Excel sheet. Now we only given only 4 student details. But we can add more number of students

Attendance Sheet of Class



After all the process of the application, now its time to generate the attendance report of the specific class. In this sheet the student ID and Student name is marked and the student in time to class is also marked.

8. CONCLUSION

It has been a great pleasure for our team to work on this exciting project. This project proved good for us to as it provided practical knowledge of programming in

PYHTON and XML, but also about all handling procedure related with **“ATTENDANCE MONITORING USING FACE RECOGNITION”.** Application system has been developed as the best flexible and efficient project within the available resource and time.

The objective of this project was to build a program for maintaining the attendance of all the students and match details for a attendance monitoring using face recognition project which helps faculties and students to save a lot of time in taking attendance taken in various classes or periods. Our project provides students to get attendace from anywhere in class

It will provide the facility so the admin so that they can keep tracks of all the students attendances. The security of the system is also one of the prime concerns.

As the many things are computerized day to day, to reduced human error and to increase the efficiency. This project will provide better opportunities and guidance in future to develop projects independently. The Face Detection algorithm has become a major factor in today’s world, thus the Attendance monitoring system will helpful to the teachers and students.

It is an local system that may not have all the capabilities that are required for a attendance monitoring using face recognition task yet, but it is built over a solid structure that will allow to keep progressing with the conviction that what has been done, has been done in the best way.

REFERENCE

REFERENCE

[1] https://www.researchgate.net/publication/326986115_Face_Detection_and_Recognition_Student_Attendance_System

[2] Wendy Boggs, Michael Boggs- Mastering UML with Rational Rose- 2002

[3] Object oriented system development, Ali Bhrami-Edition 2008.

[4] https://www.superdatascience.com/blogs/opencv-face-recognition

[5] IEEE Xplore, Face Recognition, Edition 1

[6] Nithya C., Ramya Bharathi. M., Santhini. M., Sowmya. R., "Face Recognition based Automatic Attendance Management System," INTERNA-TIONAL JOURNAL OF ENGINEERING RESEARCH and TECHNOLOGY (IJERT), vol. 8, no. 08, 2020.

[7] Mohd Suhairi Md Suhaimin, Mohd Hanafi Ahmad Hijazi, Chung Seng Kheau, Chin Kim On, "Real-time mask detection and face recognition using eigenfaces and local binary pattern histogram for attendance system," Bulletin of Electrical Engineering and Informatics, vol. 10, no. 2, p. p1105 1113, April 2021.

Websites:

[1] www.tutorialpoints.com

[2] www.w3schools.com

[3] www.freetutes.com

[4] www.geeksforgeeks.com