****

# 

# Muhammad Adnan Bin Zafar

# 172100

# Muhammad Saqib

# 172106

# Muhammad Shaukat

# 172054

# Smart Attendance using Face Recognition

# Supervisor: Dr. Mateen

# 

# Department of Computer Science

# Air University, Multan

# Dedications

*This project is dedicated to our beloved parents and respected teachers for their never ending moral support and prayers which always acted as a catalyst in our academic life.*

# Acknowledgement

*We would like to express our heartfelt gratitude to a number of people and organisations for their unwavering support during our graduate studies. First and foremost, we want to express our sincere gratitude to Professor Mateen, our project supervisor, for his enthusiasm, patience, insightful comments, helpful knowledge, practical advice, and never-ending ideas, all of which have greatly aided us during our project. We were able to successfully complete this project thanks to his extensive knowledge, extensive experience, and technical skills in Smart Attendance. This project would not be possible without his support and guidance..We couldn't have imagined having a far better supervisor in our study.*

Table of Contents

[Dedications 2](#_Toc24467411)

[Acknowledgement 2](#_Toc24467412)

[List of Tables 5](#_Toc24467413)

[List of Figures 6](#_Toc24467414)

[Abstract 9](#_Toc24467415)

[Chapter 1: Introduction 2](#_Toc24467416)

[1.1 Background 2](#_Toc24467417)

[1.2 Objectives 3](#_Toc24467418)

[1.3 Scope 3](#_Toc24467419)

[1.4 Problem Statement 3](#_Toc24467420)

[Summary 4](#_Toc24467421)

[Chapter 2: Literature Review 6](#_Toc24467422)

[Chapter 3: Planning and Methodology 8](#_Toc24467423)

[3.1 Programming methodology 10](#_Toc24467424)

[3.2 Process Methodology 10](#_Toc24467425)

[3.3 system implementation 11](#_Toc24467426)

[3.4 system architecture 12](#_Toc24467426)

[3.5 system internal components 12](#_Toc24467426)

[3.6 tools and technology 12](#_Toc24467426)

[Summary 15](#_Toc24467427)

[Chapter 4: System Specification 16](#_Toc24467428)

[4.1 existing system 17](#_Toc24467429)

[4.2 proposed system 17](#_Toc24467431)

[4.3 Requirement specifations 18](#_Toc24467432)

[4.3.1 functional requirements 18](#_Toc24467433)

[4.3.2 Non-Functional Requirements 18](#_Toc24467433)

[4.4 use case diagram 19](#_Toc24467436)

[4.5 process flow 37](#_Toc24467439)

[4.6 Actors 37](#_Toc24467439)

[4.7 Class Diagram 39](#_Toc24467447)

[4.8 graphical user interface 40](#_Toc24467447)

[4.8 SYSTEM REQUIREMENTS and SOFTWARE INTERFACES 43](#_Toc24467444)

[Summary 44](#_Toc24467451)

[Chapter 5: System Design 45](#_Toc24467452)

[5.1 Design Goals 46](#_Toc24467453)

[5.2 System Architecture Diagram 4](#_Toc24467454)7

[5.3 sequence diagram 48](#_Toc24467455)

[5.4 activity diagram 49](#_Toc24467458)

[5.5 system flow diagram 53](#_Toc24467460)

[5.7 Data Dictionary 50](#_Toc24467459)

[5.8 Deployment Diagram 55](#_Toc24467464)

[Summary 56](#_Toc24467451)

[Chapter 6: CODING 58](#_Toc24467452)

[6.1 main screen 59](#_Toc24467453)

[6.2 admin side 62](#_Toc24467453)

[6.3 teacher side 71](#_Toc24467453)

[6.4 invigilator side 75](#_Toc24467453)

[6.5 registration 79](#_Toc24467453)

[6.6 features 82](#_Toc24467453)

[Chapter 7: software testing 117](#_Toc24467452)

[7.1 gui testing 118](#_Toc24467453)

[7.2 usability testing 120](#_Toc24467453)

[7.3 exception handling testing 121](#_Toc24467453)

[7.4 performance testing 121](#_Toc24467453)

[Chapter 8: conclusion 123](#_Toc24467452)

[8.1 conclusion 124](#_Toc24467453)

[8.2 future enhancement 124](#_Toc24467453)

[Reference 125](#_Toc24467473)

[Glossary 128](#_Toc24467474)

# List of Tables

[Table 2. 1 Limitations](#_bookmark15) 7

[Table 4. 1 Login Use Case](#_bookmark15) 22

[Table 4. 2 Verification Use Case](#_bookmark15) 23

[Table 4. 3 Registration Use Case](#_bookmark15) 24

[Table 4. 4 Initiate Camera Use Case](#_bookmark15) 25

[Table 4. 5 Capturing at Different Angles Use Case](#_bookmark15) 26

[Table 4. 6 Entering Detail Use Case](#_bookmark15) 27

[Table 4. 7 Saved to Database Use Case](#_bookmark15) 28

[Table 4. 8 Initiate Attendance Module](#_bookmark15) 30

[Table 4. 9 Initiate Camera](#_bookmark15) 31

[Table 4. 10 Face Comparison](#_bookmark15) 32

[Table 4. 11 Face Reorganization](#_bookmark15) 33

[Table 4. 12 Mark Present](#_bookmark15) 34

[Table 4. 13 Mark Absent](#_bookmark15) 35

[Table 4. 14 Report](#_bookmark15) 36

[Table 5. 1 Data Dictionary](#_bookmark15) 55

[Table 7. 1 Login screen test case](#_bookmark15) 118

[Table 7. 2 Student registration test case](#_bookmark15) 119

[Table 7. 3 Home page test case](#_bookmark15) 119

[Table 7. 4 Student Attendance test case](#_bookmark15) 120

[Table 7. 5 Invigilation system test case](#_bookmark15) 120

[Table 7. 6 login test case](#_bookmark15) 120

[Table 7. 7 insert, update test case](#_bookmark15) 121

[Table 7. 8 System performance test case](#_bookmark15) 122

# List of Figures

Figure 3. 1 Process Methodology 11

Figure 3. 2 Face Coordinates 13

Figure 3. 3 Gantt Chart 14

Figure 4. 1 Main Use Case Diagram of whole system 19

Figure 4. 2 Main Use Case Diagram of Registration of Students 20

Figure 4. 3 Main Use Case Diagram of Invigilation of Students 21

Figure 4. 4 Login Use Case Diagram 22

Figure 4. 5 Verification Use Case Diagram 23

Figure 4. 6 Registration Use Case Diagram 24

Figure 4. 7 Initiate Camera Use Case Diagram 25

Figure 4. 8 Capture Image at different angle Use Case Diagram 26

Figure 4. 9 Entering detail Use Case Diagram 27

Figure 4. 10 Saved to database Use Case Diagram 28

Figure 4. 11 Use Case of Attendance of Students Diagram 29

Figure 4. 12 Initiate Attendance Diagram 30

Figure 4. 13 Initiate Camera Use Case Diagram 31

Figure 4. 14 Comparison Use Case Diagram 32

Figure 4. 15 Reorganization Use Case Diagram 33

Figure 4. 16 Mark Present Use Case Diagram 34

Figure 4. 17 Mark Absent Use Case Diagram 35

Figure 4. 18 Report Use Case Diagram 36

Figure 4. 19 Process flow diagram 37

Figure 4. 20 Class Diagram 39

Figure 4. 21 Main menu 40

Figure 4. 22 Admin panel 40

Figure 4. 23 Admin login 41

Figure 4. 24 Invigilator panel 41

Figure 4. 25 Teacher panel 42

Figure 5. 1 System Architecture Diagram 47

Figure 5. 2 Registration of Students Sequence Diagram 48

Figure 5. 3 Attendance of Student Sequence Diagram 49

Figure 5. 4 Admin Activity Diagram 50

Figure 5. 5 Teacher Activity Diagram 51

Figure 5. 6 Invigilator Activity Diagram 52

Figure 5. 7 Smart Attendance flow diagram 53

Figure 5. 8 Context Level Diagram 54

Figure 5. 9 Level 1 Data Flow Diagram 55

Figure 5. 10 Deployment Diagram 56

# Abstract

Face recognition technology has progressed significantly in the changing world. Smart Attendance with Real-Time Face Recognition may be a practical solution for managing student attendance systems on a day-to-day basis. Face recognition-based attendance system is a method of identifying a student's face for the purpose of taking attendance using face biometrics, high-definition display video, and other information technology. In my face recognition project, a computer machine will be able to quickly and accurately locate and identify human faces in photographs or videos recorded by a surveillance camera. For improving the performance of face recognition, a variety of algorithms and techniques have been developed, but the principle to be applied here is Deep Learning. This work proposes a replacement methodology during which attendance of every individual student during a class room is automatically updated during a database by analyzing their faces and comparing them with the predefined images by means of face recognition module. The proposed system is achieved an accuracy of 93% to 98% for face recognition.

Keywords: face recognition, Deep learning, biometrics, camera, invigilation

**CHAPTER 1**

**INTRODUCTION**

# Chapter 1: Introduction

SMART ATTENDANCE is an AI powered face recognition solution which uses Computer Vision and Machine Learning algorithms to mark the attendance of the workers or students of the organization. The system will work on face recognition where each student within the class are going to be photographed and their details are going to be stored during a server. The teacher can then record the attendance by just clicking some pictures of the classroom. The system will recognize the faces and verify the presence or absence of every student. The technology aims to include remarkable knowledge-based technological advances in recent years. Deep Learning is an intriguing area that allows a computer to coach itself by providing some datasets as input and then providing an acceptable output during testing using various learning algorithms.

In today's educational institutions, attendance is viewed as a critical factor for both students and teachers. With the introduction of deep learning technology, the machine can now automatically detect the students' attendance performance.

In general, a student's attendance scheme can be held in two separate ways, namely,

Manual Attendance System (MAS)

Automated Attendance System (AAS).

Manual Student Attendance Management is a method in which a teacher in charge of a specific subject must manually call the students' names and record their attendance.

Manual attendance will be perceived as a time-consuming operation, and it is likely that the teacher may forget someone, or that students may react to the absence of their friends several times. As a result, the issue occurs when we consider the conventional method of taking attendance in the classroom. To resolve all of these problems, we use the Automatic Attendance System (AAS), which consists of an application programme for tracking student attendance. Each classroom or department has its own user name and password.

Furthermore, since there is always the possibility of proxy attendance while marking attendance, we are developing an automated invigilation system that will automatically mark students' attendance as well as detect any suspicious gestures, resulting in effective invigilation. So its additional feature is invigilation (actually holding an eye on exam takers to make sure they aren't cheating).

## Background

*We use video call meetings instead of physical meetings because we know that face recognition technologies have become relevant in all fields as a result of Covid-19. However, we decide that instead of building a project that will mark student attendance using facial recognition technology and deep learning, we will create a project that will assist teachers in marking student attendance in order to save time and see records in a database. In order to begin invigilation, the invigilator must simply turn on the machine. The camera will begin recording and identify unusual movements (head movement, gaze detection). If the camera detects any suspicious activity, the buzzer will sound to warn the invigilator.*

## Objectives

* To develop SMART ATTENDANCE that is reliable, practical, and eliminates disturbance, deception and time wastes in attendance.
* To develop a system that will get the performance of the students without any deception.
* To improve the precision of identifying faces and suspicious movements as much as possible.

## Scope

Since a person's face offers a distinctive identification, it can be used to identify and verify the person's identity. Face recognition is a non-intrusive form of identifying someone. The chances of fake attendance and proxies can be minimised by using this method. In certain cases, face recognition using the Eigen face approach aids in the resolution of issues such as lightning and head pose. However, the system's accuracy is still lacking, owing to the face detection method's sensitivity to head tilt issues. We'll need to find a reliable face detection system. Other supervised methods in the framework can be very useful. Furthermore, we can simplify the system and make it more effective and sing multiple face detections to mark attendance of all identifiable faces in a single attempt makes the process more effective.

* Automated attendance of students will be done.
* Lecturer will activate the attendance using a desktop application.
* Video or images will be utilized to provide inputs to the system.

## Problem Statement

### Our educational institution's attendance scheme has several flaws. For example, there is a possibility of deception due to human physical limitations. Traditional attendance systems waste time and expose you to the possibility of proxy attendance. As a result, by automating this system, we will be able to reduce the risk of deception in invigilation and proxy attendance, as well as time wasting in the attendance system. As we develop the automated system, there will be some issues, such as not being able to recognise a face due to environmental factors such as low light, which could result in incorrect or no attendance. Problems will be considered when designing the method, and algorithms will be used to solve this problem. Another issue we'll face is facial changes over time, which we'll address using deep learning algorithms.

### Regarding the coronavirus scare, several government agencies and private businesses have suspended their use of the commonly used fingerprint biometric attendance system. So now is the time to update the device to a non-touch, infection-free face recognition attendance system. And, due to a fundamental issue with biometric participation, there are just a handful.

### Coronavirus spreads from human touch. If an employee touch the biometric attendance system with coronavirus and touch her surface, she is susceptible to getting infected from coronavirus.

### if finger is injured again a finger print scanner won't be able to scan the fingers.

### Biometric attendances devices take more time to mark the attendance

## Summary

This chapter is detailed description of introduction, background, scope. All the specifications are presented in this chapter with full detail.

**Chapter 2**

**Literature Review**

# Chapter 2: Literature Review

*This chapter will discuss the work that has already been completed on our project. We'll talk about the work we've done since our project has module attendance. We'll also explore the shortcomings or disadvantages in existing work, and then explain how our project will fix those limitations or drawbacks.*

## RFID Base Attendance system:

*RFID stands for radio frequency identification, and it is defined as a new concept of IOT (internet of things) that can assist in the upgrade of attendance systems (according to a 2019 study). The definition is similar to that of a barcode, but with a few improvements.*

### RFID Tag:

*A RFID tag is a type of electronic tag that uses radio waves to share information with an RFID reader. Antenna and integrated circuits are the two main parts of it. RFID passes and collects signals from antenna and integrated circuits. It has two main parts, each with its own purpose. Antenna receives radio waves, while integrated circuits process and store data.*

### RFID Reader:

*It's a computer that reads RFID tags and extracts data from them. Data from the tag to the RFID reader is actually moved or propagated by radio waves in RFID readers.*

### Working Mechanism RFID

*Tag charges itself after receiving a signal from the reader via antenna. The reaction is then transferred to the antenna by the Charged tag. The antenna recognises the data and sends it to the reader. Finally, the reader reads the directions and follows them.*

## Fingerprint base Attendance System:

*The main aim of the fingerprint-based attendance system is to simplify the current attendance system and remove the issues that plague the current system. In this scheme, the student or employee must first register his or her fingerprint, which is then stored in the database. When a student or employee scans his or her fingerprint, it is compared to a fingerprint that already exists in the database, and attendance is registered if the fingerprints match successfully.*

## Attendance through voice recognition:

*This initiative was created with the idea that workers who work outside of the system would benefit from it. There are two modules in it, one for voice recognition and the other for fingerprint recognition. This device monitors attendance through a mobile app. This section will cover the system's voice recognition technology. In this method, a person's voice is transformed to electrical signals, which are then converted into a voice print. Furthermore, these prints will be saved in the database so that they can be matched. So how does this technology work? First, some random text is shown on the system, and then the individual speaks those same characters. The person's voice will be compared to the current voices in the database, after which attendance will be recorded and the home screen will be shown again. If the voice of the individual speaking the random numbers does not fit any of the voices in the database, a message such as "Invalid voice" will appear on the screen.*

## Limitations or Drawbacks

*In an RFID-based attendance scheme, if another student has his friend's card, he or she will use it to mark his or her attendance, and if the same card is swiped twice, the next day's attendance of that student will be marked if the code is not written correctly. Fingerprint-based attendance is superior to paper-based attendance, which runs the risk of misplacing the register and fraud in marking the attendance of students who aren't even in class. However, documenting attendance takes a long time. VRT has automated the attendance system, but it still has some flaws, such as taking a long time to mark every student's attendance, particularly in classes with a large number of students. Furthermore, when there are other sounds present, it can sometimes trigger an error by failing to recognise the intended voice. Emotion Recognition from Skeletal Gestures employs deep learning technology, but precision is not guaranteed.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SNO | Author | Previous System | Problem | Solution |
| 1 | Ahmadh Rifai Kariapper | RFID base attendance system | Author described  paper based attendance create troubles like keeping the attendance sheet, forged documentation, unlawful acts by students etc. | Using Radio Frequency Identification technology helps overcome the trouble of forged documentation, unlawful activities etc. |
| 2 | Imran Anwar Ujan | Fingerprint Base Attendance System | Systems  are manual and there is chance of proxy attendance | Marking the  attendance through fingerprint lessens the chances  of proxy attendance |
| 3 | Benfano Soewito | Attendance through voice recognition | Present system is  time consuming and there is  no tool that can monitor working hours of  employee outside the  office | Voice Recognition Technology will save time and can monitor employees working time. |
| 4 | Partha Pratim Debnath | Detection of suspecious activity in class room | Illegal activities  occur during the exam which  remains undetected due to human  physical limitations | Detecting suspicious  behaviors by automating the invigilation system minimize unlawful activities happened in the examination hall |

**Table 2.1 Limitations**

## Proposed Solution

*Existing attendance programmes have advantages as well as drawbacks. So, in order to address the issues or weaknesses, we are designing an automated attendance system that utilises the efficient deep learning/machine learning technique. Attendance is recorded in this method when the camera identifies a face and compares it to a previously registered face in the database. Problems such as time waste, proxy attendance, and suspicious activities, among others, will be reduced as a result.*

**Chapter 3**

**Planning and Methodology**

# Chapter 3: Planning and Methodology

We'll talk about our project's methodology and flow in this chapter.

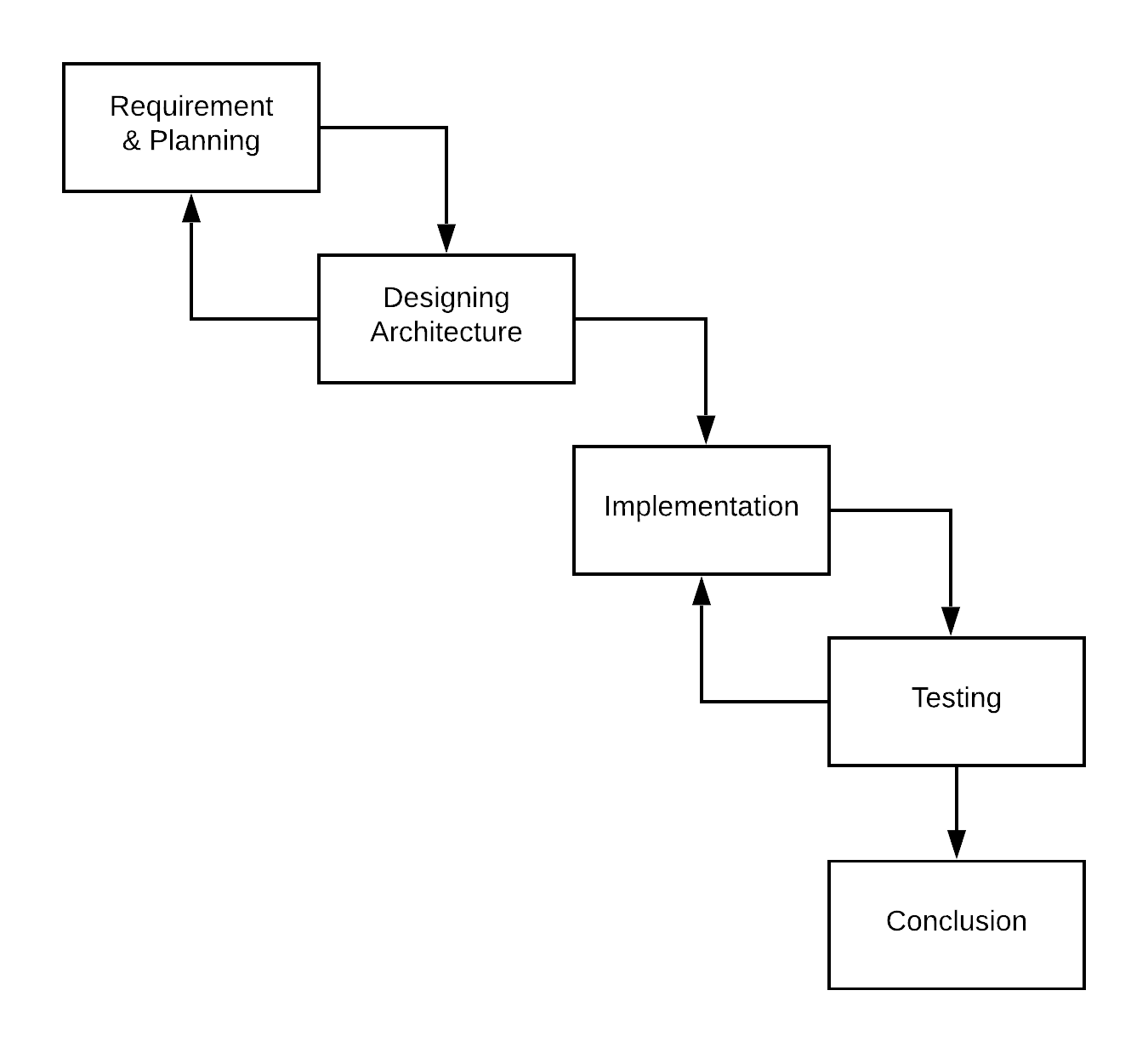
The systematic, theoretical analysis of the techniques used in a field of research is known as methodology. It entails a theoretical review of a body of methods and concepts associated with a field of study. We'll also take a close look at the design and modules used in the Smart Attendance Framework, as well as their implementation. We'll also go through the tools and technologies that were used in the construction of this framework, as well as the system's overall flow.

## Programming Methodology

Our framework is divided into two parts: the front end, which consists of a python-based GUI that serves as a client, and the backend, which consists of logic and is also based on Python and serves as a server. We also know that both languages cannot communicate directly with each other, so we used IPC (Inter Personal Communication) techniques with the zero library as a bridge to communicate between them. With the support of the Zero PC Library, Electron JS calls python functions and exchanges data via TCP.

## Process Methodology

We'll be using the conventional waterfall approach with some overlaps between a few phases as needed due to a lack of time and a lot of research and development work necessary to complete the project. Overall, We won't have time to go back and reconsider a move, so We'll take great care in each one and work from the top down to finish the project on time.

The measures We'll take during the process are depicted in the diagrams below.

**FIGURE 3.1: Process Methodology**

## System Implementation

In this chapter, we'll talk about how the system will be implemented in practise, including the framework of the system, its key components, and the methods and techniques that will be used to fully implement the system. This system is actually a desktop application whose purpose is to monitor student attendance and invigilate them. To accomplish its goal, our framework makes use of student data sets and deep learning/machine learning algorithms.

## System Architecture

The high level representation of our system's configuration will be addressed in system architecture. Our system is designed around five main components and has two main goals. The key components are Image Acquisition, Face identification, Feature extraction, Facial recognition, and Body movement detection, and the goals are attendance marking and invigilation of students. Our framework is a desktop application that, once installed, will take care of the majority of the work. The students' data will be obtained first and stored in a SQL database. The method uses deep learning algorithms to extract data from a database and perform operations on it. The model will be trained using a CNN (convolutional neural network).

## System Internal Components

Internal system components are the key pillars that enable the Smart Attendance System to achieve its goals..

### Face Detection

Face detection in a picture can be achieved using a number of deep learning techniques, but we used the multi-tasked cascaded convolutional neural network, or MTCNN. This network has a three-network structure that aids in the suggestion of facial landmarks.

### Feature Extraction

We use a neural network technique to remove the point from the face in this section of the method. FaceNet is a neural network that learns how to map face images to a compact Euclidean space where distances are proportional to how close two faces are. We may assume that two faces are more alike if the distance between them is smaller.

### Facial Recognition

FaceNet, a deep learning neural network, recognises an individual's face. FaceNet maximises the distances between two distinct images while minimising the distances between two identical images.

### Body Movement Detection

Eye gaze detection and head movement detection are two types of body movement detection. Geometric relations and facial landmarks are used to obtain the eye region. Then, using the obtained eye area, we can identify the gaze direction. Python, openCv, and the dlib library are used to achieve this. The Dlib library is also used to track head movement. In head movement detection, we take pictures of various head motions, train our model on them, and then use CNN to arrive at a conclusion.

## Tools and Technology

### Python:

It is a versatile programming language that adheres to object-oriented principles.

It aids in code clarity and white space use by writing logical code for both small

and large critical problems.

### Keras

Keras is a neural networking API. It's user-friendly, extensible, and, most importantly,

it's Python-friendly.

### Tensorflow

It is an artificial intelligence library that is used to build models for large-scale neural networks. It can also be used to make forecasts, learn, and discover new things.

### Dlib library:

In computer vision, it's popular to use the Dlib library to estimate position on 68 coordinates, such as X.Y. This includes pre-trained models that serve as facial detectors and map the points on a person's face in the manner shown below.

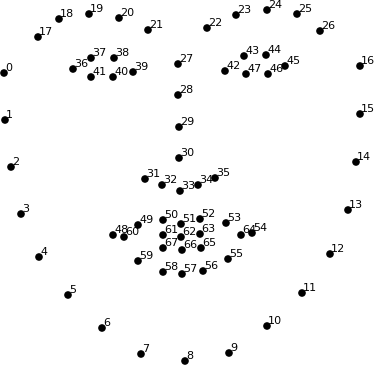


Figure 3.2: Face Coordinates

## System Flow:

We will use an incremental model to build this application because the requirements of the stakeholders will change. As a result, the entire system is divided into four stages.

**Phase 1:**

### In the first step, we will create a desktop application to collect student data, prepare data sets, and train the model.

### Phase 2:

### We'll be working on the attendance module and testing the qualified model during this process on the set of data

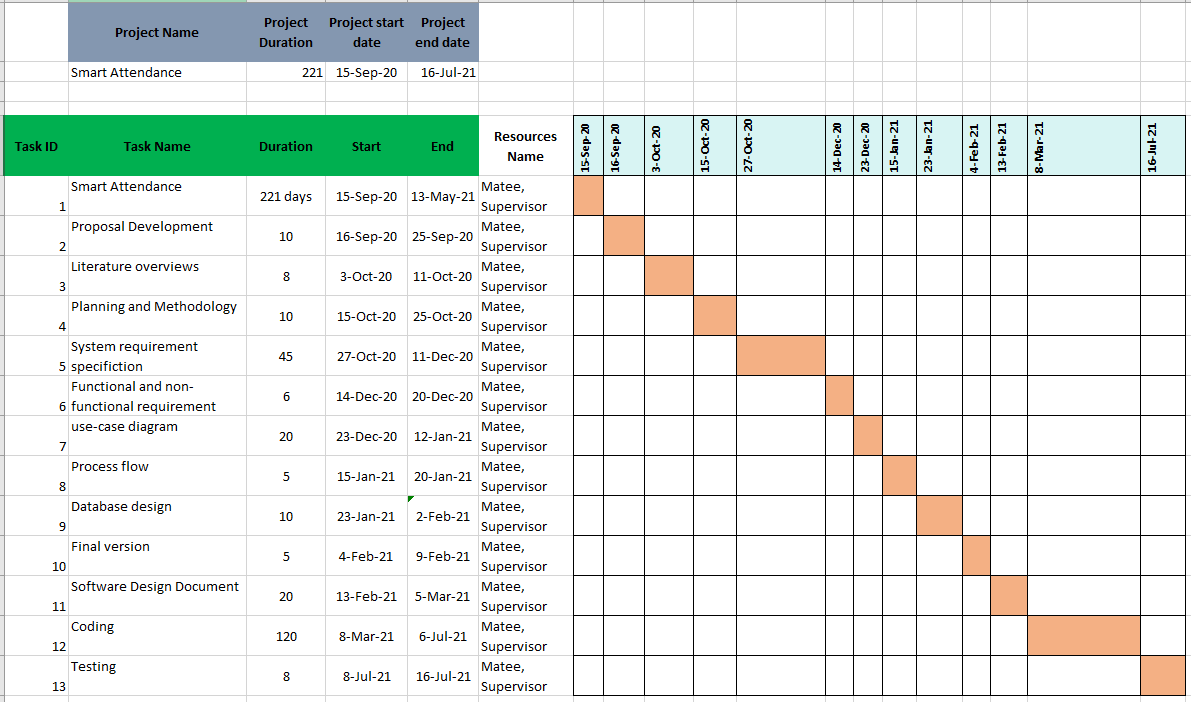
### Phase 3:

We will focus on detecting body movements and ensuring that the expected body parts movement is identified and recognised with greater accuracy in this process.

**Phase 4:**

We'll combine all of the modules and test the programme as a whole using a desktop application.

#### Process Gantt chart



**Figure 3. 3 Process Gantt Chart**

## Summary

Before beginning the project, develop a project plan that provides information about the deliverables, when they will begin, when they will end, and how they will be built. It's a plan that the project team makes in order to reach deadlines. Methodology is a crucial factor that describes how we can accomplish our objectives. Due to predefined specifications, a waterfall model is used in this project. In light of the above, this study established an automated attendance registering system using the Waterfall Software Development Life Cycle (SDLC) model, with the intention of creating a mobile-responsive and simple application for both lecturers and students.

**Chapter 4**

**System Specification**

# Chapter 4: System Specification

In this chapter, we will clearly define the weakness or disadvantage of the currently implemented attendance system, as well as how our system will address those limitations. We'll mostly talk about the needs of the stakeholders. Furthermore, we will build use case diagrams and use cases to assist in clearly understanding the needs of each particular stakeholder.

## Existing System:

Currently, the most automated attendance method we use is attendance via web application, and invigilation is human-based, with invigilators hired to invigilate students to prevent any deception. There are shortcomings in the current structure.

* + - Time consuming
    - Chances of deception
    - Chances of error in marking the attendance
    - Proxy attendance

## Proposed system:

We're automating the whole attendance system, which will fix all of the current system's issues. Basically, attendance will be recorded using a camera. The camera will identify a person's face, compare it to existing pictures of the student stored in the database, and if the match is accurate, the attendance will be recorded.

By having an autonomous system, this system would be able to solve all of the current system problems.

## Requirement Specifications

### Functional Requirements

* + - * User should be able to control student record
      * An unauthorized person should not be able to use the system
      * System should be attached to high definition camera
      * A person who has access to the system needs to login first and then use it
      * Changes made to the system should be error free

### Non-Functional Requirements

* + - * There should be maximum accuracy
      * The interface of the web application must be user-friendly
      * System should be efficient and effective
      * If at some time any change require it should be easy to implement that change in the project
      * System should be secure and performance should be better

### Student Requirement

* + - * Student should enter his/her details while registration
      * Student need to sit before the camera and camera will take 20 to 25 pictures of him/her.
      * Student should sit properly in the class in a way that his/her face will confront the camera

### Administration Requirement

* + - * Only the administrator will be able to make changes in the system if needed
      * He/she is responsible for seeing that the student enter his/her detail properly.
      * He/she is the only responsible to make changes in the data set
      * Administrator must be log in to register students

### Teaching Staff Requirement

* + - * Teacher should be log in to the system for starting attendance system.
      * Teacher should enter the details of the lecture he/she is going to give(may/may not be included).
      * Attendance report will be generated at the end of the lecture(may/may not be included).

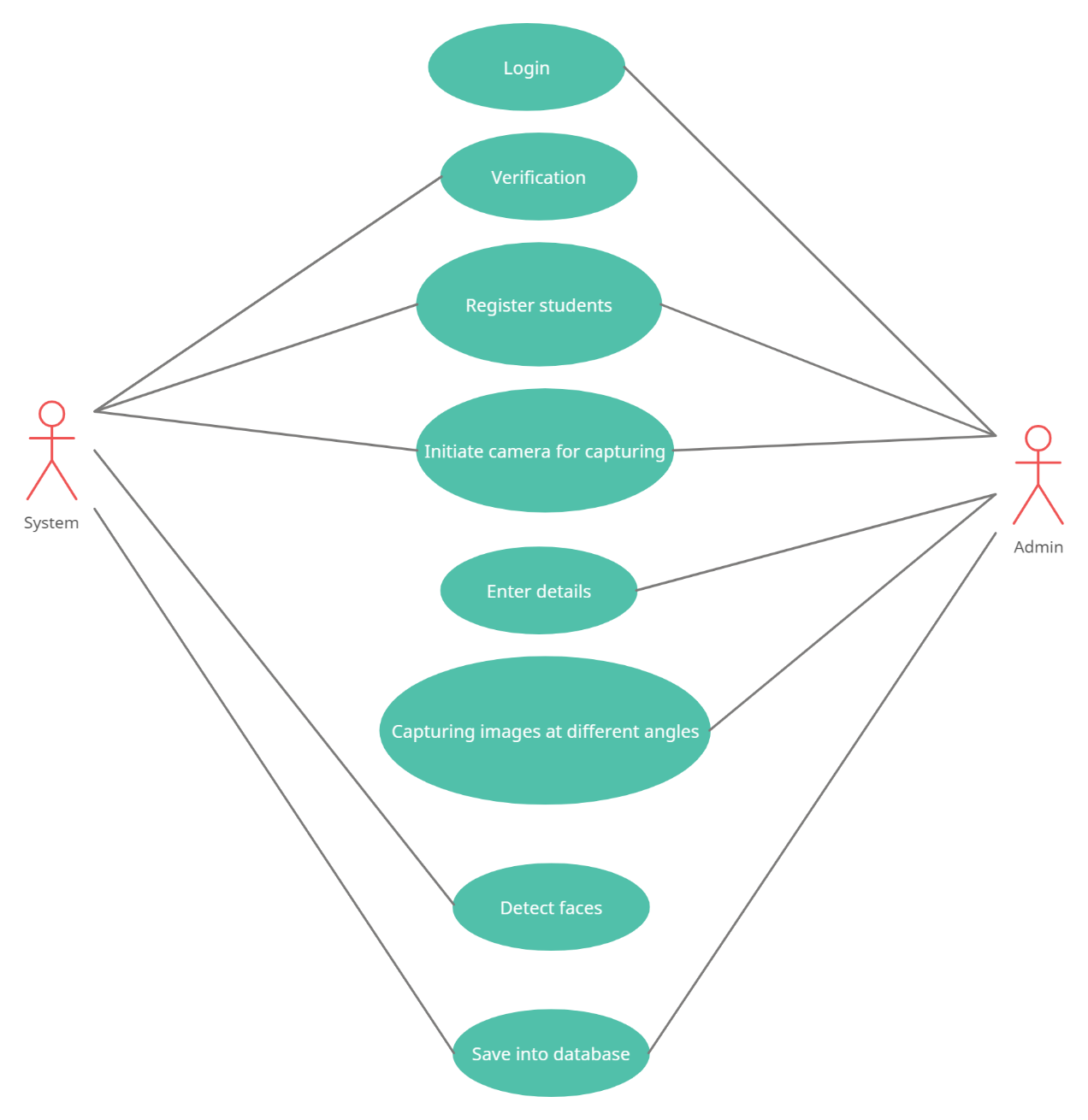
## Use-Cases diagram

We build use case diagrams to ensure the system's overall functionality. It also assists in the good interpretation of the specifications.



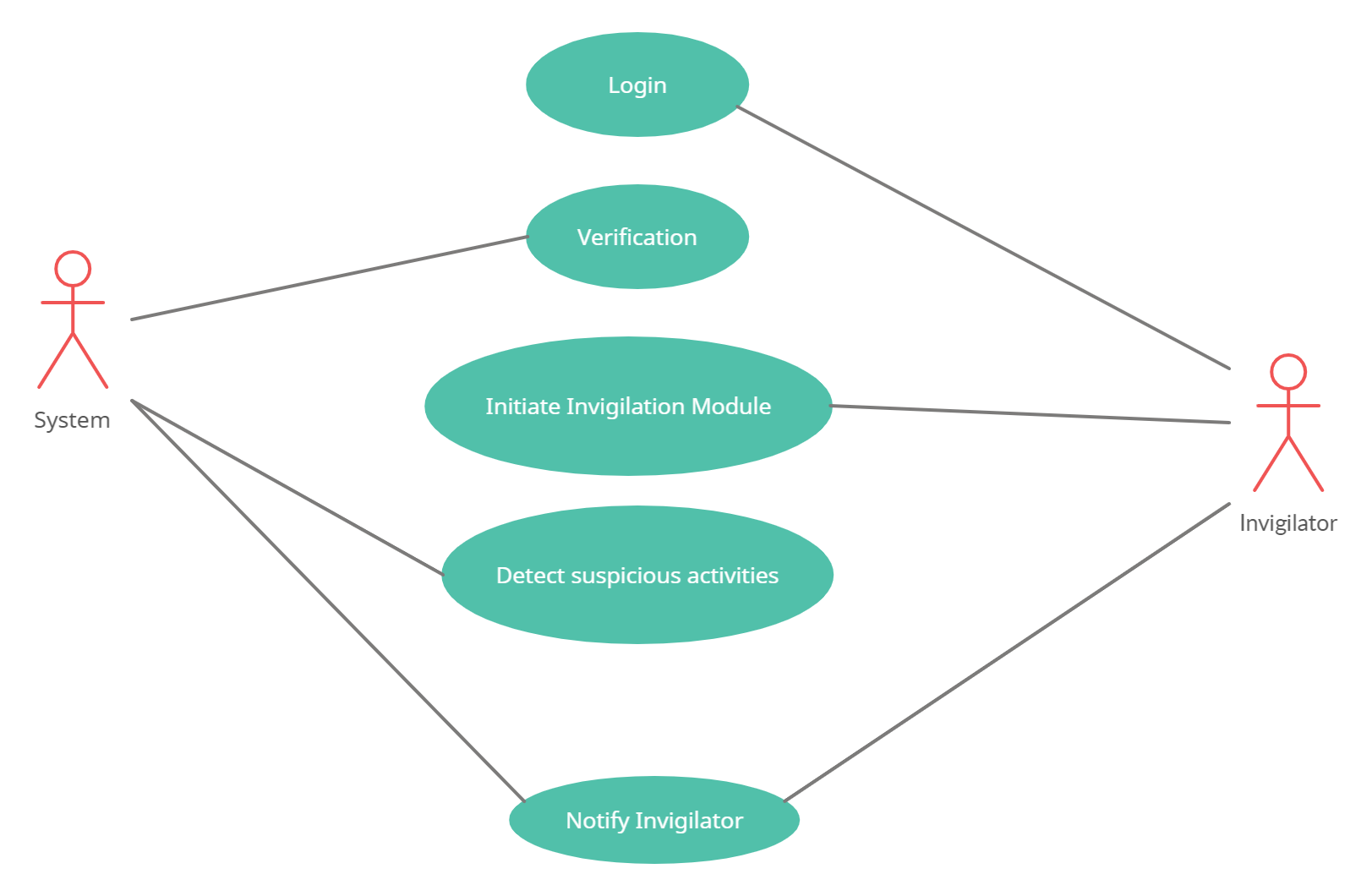
**Figure 4.1: Main Use Case Diagram of whole system**

## Use cases of Registration of students

****

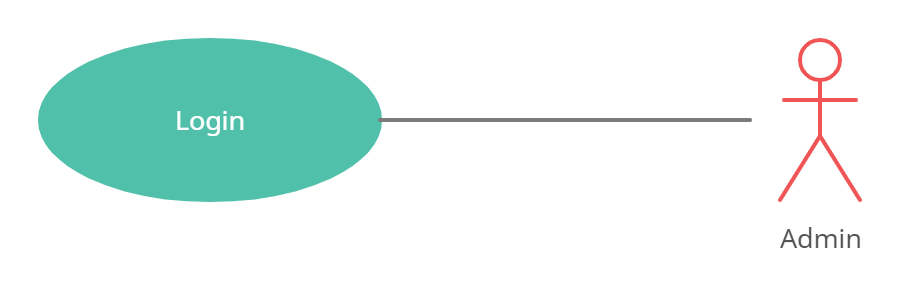
**Figure 4.2: Main Use Case Diagram of Registration of Students**

## Use cases of Invigilation of students

****

**Figure 4.3: Main Use Case Diagram of Invigilation of Students**

## Log-in Use Case:

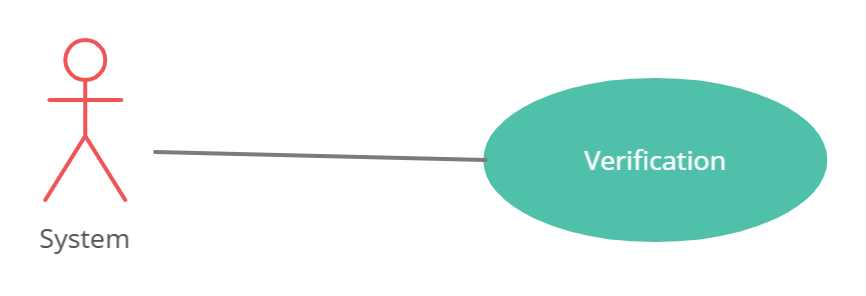


**Figure 4.4: Login Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 1 |
| Title | Log-in |
| Description | Admin will login to the system through desktop application |
| Primary Actor | admin |
| Pre Condition | Account must be created |
| Post Condition | Admin can use the system |

**Table 4.1: Login Use Case**

## Verification Use Case:

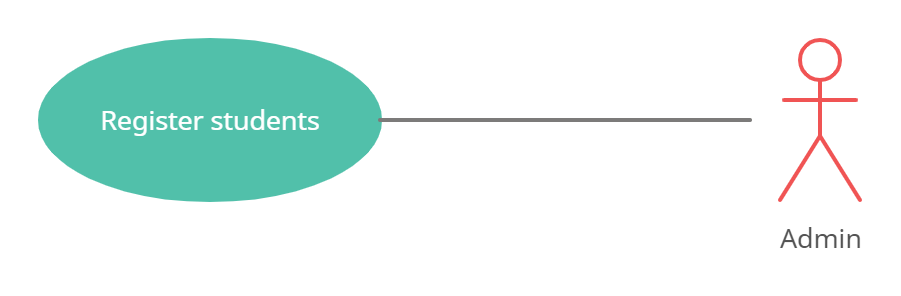


**Figure 4.5: Verification Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 2 |
| Title | Verification |
| Description | System will verify the person by checking email and password |
| Primary Actor | System |
| Pre Condition | User Credential must be entered |
| Post Condition | System allows its use. |

**Table 4.2: Verification Use Case**

## Registration Use Case:



**Figure 4.6: Registration Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 3 |
| Title | Registration |
| Description | Registration Interface will be provided when admin open the registration module |
| Primary Actor | Admin |
| Pre Condition | Admin must have logged-in |
| Post Condition | Admin can register the student. |

**Table 4.3: Registration Use Case**

## Initiate Camera Use Case:

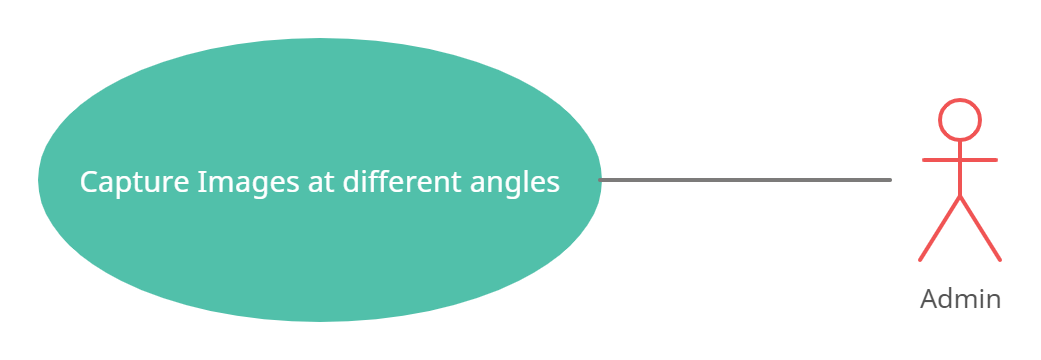


**Figure 4.7: Initiate Camera Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 4 |
| Title | Initiate Camera |
| Description | Admin request for initiating camera and system will initiate the camera |
| Primary Actor | System and Admin |
| Pre Condition | Admin should be in the Registration module of the system |
| Post Condition | Admin will be able to take pictures of the student at different angles |

**Table 4.4: Initiate Camera Use Case Diagram**

## Capture Image at different angle Use Case:

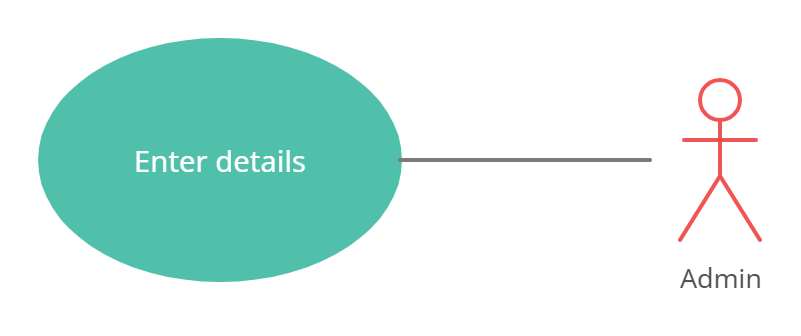
****

**Figure 4.8: Capture Image at different angle Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 5 |
| Title | Capturing Images at different angles |
| Description | Admin can take picture at different angles . |
| Primary Actor | Admin |
| Pre Condition | Admin should initiate camera first. |
| Post Condition | Admin will be able to make a dataset. |

**Table 4.5: Capturing at Different Angles Use Case**

## Entering detail Use Case:



**Figure 4.9: Entering detail Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 6 |
| Title | Enter Details |
| Description | Admin will enter detail of the student e.g. name ,enrollment etc. |
| Primary Actor | Admin |
| Pre Condition | AImages should be captured |
| Post Condition | Admin will save the data into database. |

**Table 4.6: Entering Detail Use Case**

## Saved to database Use Case:

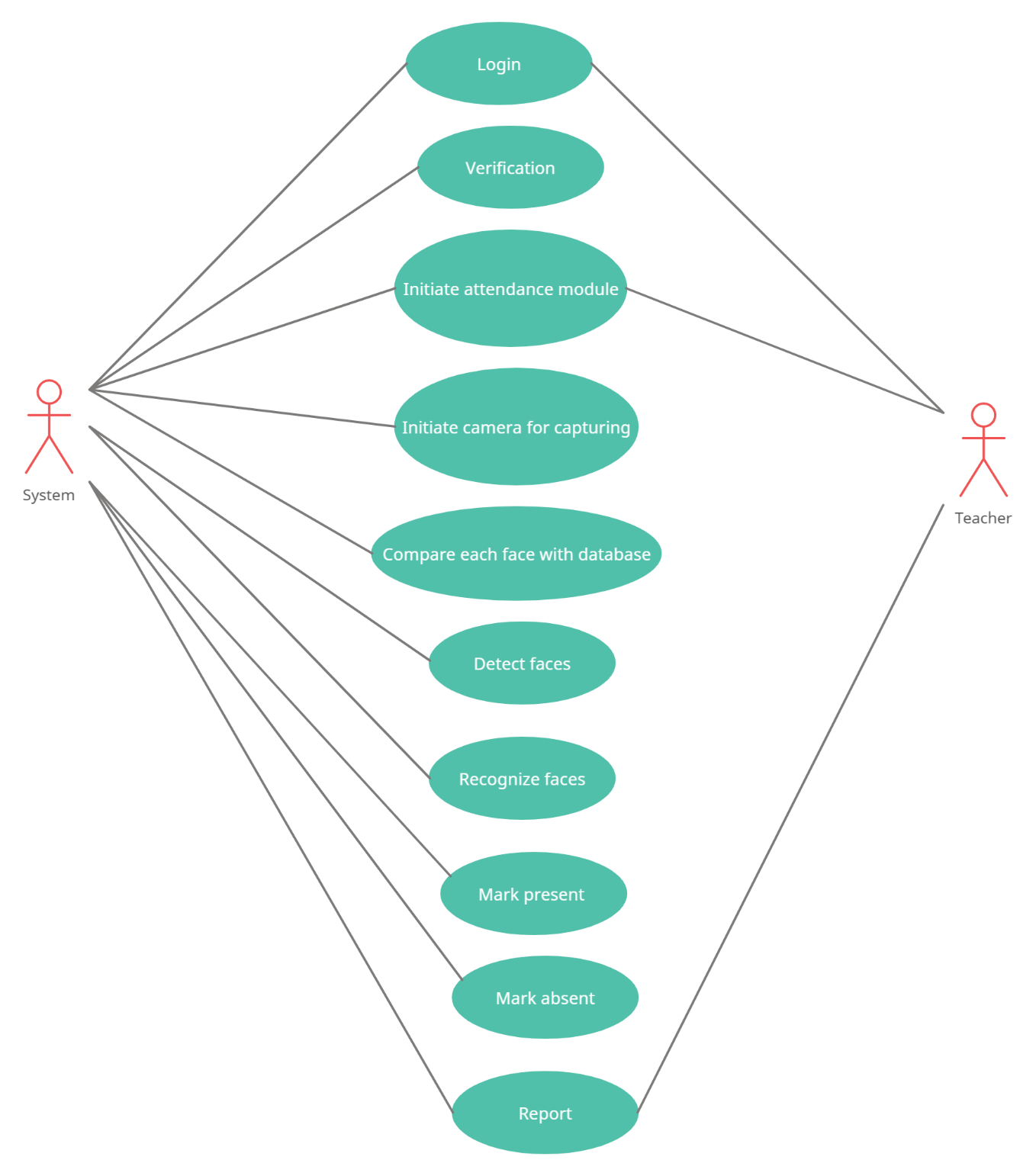
****

**Figure 4.10: Saved to database Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 7 |
| Title | Saved into Database |
| Description | Admin will request to save the data in database and system will save it |
| Primary Actor | System and Admin |
| Pre Condition | All the necessary data should be collected |
| Post Condition | Student will be registered in the institutes database. |

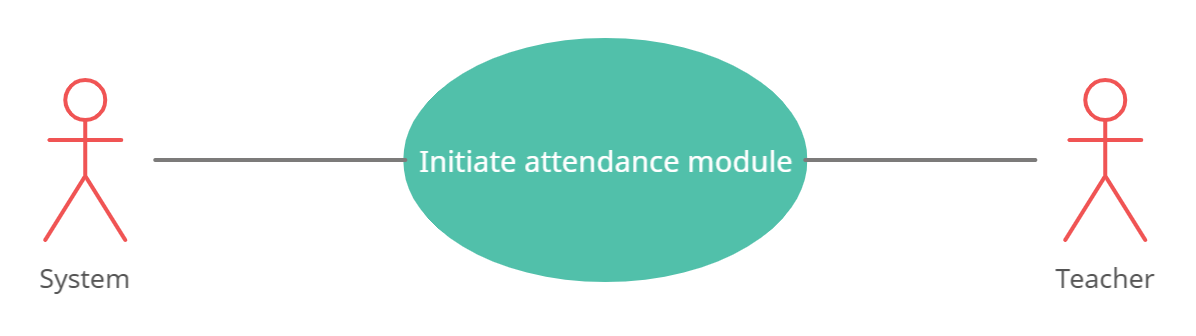
**Table 4.7: Saved to Database Use Case**

## Use cases of Attendance of students

****

**Figure 4.11: Use Case of Attendance of Students Diagram**

## Initiate Attendance Module Use Case:

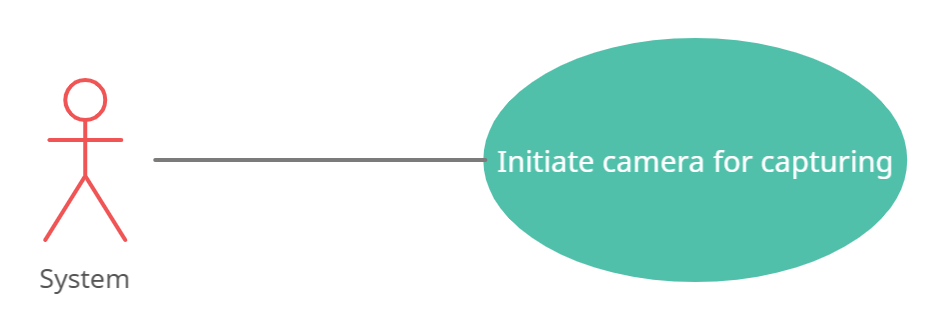


**Figure 4.12: Initiate Attendance Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 8 |
| Title | Initiate Attendance Module |
| Description | Teacher will request to initiate attendance module and system will start  the system for attendance |
| Primary Actor | System and Admin |
| Pre Condition | Teacher should be logged –in |
| Post Condition | Teacher will be able to get the notification about attendance. |

**Table 4.8: Initiate Attendance Module**

## Initiate Camera for Capturing:

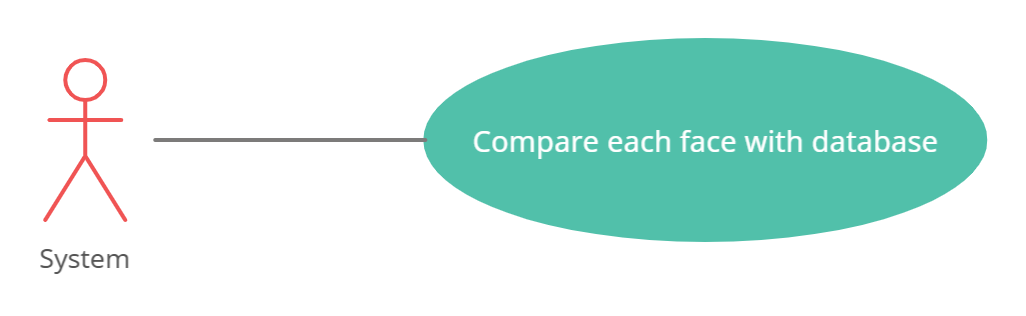


**Figure 4.13: Initiate Camera Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 9 |
| Title | Initiate camera for capturing |
| Description | System will automatically start camera and taking pictures. |
| Primary Actor | System |
| Pre Condition | Attendance module should be initiated |
| Post Condition | System will send the faces and for analyzing. |

**Table 4.9: Initiate Camera**

## Compare each face with database Use Case:

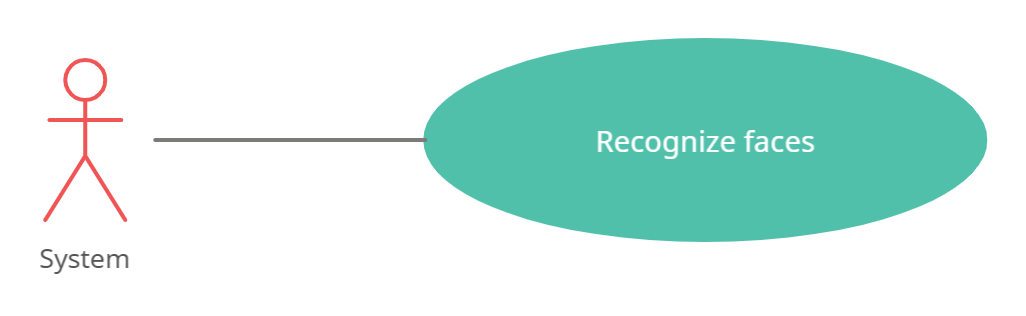


**Figure 4.14: Comparison Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 10 |
| Title | Compare each face with database |
| Description | System will compare the images sent from camera |
| Primary Actor | System |
| Pre Condition | Images should be present for comparison |
| Post Condition | System will be able to recognize the face of the student |

**Table 4.10: Face Comparison**

## Recognize Face Use Case:

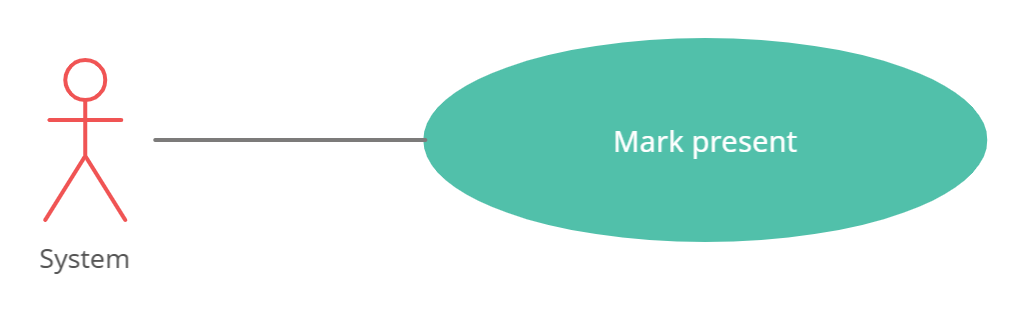


**Figure 4.15: Reorganization Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 11 |
| Title | Recognize Face |
| Description | System after comparison will recognize the face. |
| Primary Actor | System |
| Pre Condition | Detected faces should be compared |
| Post Condition | System will be able to mark present or absent |

**Table 4.11: Face Reorganization**

## Mark Present Use Case:

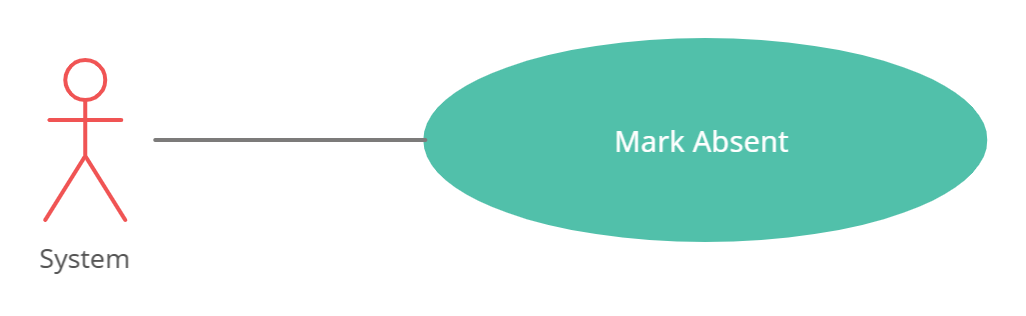


**Figure 4.16: Mark Present Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 12 |
| Title | Mark Present |
| Description | System will mark present if recognized successfully. |
| Primary Actor | System |
| Pre Condition | Face should be successfully matched |
| Post Condition | System will be able to generate notification |

**Table 4.12: Mark Present**

## Mark Absent Use Case:



**Figure 4.17: Mark Absent Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 13 |
| Title | Mark absent |
| Description | System will mark absent if face not detected. |
| Primary Actor | System |
| Pre Condition | Face not detected |
| Post Condition |  |

**Table 4.13: Mark Absent**

## Report Use Case:

****

**Figure 4.18: Report Use Case Diagram**

|  |  |
| --- | --- |
| Use Case ID | UC 14 |
| Title | Report |
| Description | Teacher will request for the report and system will generate report |
| Primary Actor | System and Admin |
| Pre Condition | Attendance should be marked |
| Post Condition | Exit |

## Table 4.14: Report

## Process Flow

## Figure 4.19 process flow

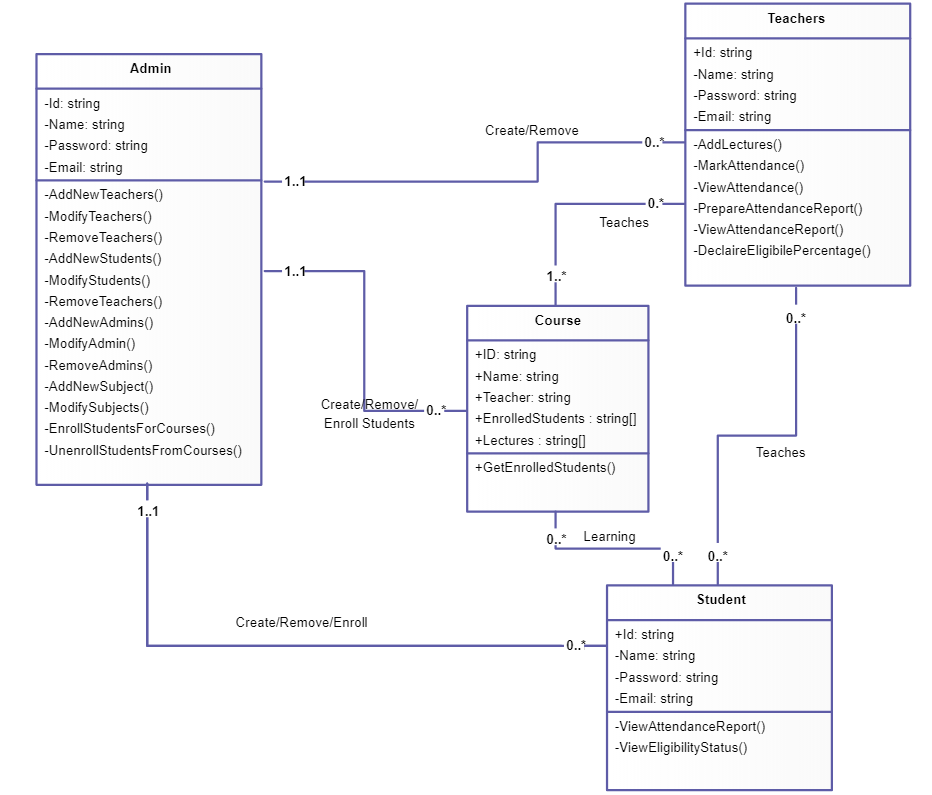
## Actors

The actor for these systems are:

* + - Administrator
    - End Users

Where the system administrator can make adjustments if necessary. He or she is in charge of ensuring that the student correctly enters his or her details. He or she is the only one who can alter the data collection. To register students, the administrator logs in. Teachers who keep track of attendance are end users.

## Class Diagram

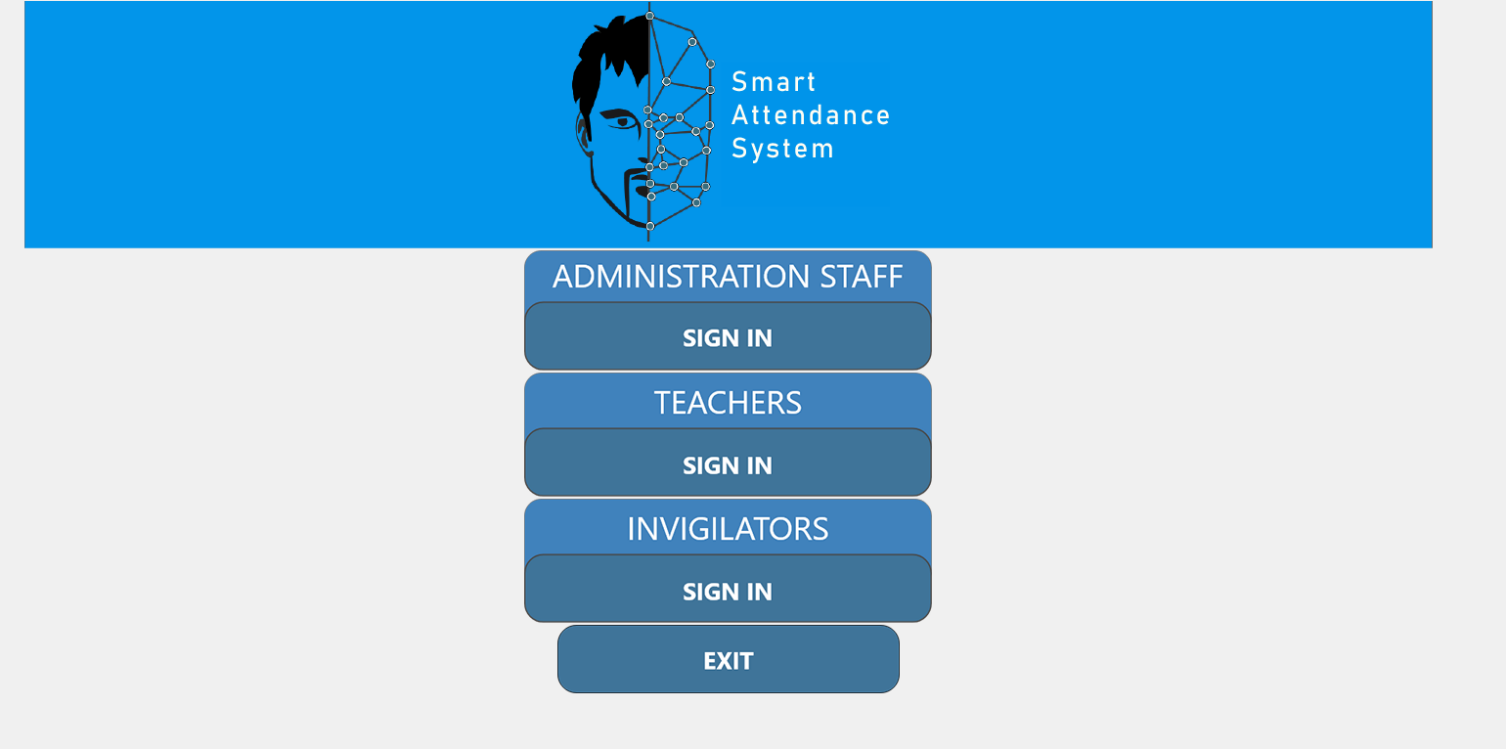


**Figure 4. 20 Class Diagram**

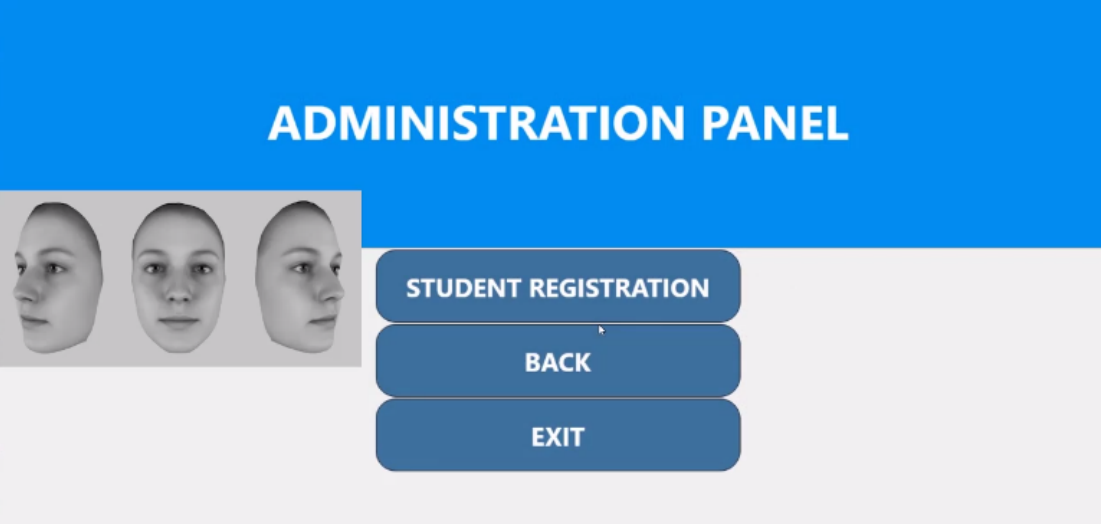
## Graphical User Interface (GUI)

## System Interfaces

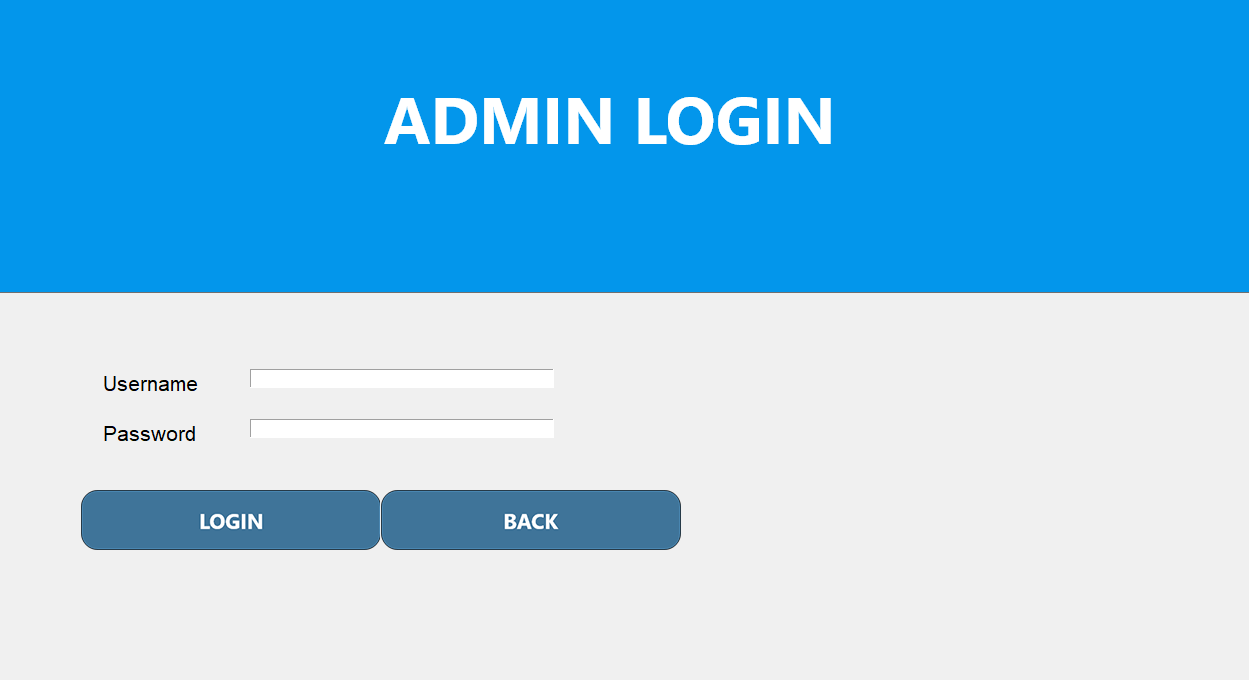
The home page of our desktop application is shown in figure 4.20. Our application will first register the students and prepare a data set of individual students. For attendance and invigilation, it is necessary to login and initiate the respective module. Main Screen of Registration, Invigilation and Attendance is shown in Fig 4.21. Fig 4.22 and Fig 4.23 respectively.



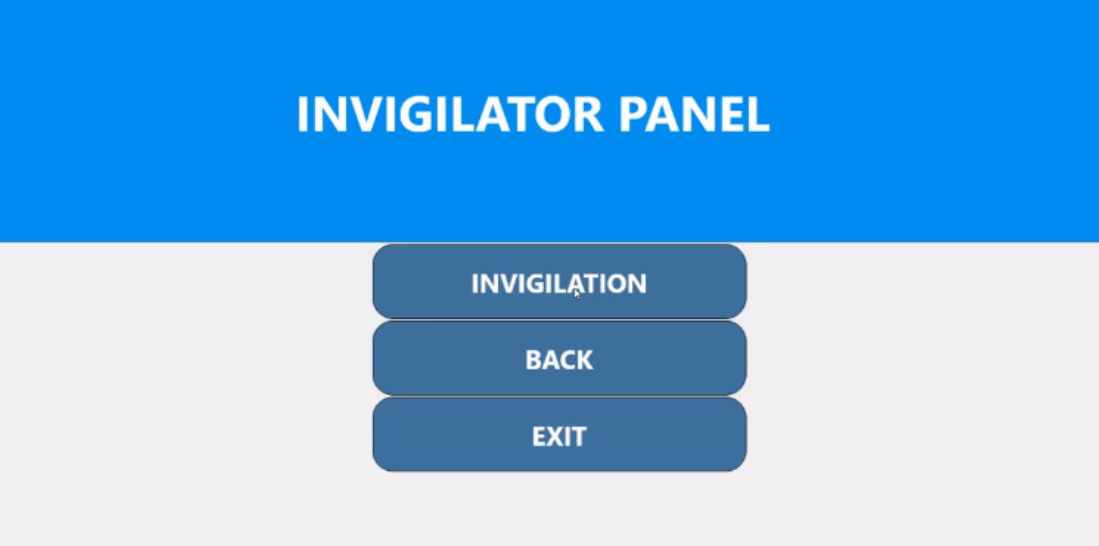
**Figure 4. 21 Main menu**



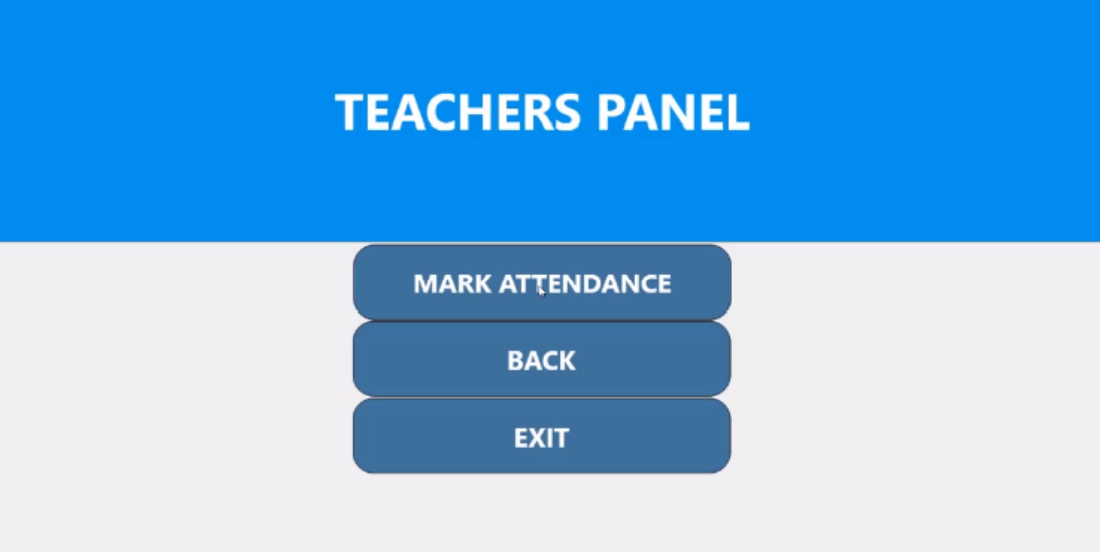
**Figure 4. 22 Admin panel**



**Figure 4. 23 Admin login**



**Figure 4. 24 Invigilator panel**



**Figure 4. 25 Teacher panel**

## SYSTEM REQUIREMENTS and SOFTWARE INTERFACES

### Hardware Requirements

* + - * Core 2 duo or higher
      * GPU (minimum 2 GB)
      * High Definition Camera

### Software Requirements

* + - * Python 3.5 or latest version
      * Wampserver/MySqlserver
      * Spyder
      * Windows 8 or higher

## Summary

This chapter provides a comprehensive overview of the architectural design, use cases, functional and non-functional specifications, and so forth. In this chapter, all of the requirements are described in great detail.

**Chapter 5**

**System Design**

# Chapter 5: System Design

With the aid of some diagrams such as an architecture diagram, a sequence diagram, and an operation diagram, we will explain the architecture of the system modules and the entire system in this chapter.

## Design Goals

**Performance**

This system's primary concern is efficiency. The rapid rise in the number of users is undeniable. As a result, the machine must be able to work best under all circumstances. The quality of a product is heavily influenced by its results. It is assumed that a product is successful if it functions properly at its peak time. As a result, the project's success has been a major concern from the start. As a developer, I think that with improved performance, administrators will be able to upload files and launch multiple campaigns without the device crashing. Memory expansion should have no effects on device performance.

**Security**

Another big concern is security. Since the product will be used on the internet, protection will be a top priority. The software can be used by people with various positions. In this University example, the user groups that will use the system are the administrator and student, although this will vary depending on the type of entity, such as a corporation. - user group has its own set of permissions. Any security or privacy problems that affect the safety of the data generated by the software system should be resolved with these access rights.

**Usability**

Usability is the most critical feature of any programme. The device must be simple to comprehend and use. Its user interface should be easy to comprehend. Furthermore, the user must not get stuck at any point. The machine must instruct itself on how to operate it. Help should be accessible at each point, and the system should be designed such that once a user has completed it, he or she remembers how to use it even after a long period of time. So, in order to keep it simple and straightforward, I've done my best to make this system simple and useful for a wide range of users.

**Availability**

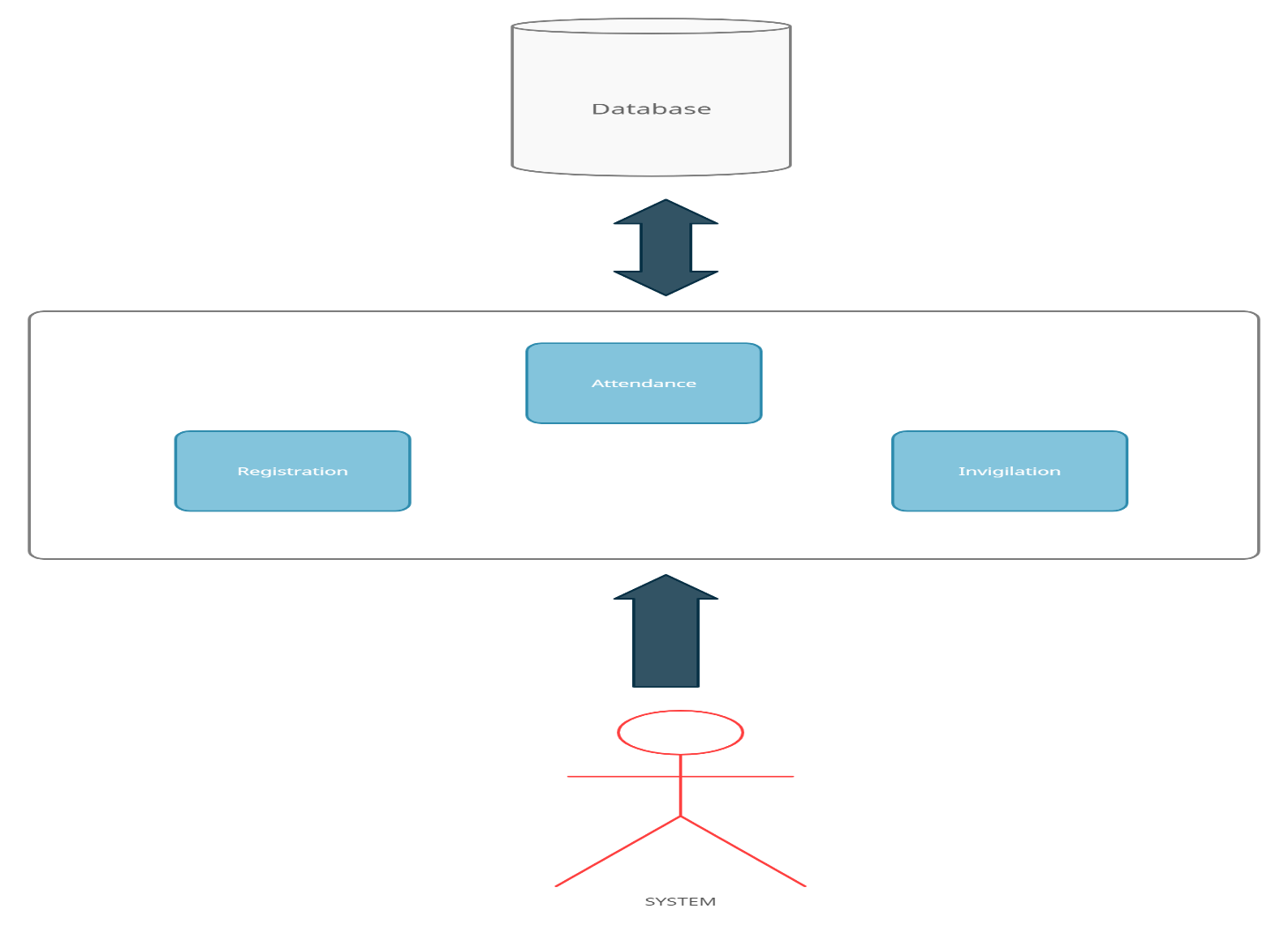
At all costs, the device should be open to users at all times. Since this is an application-based framework. As a result, it must be accessible on both ends. Logins, signups, and other operations should take milliseconds.

**Maintainable**

If the client wishes to add more features to the software system after it has been deployed, the software must be versatile enough to accommodate any changes.

## System Architecture Diagram

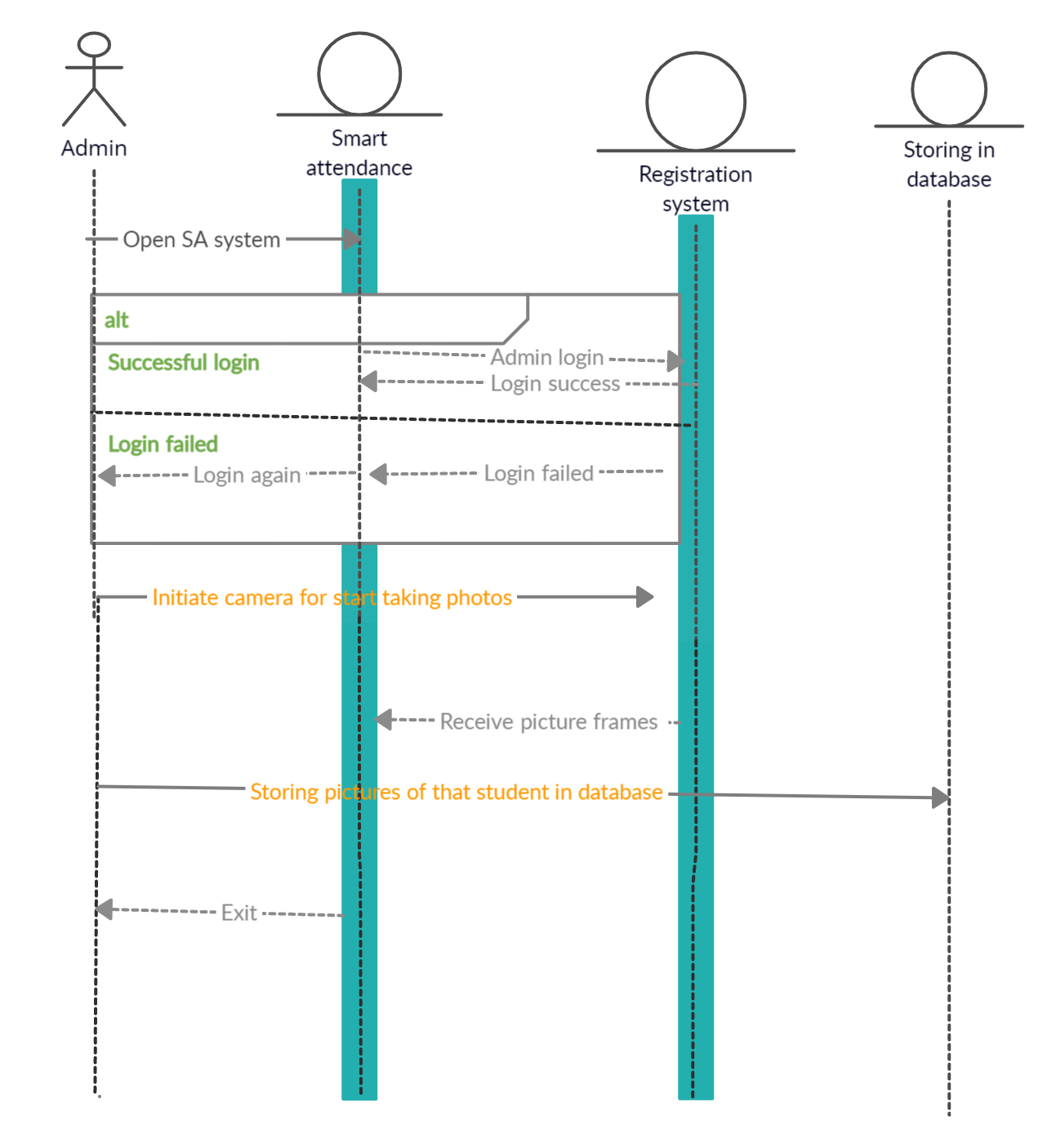
The system grants Administrator access to the registration area, where he or she can collect student data, prepare data sets, and save them to the database. After logging in the instructor and allowing him/her access to the attendance system, the camera starts taking pictures and compares them to the pictures in the database to mark student attendance. And inigilation detects suspisious activity in examination hall.

****

**Figure 5. 1 System Architecture Diagram**

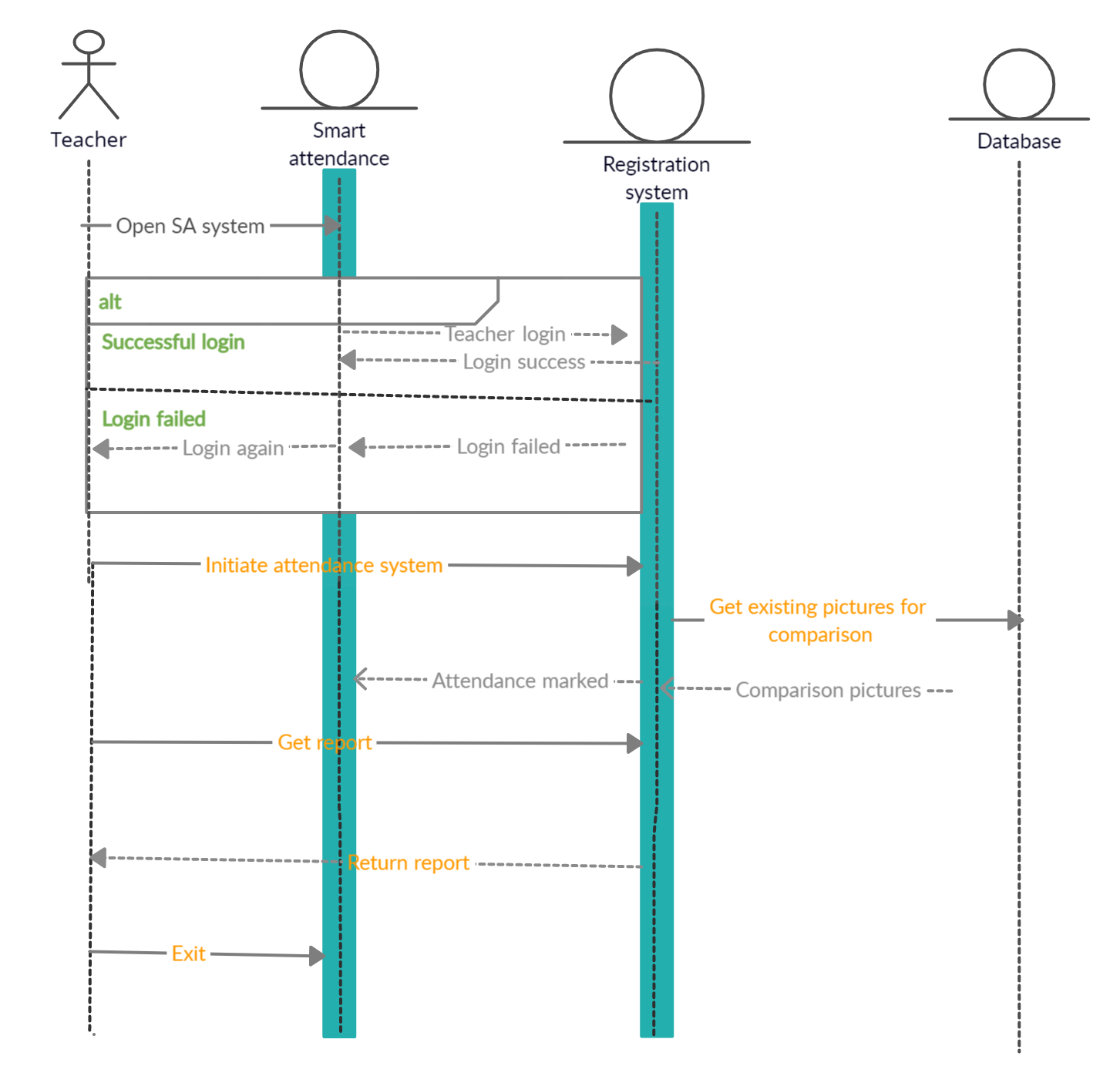
## Sequence diagram

### Registration of Students:



**Figure 5.2: Registration of Students Sequence Diagram**

### Attendance of Students:

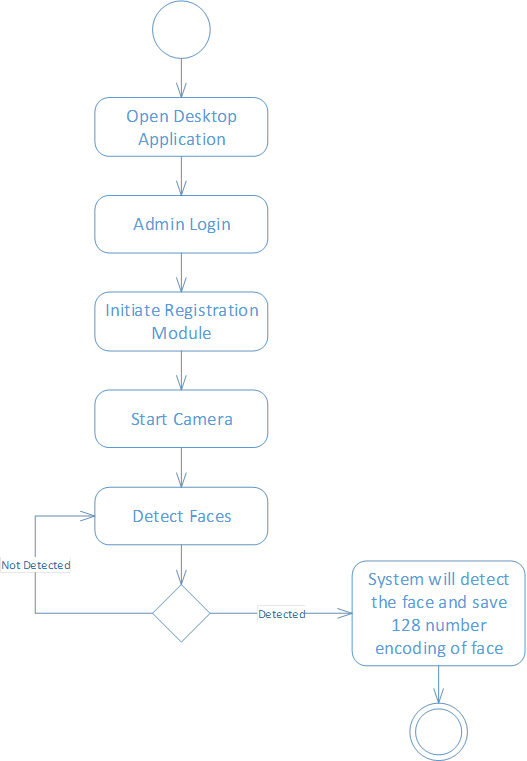
****

**Figure 5.3: Attendance of Student Sequence Diagram**

## Activity Diagram

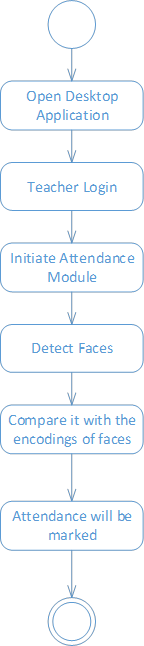
Action diagrams are perhaps the best way to illustrate dynamic facets of the system. This uses a flow chart system to control the flow of one operation to the next..

### Admin Activity Diagram



**Figure 5.4: Admin Activity Diagram**

### Teacher Activity Diagram



**Figure 5.5: Teacher Activity Diagram**

## Invigilator Activity Diagram

## 

**Figure 5.6: Invigilator Activity Diagram**

## System Flow Diagram

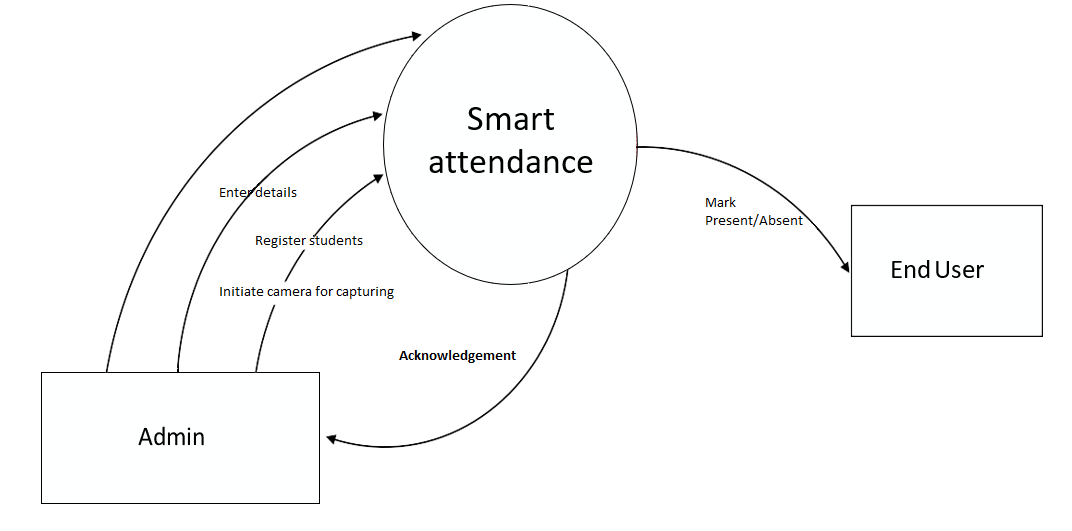
## 

## Figure 5.7 Smart Attendance flow diagram

## Data Model

**Data Flow Diagram of Smart attendance**

### Context Level



### Level 1 DFD

**Figure 5. 8 Context Level Diagram**

**Mark Attendance**



**Image matched**

**Compare Req.**



**Delete User Request**

**Send view request**

**Delete**

**User**

**View Users**

**Delete User**

**Comparing**

**Initiate attendance**

**fetch image from database**

**Smart attendance**

**opens SA**

**Admin**

**Delete Req.**

**Show records**

**Register user**

**Insert Req.**



**Stored record**

**Initiate camera**

**Database**

**Figure 5. 9 Level 1 Data Flow Diagram**

## Data Dictionary

## Table Name (e.g admin)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Is Primary Key** | **Is Null** | **Comment** |
| AdminID | varchar | Yes | No | Admin id for login |

## Teacher

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Is Primary Key** | **Is Null** | **Comment** |
| Lecturer\_id | varchar | Yes | No | The lecture id stores when registered |

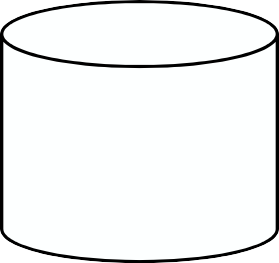
## Invigilator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Is Primary Key** | **Is Null** | **Comment** |
| Invigilator\_id | varchar | Yes | No | The invigilator id stores when registered |

**Table 5. 1 Data Dictionary**

## Deployment Diagram

**Database Server**



Fetch images data from database

**Application Server**

Compare images

report

**Student record matched**

**Mark attendance**

**Figure 5. 10 Deployment Diagram**

**Summary**

Design is the most difficult aspect of development since it determines whether a programme is accepted or rejected. All non-functional specifications must be met at this stage. This chapter uses sequence diagrams, behavioural models, and system deployment diagrams to explain the flow of phases.

**Chapter 6**

**Coding**

* 1. **Main Screen:**

import tkinter

from tkinter import \*

from tkinter import messagebox, filedialog, ttk

import os

def admin\_login():

root.destroy()

os.system("python admin\_login.py")

def teacher\_login():

root.destroy()

os.system("python teacher\_login.py")

def invigilator\_login():

root.destroy()

os.system("python invigilator\_login.py")

def Main\_Screen():

global root

root=tkinter.Tk()

root.state('zoomed')

#root.configure(bg='blue')

root.geometry("1920x1080")

photo=PhotoImage(file="ui/frt.png")

photo2=PhotoImage(file="ui/admin\_sign\_in.png")

photo3=PhotoImage(file="ui/teacher\_sign\_in.png")

photo4=PhotoImage(file="ui/Invigilator\_sign\_in.png")

photo5=PhotoImage(file="ui/exit1.png")

btn= Button(

root,

image=photo,

command="",

border=0,

text="",

compound=TOP,

)

btn.pack()

btn2= Button(

root,

image=photo2,

command=admin\_login,

border=0,

text="",

compound=TOP,

)

btn2.pack()

btn3= Button(

root,

image=photo3,

command=teacher\_login,

border=0,

text="",

compound=TOP,

)

btn3.pack()

btn4= Button(

root,

image=photo4,

command=invigilator\_login,

border=0,

text="",

compound=TOP,

)

btn4.pack()

btn5= Button(

root,

image=photo5,

command=root.destroy,

border=0,

text="",

compound=TOP,

)

btn5.pack()

root.mainloop()

Main\_Screen()

* 1. **ADMIN SIDE**

**Admin Panel:**

import tkinter

import MySQLdb

from tkinter import \*

from tkinter import messagebox, filedialog, ttk

import os

import cv2

import dlib

import numpy as np

from imutils import face\_utils

from decimal import Decimal

import math

import sys

import time

from playsound import playsound

import winsound

def registration():

os.system("python registration.py")

def mark\_attendance():

os.system("python attendance.py")

def invigilation():

os.system("python headpose.py")

def btnclick():

messagebox.showinfo("Message","button is clicked")

def main\_screen():

root.destroy()

os.system("python main\_screen.py")

def Main\_Menu():

global root

root=tkinter.Tk()

#root.configure(bg='blue')

root.geometry("1920x1080")

photo=PhotoImage(file="ui/admin\_panel.png")

photo2=PhotoImage(file="ui/registration.png")

photo3=PhotoImage(file="ui/mark\_attendance.png")

photo4=PhotoImage(file="ui/invigilation.png")

photo5=PhotoImage(file="ui/exit.png")

back=PhotoImage(file="ui/back.png")

btn= Button(

root,

image=photo,

command=btnclick,

border=0,

text="",

compound=TOP,

)

btn.pack()

btn2= Button(

root,

image=photo2,

command=registration,

border=0,

text="",

compound=TOP,

)

btn2.pack()

btn3= Button(

root,

image=back,

command=main\_screen,

border=0,

text="",

compound=TOP,

)

btn3.pack()

#

# btn4= Button(

# root,

# image=photo4,

# command=invigilation,

# border=0,

# text="",

# compound=TOP,

# )

# btn4.pack()

btn5= Button(

root,

image=photo5,

command=root.destroy,

border=0,

text="",

compound=TOP,

)

btn5.pack()

root.mainloop()

Main\_Menu()

**Admin Login:**

import tkinter

from tkinter import \*

from tkinter import messagebox, filedialog, ttk

from PIL import Image,ImageTk

import os

import mysql.connector as mysql

def main\_screen():

root.destroy()

os.system("python main\_screen.py")

def main():

global root

root = Tk()

root.title("Admin Login")

root.geometry("1920x1080")

root.state('zoomed')

global uid

global pid

photo=PhotoImage(file="ui/admin\_login.png")

photo2=PhotoImage(file="ui/login1.png")

photo3=PhotoImage(file="ui/back1.png")

btn= Button(

root,

image=photo,

command="",

border=0,

text="",

compound=TOP,

)

btn.pack()

username = Label(root, text='Username',font=('bold',16))

username.place(x=450,y=430);

password = Label(root, text='Password',font=('bold',16))

password.place(x=450,y=480);

uid=Entry(root,width="50");

uid.place(x=600,y=430)

pid=Entry(root,width="50",show='\*');

pid.place(x=600,y=480)

btn2= Button(

root,

image=photo2,

command=insert,

border=0,

text="",

compound=TOP

)

btn2.place(x=430,y=550)

btn3= Button(

root,

image=photo3,

command=main\_screen,

border=0,

text="",

compound=TOP

)

btn3.place(x=730,y=550)

root.mainloop()

def insert():

user=uid.get()

pas=pid.get()

if (user=="" or pas==""):

messagebox.showinfo("Error","Enter Username and Password")

else:

con=mysql.connect(host="localhost",user="root",password="smartattendance123",database="sas")

cursor=con.cursor()

flag=0

cursor.execute("SELECT \* FROM admin where admin\_username='" + user + "'")

for row in cursor.fetchall():

if row[1]==pas:

flag=1

break

if flag==1:

messagebox.showinfo("Login","Login Successfull")

root.destroy()

os.system("python admin\_panel.py")

else:

messagebox.showinfo("Login Failed","Incorrect Username or Password")

uid.delete(0,END)

pid.delete(0,END)

main()

* 1. **TEACHER SIDE**

**Teacher Login:**

import tkinter

from tkinter import \*

from tkinter import messagebox, filedialog, ttk

from PIL import Image,ImageTk

import os

import mysql.connector as mysql

def main\_screen():

root.destroy()

os.system("python main\_screen.py")

def main():

global root

root = Tk()

root.title("Teacher Login")

root.geometry("1920x1080")

root.state('zoomed')

#Image\_open=Image.open("ui/blue.png")

#image=ImageTk.PhotoImage(Image\_open)

#gg='#134e86'

#logo=Label(root,image=image,bg=gg)

#logo.place(x=0,y=0,bordermode="outside")

global uid

global pid

photo=PhotoImage(file="ui/teacher\_login.png")

photo2=PhotoImage(file="ui/login1.png")

photo3=PhotoImage(file="ui/back1.png")

btn= Button(

root,

image=photo,

command="",

border=0,

text="",

compound=TOP,

)

btn.pack()

username = Label(root, text='Username',font=('bold',16))

username.place(x=450,y=430);

password = Label(root, text='Password',font=('bold',16))

password.place(x=450,y=480);

uid=Entry(root,width="50");

uid.place(x=600,y=430)

pid=Entry(root,width="50",show='\*');

pid.place(x=600,y=480)

btn2= Button(

root,

image=photo2,

command=insert,

border=0,

text="",

compound=TOP

)

btn2.place(x=430,y=550)

btn3= Button(

root,

image=photo3,

command=main\_screen,

border=0,

text="",

compound=TOP

)

btn3.place(x=730,y=550)

root.mainloop()

def insert():

user=uid.get()

pas=pid.get()

if (user=="" or pas==""):

messagebox.showinfo("Error","Enter Username and Password")

else:

con=mysql.connect(host="localhost",user="root",password=" Adnan123@",database="sas")

cursor=con.cursor()

flag=0

cursor.execute("SELECT \* FROM teachers where teacher\_username='" + user + "'")

for row in cursor.fetchall():

if row[1]==pas:

flag=1

break

if flag==1:

messagebox.showinfo("Login","Login Successfull")

root.destroy()

os.system("python teacher\_panel.py")

else:

messagebox.showinfo("Login Failed","Incorrect Username or Password")

uid.delete(0,END)

pid.delete(0,END)

main()

**Teacher Panel:**

# -\*- coding: utf-8 -\*-

"""

Created on Fri Feb 21 20:36:35 2021

@author: HP Pavilion 15

"""

import tkinter

import MySQLdb

from tkinter import \*

from tkinter import messagebox, filedialog, ttk

import os

import cv2

import dlib

import numpy as np

from imutils import face\_utils

from decimal import Decimal

import math

import sys

import time

from playsound import playsound

import winsound

def registration():

os.system("python registration.py")

def mark\_attendance():

os.system("python attendance.py")

def invigilation():

os.system("python headpose.py")

def btnclick():

messagebox.showinfo("Message","button is clicked")

def main\_screen():

root.destroy()

os.system("python main\_screen.py")

def Main\_Menu():

global root

root=tkinter.Tk()

#root.configure(bg='blue')

root.geometry("1920x1080")

photo=PhotoImage(file="ui/teacher\_panel.png")

photo2=PhotoImage(file="ui/registration.png")

photo3=PhotoImage(file="ui/mark\_attendance.png")

photo4=PhotoImage(file="ui/invigilation.png")

back=PhotoImage(file="ui/back.png")

photo5=PhotoImage(file="ui/exit.png")

btn= Button(

root,

image=photo,

command=btnclick,

border=0,

text="",

compound=TOP,

)

btn.pack()

# btn2= Button(

# root,

# image=photo2,

# command=registration,

# border=0,

# text="",

# compound=TOP,

# )

# btn2.pack()

btn3= Button(

root,

image=photo3,

command=mark\_attendance,

border=0,

text="",

compound=TOP,

)

btn3.pack()

btn4= Button(

root,

image=back,

command=main\_screen,

border=0,

text="",

compound=TOP,

)

btn4.pack()

btn5= Button(

root,

image=photo5,

command=root.destroy,

border=0,

text="",

compound=TOP,

)

btn5.pack()

root.mainloop()

Main\_Menu()

#def final():

# execfile('final.py')

* 1. **Invigilator side**

**Invigilator Login:**

import tkinter

from tkinter import \*

from tkinter import messagebox, filedialog, ttk

from PIL import Image,ImageTk

import os

import mysql.connector as mysql

def main\_screen():

root.destroy()

os.system("python main\_screen.py")

def main():

global root

root = Tk()

root.title("Invigilators Login")

root.geometry("1920x1080")

root.state('zoomed')

#Image\_open=Image.open("ui/blue.png")

#image=ImageTk.PhotoImage(Image\_open)

#gg='#134e86'

#logo=Label(root,image=image,bg=gg)

#logo.place(x=0,y=0,bordermode="outside")

global uid

global pid

photo=PhotoImage(file="ui/invigilator\_login.png")

photo2=PhotoImage(file="ui/login1.png")

photo3=PhotoImage(file="ui/back1.png")

btn= Button(

root,

image=photo,

command="",

border=0,

text="",

compound=TOP,

)

btn.pack()

username = Label(root, text='Username',font=('bold',16))

username.place(x=450,y=430);

password = Label(root, text='Password',font=('bold',16))

password.place(x=450,y=480);

uid=Entry(root,width="50");

uid.place(x=600,y=430)

pid=Entry(root,width="50",show='\*');

pid.place(x=600,y=480)

btn2= Button(

root,

image=photo2,

command=insert,

border=0,

text="",

compound=TOP

)

btn2.place(x=430,y=550)

btn3= Button(

root,

image=photo3,

command=main\_screen,

border=0,

text="",

compound=TOP

)

btn3.place(x=730,y=550)

root.mainloop()

def insert():

user=uid.get()

pas=pid.get()

if (user=="" or pas==""):

messagebox.showinfo("Error","Enter Username and Password")

else:

con=mysql.connect(host="localhost",user="root",password="adnan123@",database="sas")

cursor=con.cursor()

flag=0

cursor.execute("SELECT \* FROM invigilators where invigilator\_username='" + user + "'")

for row in cursor.fetchall():

if row[1]==pas:

flag=1

break

if flag==1:

messagebox.showinfo("Login","Login Successfull")

root.destroy()

os.system("python invigilator\_panel.py")

else:

messagebox.showinfo("Login Failed","Incorrect Username or Password")

uid.delete(0,END)

pid.delete(0,END)

main()

**Invigilator Panel:**

import tkinter

import MySQLdb

from tkinter import \*

from tkinter import messagebox, filedialog, ttk

import os

import cv2

import dlib

import numpy as np

from imutils import face\_utils

from decimal import Decimal

import math

import sys

import time

from playsound import playsound

import winsound

def registration():

os.system("python registration.py")

def mark\_attendance():

os.system("python attendance.py")

def invigilation():

os.system("python headpose.py")

def invigilation2():

os.system("python eye\_gaze.py")

def btnclick():

messagebox.showinfo("Message","button is clicked")

def main\_screen():

root.destroy()

os.system("python main\_screen.py")

def Main\_Menu():

global root

root=tkinter.Tk()

#root.configure(bg='blue')

root.geometry("1920x1080")

photo=PhotoImage(file="ui/invigilator\_panel.png")

photo2=PhotoImage(file="ui/registration.png")

photo3=PhotoImage(file="ui/mark\_attendance.png")

photo4=PhotoImage(file="ui/invigilation1.png")

back=PhotoImage(file="ui/back.png")

photo5=PhotoImage(file="ui/exit.png")

photo10=PhotoImage(file="ui/invigilation2.png")

btn= Button(

root,

image=photo,

command=btnclick,

border=0,

text="",

compound=TOP,

)

btn.pack()

# btn2= Button(

# root,

# image=photo2,

# command=registration,

# border=0,

# text="",

# compound=TOP,

# )

# btn2.pack()

btn3= Button(

root,

image=photo4,

command=invigilation,

border=0,

text="",

compound=TOP,

)

btn3.pack()

btn10= Button(

root,

image=photo10,

command=invigilation2,

border=0,

text="",

compound=TOP,

)

btn10.pack()

btn4= Button(

root,

image=back,

command=main\_screen,

border=0,

text="",

compound=TOP,

)

btn4.pack()

btn5= Button(

root,

image=photo5,

command=root.destroy,

border=0,

text="",

compound=TOP,

)

btn5.pack()

root.mainloop()

Main\_Menu()

#def final():

# execfile('final.py')

* 1. **Registration**

import cv2

from align\_custom import AlignCustom

from face\_feature import FaceFeature

from mtcnn\_detect import MTCNNDetect

from tf\_graph import FaceRecGraph

import argparse

import sys

import json

import time

import numpy as np

TIMEOUT = 10 #10 seconds

def main(args):

create\_manual\_data();

'''

Description:

User input his/her name or ID -> Images from Video Capture -> detect the face -> crop the face and align it

-> face is then categorized in 3 types: Center, Left, Right

-> Extract 128D vectors( face features)

-> Append each newly extracted face 128D vector to its corresponding position type (Center, Left, Right)

-> Press Q to stop capturing

-> Find the center ( the mean) of those 128D vectors in each category. ( np.mean(...) )

-> Save

'''

def create\_manual\_data():

vs = cv2.VideoCapture(0); #get input from webcam

print("Please input new user ID:")

new\_name = input(); #ez python input()

f = open('./facerec\_128D.txt','r');

data\_set = json.loads(f.read());

person\_imgs = {"Left" : [], "Right": [], "Center": []};

person\_features = {"Left" : [], "Right": [], "Center": []};

print("Please start turning slowly. Press 'q' to save and add this new user to the dataset");

while True:

\_, frame = vs.read();

rects, landmarks = face\_detect.detect\_face(frame, 80); # min face size is set to 80x80

for (i, rect) in enumerate(rects):

aligned\_frame, pos = aligner.align(160,frame,landmarks[:,i]);

if len(aligned\_frame) == 160 and len(aligned\_frame[0]) == 160:

person\_imgs[pos].append(aligned\_frame)

cv2.imshow("Captured face", aligned\_frame)

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

break

vs.release()

cv2.destroyAllWindows()

for pos in person\_imgs: #there r some exceptions here, but I'll just leave it as this to keep it simple

person\_features[pos] = [np.mean(extract\_feature.get\_features(person\_imgs[pos]),axis=0).tolist()]

data\_set[new\_name] = person\_features;

f = open('./facerec\_128D.txt', 'w');

f.write(json.dumps(data\_set))

if \_\_name\_\_ == '\_\_main\_\_':

parser = argparse.ArgumentParser()

parser.add\_argument("--mode", type=str, help="Run camera recognition", default="camera")

args = parser.parse\_args(sys.argv[1:]);

FRGraph = FaceRecGraph();

MTCNNGraph = FaceRecGraph();

aligner = AlignCustom();

extract\_feature = FaceFeature(FRGraph)

face\_detect = MTCNNDetect(MTCNNGraph, scale\_factor=2); #scale\_factor, rescales image for faster detection

main(args);

* 1. **Features**

**align\_custom:**

'''

Implement Dlib Face alignment strategy

However, this method/approach doesn't deform the original image like Dlib does.

This also categorizes the face in 3 types: Center, Left, Right

Align face based on facial landmarks

'''

import math

import cv2

import numpy as np

class AlignCustom(object):

def \_\_init\_\_(self):

pass

def getPos(self, points):

if abs(points[0] - points[2]) / abs(points[1] - points[2]) > 2:

return "Right";

elif abs(points[1] - points[2]) / abs(points[0] - points[2]) > 2:

return "Left";

return "Center"

def list2colmatrix(self, pts\_list):

"""

convert list to column matrix

Parameters:

----------

pts\_list:

input list

Retures:

-------

colMat:

"""

assert len(pts\_list) > 0

colMat = []

for i in range(len(pts\_list)):

colMat.append(pts\_list[i][0])

colMat.append(pts\_list[i][1])

colMat = np.matrix(colMat).transpose()

return colMat

def find\_tfrom\_between\_shapes(self, from\_shape, to\_shape):

"""

find transform between shapes

Parameters:

----------

from\_shape:

to\_shape:

Retures:

-------

tran\_m:

tran\_b:

"""

assert from\_shape.shape[0] == to\_shape.shape[0] and from\_shape.shape[0] % 2 == 0

sigma\_from = 0.0

sigma\_to = 0.0

cov = np.matrix([[0.0, 0.0], [0.0, 0.0]])

# compute the mean and cov

from\_shape\_points = from\_shape.reshape(int(from\_shape.shape[0] / 2), 2)

to\_shape\_points = to\_shape.reshape(int(to\_shape.shape[0] / 2), 2)

mean\_from = from\_shape\_points.mean(axis=0)

mean\_to = to\_shape\_points.mean(axis=0)

for i in range(from\_shape\_points.shape[0]):

temp\_dis = np.linalg.norm(from\_shape\_points[i] - mean\_from)

sigma\_from += temp\_dis \* temp\_dis

temp\_dis = np.linalg.norm(to\_shape\_points[i] - mean\_to)

sigma\_to += temp\_dis \* temp\_dis

cov += (to\_shape\_points[i].transpose() - mean\_to.transpose()) \* (from\_shape\_points[i] - mean\_from)

sigma\_from = sigma\_from / to\_shape\_points.shape[0]

sigma\_to = sigma\_to / to\_shape\_points.shape[0]

cov = cov / to\_shape\_points.shape[0]

# compute the affine matrix

s = np.matrix([[1.0, 0.0], [0.0, 1.0]])

u, d, vt = np.linalg.svd(cov)

if np.linalg.det(cov) < 0:

if d[1] < d[0]:

s[1, 1] = -1

else:

s[0, 0] = -1

r = u \* s \* vt

c = 1.0

if sigma\_from != 0:

c = 1.0 / sigma\_from \* np.trace(np.diag(d) \* s)

tran\_b = mean\_to.transpose() - c \* r \* mean\_from.transpose()

tran\_m = c \* r

return tran\_m, tran\_b

def align(self, desired\_size, img, landmarks, padding=0.1):

"""

Align face in BGR format.

:param size: size image

:type size: number

:param img\_face: face image detected

:type img\_face: array 3D

:return aligned\_face: align face

:rtype aligned\_face: array 3D

:return pos: position of face

:rtype pos: 'Left', 'Center', 'Right'

"""

shape = []

for k in range(int(len(landmarks) / 2)):

shape.append(landmarks[k])

shape.append(landmarks[k + 5])

if padding > 0:

padding = padding

else:

padding = 0

# average positions of face points

mean\_face\_shape\_x = [0.224152, 0.75610125, 0.490127, 0.254149, 0.726104]

mean\_face\_shape\_y = [0.2119465, 0.2119465, 0.628106, 0.780233, 0.780233]

from\_points = []

to\_points = []

for i in range(int(len(shape) / 2)):

x = (padding + mean\_face\_shape\_x[i]) / (2 \* padding + 1) \* desired\_size

y = (padding + mean\_face\_shape\_y[i]) / (2 \* padding + 1) \* desired\_size

to\_points.append([x, y])

from\_points.append([shape[2 \* i], shape[2 \* i + 1]])

# convert the points to Mat

from\_mat = self.list2colmatrix(from\_points)

to\_mat = self.list2colmatrix(to\_points)

# compute the similar transfrom

tran\_m, tran\_b = self.find\_tfrom\_between\_shapes(from\_mat, to\_mat)

probe\_vec = np.matrix([1.0, 0.0]).transpose()

probe\_vec = tran\_m \* probe\_vec

scale = np.linalg.norm(probe\_vec)

angle = 180.0 / math.pi \* math.atan2(probe\_vec[1, 0], probe\_vec[0, 0])

from\_center = [(shape[0] + shape[2]) / 2.0, (shape[1] + shape[3]) / 2.0]

to\_center = [0, 0]

to\_center[1] = desired\_size \* 0.4

to\_center[0] = desired\_size \* 0.5

ex = to\_center[0] - from\_center[0]

ey = to\_center[1] - from\_center[1]

rot\_mat = cv2.getRotationMatrix2D((from\_center[0], from\_center[1]), -1 \* angle, scale)

rot\_mat[0][2] += ex

rot\_mat[1][2] += ey

chips = cv2.warpAffine(img, rot\_mat, (desired\_size, desired\_size))

return chips, self.getPos(landmarks)

**Attendance:**

import cv2

from align\_custom import AlignCustom

from face\_feature import FaceFeature

from mtcnn\_detect import MTCNNDetect

from tf\_graph import FaceRecGraph

import argparse

import sys

import json

import time

import numpy as np

from tkinter import \*

from tkinter import messagebox, filedialog, ttk

import mysql.connector as mysql

import datetime

TIMEOUT = 10 #10 seconds

def main(args):

camera\_recog();

def camera\_recog():

print("[INFO] camera sensor warming up...")

con=mysql.connect(host="localhost",user="root",password="Khattak1420@",database="ais")

cursor=con.cursor()

vs = cv2.VideoCapture(0); #get input from webcam

detect\_time = time.time()

a = []

currentDT = datetime.datetime.now()

present= "Present"

classid="BSCS"

today = str(currentDT)

while True:

\_,frame = vs.read();

#u can certainly add a roi here but for the sake of a demo i'll just leave it as simple as this

rects, landmarks = face\_detect.detect\_face(frame,80);#min face size is set to 80x80

aligns = []

positions = []

for (i, rect) in enumerate(rects):

aligned\_face, face\_pos = aligner.align(160,frame,landmarks[:,i])

if len(aligned\_face) == 160 and len(aligned\_face[0]) == 160:

aligns.append(aligned\_face)

positions.append(face\_pos)

else:

print("Align face failed") #log

if(len(aligns) > 0):

features\_arr = extract\_feature.get\_features(aligns)

recog\_data = findPeople(features\_arr,positions)

for (i,rect) in enumerate(rects):

cv2.rectangle(frame,(rect[0],rect[1]),(rect[2],rect[3]),(255,0,0)) #draw bounding box for the face

cv2.putText(frame,recog\_data[i][0]+" - "+str(recog\_data[i][1])+"%",(rect[0],rect[1]),cv2.FONT\_HERSHEY\_SIMPLEX,1,(255,255,255),1,cv2.LINE\_AA)

student = recog\_data[i][0]

if student not in a:

a.append(student)

if 'Unknown' in a:

a.remove('Unknown')

print (a)

cv2.imshow("Frame",frame)

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

break

for i in a:

cursor.execute("INSERT INTO attendance (class,student,date,stat) values('"+classid+"','"+i+"','"+today+"','"+present+"')")

cursor.execute("commit");

root = Tk()

root.withdraw()

messagebox.showinfo("Attendance","Attendance Marked Successfully")

vs.release()

cv2.destroyAllWindows()

#root.destroy()

'''

facerec\_128D.txt Data Structure:

{

"Person ID": {

"Center": [[128D vector]],

"Left": [[128D vector]],

"Right": [[128D Vector]]

}

}

This function basically does a simple linear search for

^the 128D vector with the min distance to the 128D vector of the face on screen

'''

def findPeople(features\_arr, positions, thres = 0.6, percent\_thres = 70):

'''

:param features\_arr: a list of 128d Features of all faces on screen

:param positions: a list of face position types of all faces on screen

:param thres: distance threshold

:return: person name and percentage

'''

f = open('./facerec\_128D.txt','r')

data\_set = json.loads(f.read());

returnRes = [];

for (i,features\_128D) in enumerate(features\_arr):

result = "Unknown";

smallest = sys.maxsize

for person in data\_set.keys():

person\_data = data\_set[person][positions[i]];

for data in person\_data:

distance = np.sqrt(np.sum(np.square(data-features\_128D)))

if(distance < smallest):

smallest = distance;

result = person;

percentage = min(100, 100 \* thres / smallest)

if percentage <= percent\_thres :

result = "Unknown"

returnRes.append((result,percentage))

return returnRes

if \_\_name\_\_ == '\_\_main\_\_':

parser = argparse.ArgumentParser()

parser.add\_argument("--mode", type=str, help="Run camera recognition", default="camera")

args = parser.parse\_args(sys.argv[1:]);

FRGraph = FaceRecGraph();

MTCNNGraph = FaceRecGraph();

aligner = AlignCustom();

extract\_feature = FaceFeature(FRGraph)

face\_detect = MTCNNDetect(MTCNNGraph, scale\_factor=2); #scale\_factor, rescales image for faster detection

main(args);

**eye\_gaze:**

import cv2

import numpy as np

import dlib

from math import hypot

import time

import sys

from playsound import playsound

import winsound

cap = cv2.VideoCapture(0)

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor("shape\_predictor\_68\_face\_landmarks.dat")

def midpoint(p1 ,p2):

return int((p1.x + p2.x)/2), int((p1.y + p2.y)/2)

font = cv2.FONT\_HERSHEY\_PLAIN

def get\_gaze\_ratio(eye\_points, facial\_landmarks):

left\_eye\_region = np.array([(facial\_landmarks.part(eye\_points[0]).x, facial\_landmarks.part(eye\_points[0]).y),

(facial\_landmarks.part(eye\_points[1]).x, facial\_landmarks.part(eye\_points[1]).y),

(facial\_landmarks.part(eye\_points[2]).x, facial\_landmarks.part(eye\_points[2]).y),

(facial\_landmarks.part(eye\_points[3]).x, facial\_landmarks.part(eye\_points[3]).y),

(facial\_landmarks.part(eye\_points[4]).x, facial\_landmarks.part(eye\_points[4]).y),

(facial\_landmarks.part(eye\_points[5]).x, facial\_landmarks.part(eye\_points[5]).y)], np.int32)

# cv2.polylines(frame, [left\_eye\_region], True, (0, 0, 255), 2)

height, width, \_ = frame.shape

mask = np.zeros((height, width), np.uint8)

cv2.polylines(mask, [left\_eye\_region], True, 255, 2)

cv2.fillPoly(mask, [left\_eye\_region], 255)

eye = cv2.bitwise\_and(gray, gray, mask=mask)

min\_x = np.min(left\_eye\_region[:, 0])

max\_x = np.max(left\_eye\_region[:, 0])

min\_y = np.min(left\_eye\_region[:, 1])

max\_y = np.max(left\_eye\_region[:, 1])

gray\_eye = eye[min\_y: max\_y, min\_x: max\_x]

\_, threshold\_eye = cv2.threshold(gray\_eye, 70, 255, cv2.THRESH\_BINARY)

height, width = threshold\_eye.shape

left\_side\_threshold = threshold\_eye[0: height, 0: int(width / 2)]

left\_side\_white = cv2.countNonZero(left\_side\_threshold)

right\_side\_threshold = threshold\_eye[0: height, int(width / 2): width]

right\_side\_white = cv2.countNonZero(right\_side\_threshold)

if left\_side\_white == 0:

gaze\_ratio = 1

elif right\_side\_white == 0:

gaze\_ratio = 5

else:

gaze\_ratio = left\_side\_white / right\_side\_white

return gaze\_ratio

while True:

\_, frame = cap.read()

new\_frame = np.zeros((500, 500, 3), np.uint8)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = detector(gray)

for face in faces:

x, y = face.left(), face.top()

x1, y1 = face.right(), face.bottom()

cv2.rectangle(frame, (x, y), (x1, y1), (0, 255, 0), 2)

landmarks = predictor(gray, face)

# Gaze detection

gaze\_ratio\_left\_eye = get\_gaze\_ratio([36, 37, 38, 39, 40, 41], landmarks)

gaze\_ratio\_right\_eye = get\_gaze\_ratio([42, 43, 44, 45, 46, 47], landmarks)

gaze\_ratio = (gaze\_ratio\_right\_eye + gaze\_ratio\_left\_eye) / 2

#winsound.PlaySound("alertt",winsound.SND\_FILENAME)

if gaze\_ratio <= 0.60:

new\_frame[:] = (0, 0, 255)

cv2.putText(frame, "Left", (50, 100), font, 2, (0, 0, 255), 3)

#playsound('alert.mp3')

playsound('alert\_new.mp3')

print(gaze\_ratio)

#playsound('alert\_new.mp3')

start = time.time()

while time.time() > start + 5:

cv2.rectangle(frame, (x, y), (x1, y1), (0, 0, 255), 2)

playsound('alert\_new.mp3')

print(gaze\_ratio)

elif gaze\_ratio > 1.10:

cv2.putText(frame, "Right", (50, 100), font, 2, (0, 0, 255), 3)

print(gaze\_ratio)

playsound('alert\_new.mp3')

else:

new\_frame[:] = (255, 0, 0)

cv2.putText(frame, "Center", (50, 100), font, 2, (0, 0, 255), 3)

#playsound('alert\_new.mp3')

print(gaze\_ratio)

# start = time.time()

# while time.time() > start + 5:

# cv2.rectangle(frame, (x, y), (x1, y1), (0, 0, 255), 2)

# playsound('alert.mp3')

cv2.imshow("Frame", frame)

# cv2.imshow("New frame", new\_frame)

key = cv2.waitKey(1)

if key == 27:

break

cap.release()

cv2.destroyAllWindows()

**face\_feature:**

'''

Run the pretrained model to extract 128D face features

'''

import tensorflow as tf

from architecture import inception\_resnet\_v1 as resnet

from tensorflow.python.platform import gfile

import numpy as np

import os

class FaceFeature(object):

def \_\_init\_\_(self, face\_rec\_graph, model\_path = 'models/20170512-110547.pb'):

'''

:param face\_rec\_sess: FaceRecSession object

:param model\_path:

'''

print("Loading model...")

with face\_rec\_graph.graph.as\_default():

self.sess = tf.Session()

with self.sess.as\_default():

self.\_\_load\_model(model\_path)

self.x = tf.get\_default\_graph() \

.get\_tensor\_by\_name("input:0")

self.embeddings = tf.get\_default\_graph() \

.get\_tensor\_by\_name("embeddings:0")

self.phase\_train\_placeholder = tf.get\_default\_graph() \

.get\_tensor\_by\_name("phase\_train:0")

print("Model loaded")

def get\_features(self, input\_imgs):

images = load\_data\_list(input\_imgs,160)

feed\_dict = {self.x: images, self.phase\_train\_placeholder: False}

return self.sess.run(self.embeddings, feed\_dict = feed\_dict)

def \_\_load\_model(self, model):

# Check if the model is a model directory (containing a metagraph and a checkpoint file)

# or if it is a protobuf file with a frozen graph

model\_exp = os.path.expanduser(model)

if os.path.isfile(model\_exp):

print('Model filename: %s' % model\_exp)

with gfile.FastGFile(model\_exp, 'rb') as file\_:

graph\_def = tf.GraphDef()

graph\_def.ParseFromString(file\_.read())

tf.import\_graph\_def(graph\_def, name='')

else:

print('Model directory: %s' % model\_exp)

meta\_file, ckpt\_file = get\_model\_filenames(model\_exp)

print('Metagraph file: %s' % meta\_file)

print('Checkpoint file: %s' % ckpt\_file)

saver = tf.train.import\_meta\_graph(os.path.join(model\_exp, meta\_file))

saver.restore(tf.get\_default\_session(), os.path.join(model\_exp, ckpt\_file))

def get\_model\_filenames(model\_dir):

files = os.listdir(model\_dir)

meta\_files = [s for s in files if s.endswith('.meta')]

if len(meta\_files) == 0:

raise ValueError('No meta file found in the model directory (%s)' % model\_dir)

elif len(meta\_files) > 1:

raise ValueError('There should not be more than one meta file \

in the model directory (%s)' % model\_dir)

meta\_file = meta\_files[0]

meta\_files = [s for s in files if '.ckpt' in s]

max\_step = -1

for file\_ in files:

step\_str = re.match(r'(^model-[\w\- ]+.ckpt-(\d+))', file\_)

if step\_str is not None and len(step\_str.groups()) >= 2:

step = int(step\_str.groups()[1])

if step > max\_step:

max\_step = step

ckpt\_file = step\_str.groups()[0]

return meta\_file, ckpt\_file

def tensorization(img):

'''

Prepare the imgs before input into model

:param img: Single face image

:return tensor: numpy array in shape(n, 160, 160, 3) ready for input to cnn

'''

tensor = img.reshape(-1, Config.Align.IMAGE\_SIZE, Config.Align.IMAGE\_SIZE, 3)

return tensor

#some image preprocess stuff

def prewhiten(x):

mean = np.mean(x)

std = np.std(x)

std\_adj = np.maximum(std, 1.0 / np.sqrt(x.size))

y = np.multiply(np.subtract(x, mean), 1 / std\_adj)

return y

def load\_data\_list(imgList, image\_size, do\_prewhiten=True):

images = np.zeros((len(imgList), image\_size, image\_size, 3))

i = 0

for img in imgList:

if img is not None:

if do\_prewhiten:

img = prewhiten(img)

images[i, :, :, :] = img

i += 1

return images

**headpose:**

import cv2

import dlib

import numpy as np

from imutils import face\_utils

from decimal import Decimal

import math

import sys

import time

from playsound import playsound

import winsound

face\_landmark\_path = './shape\_predictor\_68\_face\_landmarks.dat'

K = [6.5308391993466671e+002, 0.0, 3.1950000000000000e+002,

0.0, 6.5308391993466671e+002, 2.3950000000000000e+002,

0.0, 0.0, 1.0]

D = [7.0834633684407095e-002, 6.9140193737175351e-002, 0.0, 0.0, -1.3073460323689292e+000]

cam\_matrix = np.array(K).reshape(3, 3).astype(np.float32)

dist\_coeffs = np.array(D).reshape(5, 1).astype(np.float32)

object\_pts = np.float32([[6.825897, 6.760612, 4.402142],

[1.330353, 7.122144, 6.903745],

[-1.330353, 7.122144, 6.903745],

[-6.825897, 6.760612, 4.402142],

[5.311432, 5.485328, 3.987654],

[1.789930, 5.393625, 4.413414],

[-1.789930, 5.393625, 4.413414],

[-5.311432, 5.485328, 3.987654],

[2.005628, 1.409845, 6.165652],

[-2.005628, 1.409845, 6.165652],

[2.774015, -2.080775, 5.048531],

[-2.774015, -2.080775, 5.048531],

[0.000000, -3.116408, 6.097667],

[0.000000, -7.415691, 4.070434]])

reprojectsrc = np.float32([[10.0, 10.0, 10.0],

[10.0, 10.0, -10.0],

[10.0, -10.0, -10.0],

[10.0, -10.0, 10.0],

[-10.0, 10.0, 10.0],

[-10.0, 10.0, -10.0],

[-10.0, -10.0, -10.0],

[-10.0, -10.0, 10.0]])

line\_pairs = [[0, 1], [1, 2], [2, 3], [3, 0],

[4, 5], [5, 6], [6, 7], [7, 4],

[0, 4], [1, 5], [2, 6], [3, 7]]

def get\_head\_pose(shape):

image\_pts = np.float32([shape[17], shape[21], shape[22], shape[26], shape[36],

shape[39], shape[42], shape[45], shape[31], shape[35],

shape[48], shape[54], shape[57], shape[8]])

\_, rotation\_vec, translation\_vec = cv2.solvePnP(object\_pts, image\_pts, cam\_matrix, dist\_coeffs)

reprojectdst, \_ = cv2.projectPoints(reprojectsrc, rotation\_vec, translation\_vec, cam\_matrix,

dist\_coeffs)

reprojectdst = tuple(map(tuple, reprojectdst.reshape(8, 2)))

# calc euler angle

rotation\_mat, \_ = cv2.Rodrigues(rotation\_vec)

pose\_mat = cv2.hconcat((rotation\_mat, translation\_vec))

\_, \_, \_, \_, \_, \_, euler\_angle = cv2.decomposeProjectionMatrix(pose\_mat)

return reprojectdst, euler\_angle

def main():

# return

cap = cv2.VideoCapture(0)

if not cap.isOpened():

print("Unable to connect to camera.")

return

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor(face\_landmark\_path)

while cap.isOpened():

ret, frame = cap.read()

if ret:

face\_rects = detector(frame, 0)

if len(face\_rects) > 0:

shape = predictor(frame, face\_rects[0])

shape = face\_utils.shape\_to\_np(shape)

reprojectdst, euler\_angle = get\_head\_pose(shape)

face\_angle=Decimal("{:7.2f}".format(euler\_angle[1, 0],(20, 50), cv2.FONT\_HERSHEY\_SIMPLEX,0.75, (0, 0, 0), thickness=2))

if(face\_angle>=-20) and (face\_angle<=20):

for (x, y) in shape:

cv2.circle(frame, (x, y), 1, (0, 255, 0), -1)

#print("Normal")

cv2.putText(frame,"Normal", (20, 20), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (0, 0, 0) , thickness=2)

else:

cv2.putText(frame,"Not Normal", (20, 20), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (0, 0, 0) , thickness=2)

playsound('alert\_new.mp3')

cv2.imshow("AIS Invigilation", frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break;

cap.release()

cv2.destroyAllWindows()

if \_\_name\_\_ == '\_\_main\_\_':

main()

**invigilation:**

import tkinter

from tkinter import \*

from tkinter import messagebox

import os

def invigilation():

os.system("python headpose.py")

def invigilator():

def main\_menu():

root.destroy()

os.system("python main\_menu.py")

root=tkinter.Tk()

#root.configure(bg='blue')

root.geometry("1920x1080")

photo=PhotoImage(file="ui/invigilator\_panel.png")

photo2=PhotoImage(file="ui/invigilation1.png")

photo3=PhotoImage(file="ui/invigilation2.png")

photo4=PhotoImage(file="ui/signout.png")

btn= Button(

root,

image=photo,

border=0,

text="",

compound=TOP,

)

btn.pack()

btn2= Button(

root,

image=photo2,

command=invigilation,

border=0,

text="",

compound=TOP,

)

btn2.pack()

btn3= Button(

root,

image=photo3,

command=invigilation,

border=0,

text="",

compound=TOP,

)import tkinter

from tkinter import \*

from tkinter import messagebox

import os

def invigilation():

os.system("python headpose.py")

def invigilator():

def main\_menu():

root.destroy()

os.system("python main\_menu.py")

root=tkinter.Tk()

#root.configure(bg='blue')

root.geometry("1920x1080")

photo=PhotoImage(file="ui/invigilator\_panel.png")

photo2=PhotoImage(file="ui/invigilation1.png")

photo3=PhotoImage(file="ui/invigilation2.png")

photo4=PhotoImage(file="ui/signout.png")

btn= Button(

root,

image=photo,

border=0,

text="",

compound=TOP,

)

btn.pack()

btn2= Button(

root,

image=photo2,

command=invigilation,

border=0,

text="",

compound=TOP,

)

btn2.pack()

btn3= Button(

root,

image=photo3,

command=invigilation,

border=0,

text="",

compound=TOP,

)

btn3.pack()

btn4= Button(

root,

image=photo4,

command=main\_menu,

border=0,

text="",

compound=TOP,

)

btn4.pack()

root.mainloop()

invigilator()

btn3.pack()

btn4= Button(

root,

image=photo4,

command=main\_menu,

border=0,

text="",

compound=TOP,

)

btn4.pack()

root.mainloop()

invigilator()

**MTCNN\_detect:**

'''

Tensorflow implementation of the mtcnn face detection algorithm

'''

from six import string\_types, iteritems

import numpy as np

import tensorflow as tf

import cv2

import os

class MTCNNDetect(object):

def \_\_init\_\_(self, face\_rec\_graph, model\_path = "models", threshold = [0.6, 0.7, 0.7], factor = 0.709, scale\_factor = 1):

'''

:param face\_rec\_sess: FaceRecSession

:param threshold: detection threshold

:param factor: default 0.709 image pyramid -- magic number

:param model\_path:

'''

self.threshold = threshold

self.factor = factor

self.scale\_factor = scale\_factor;

with face\_rec\_graph.graph.as\_default():

print("Loading MTCNN Face detection model")

self.sess = tf.Session()

if not model\_path:

model\_path, \_ = os.path.split(os.path.realpath(\_\_file\_\_))

with tf.variable\_scope('pnet'):

data = tf.placeholder(tf.float32, (None, None, None, 3), 'input')

pnet = PNet({'data': data})

pnet.load(os.path.join(model\_path, 'det1.npy'), self.sess)

with tf.variable\_scope('rnet'):

data = tf.placeholder(tf.float32, (None, 24, 24, 3), 'input')

rnet = RNet({'data': data})

rnet.load(os.path.join(model\_path, 'det2.npy'), self.sess)

with tf.variable\_scope('onet'):

data = tf.placeholder(tf.float32, (None, 48, 48, 3), 'input')

onet = ONet({'data': data})

onet.load(os.path.join(model\_path, 'det3.npy'), self.sess)

self.pnet = lambda img: self.sess.run(('pnet/conv4-2/BiasAdd:0', 'pnet/prob1:0'), feed\_dict={'pnet/input:0': img})

self.rnet = lambda img: self.sess.run(('rnet/conv5-2/conv5-2:0', 'rnet/prob1:0'), feed\_dict={'rnet/input:0': img})

self.onet = lambda img: self.sess.run(('onet/conv6-2/conv6-2:0', 'onet/conv6-3/conv6-3:0', 'onet/prob1:0'),

feed\_dict={'onet/input:0': img})

print("MTCNN Model loaded")

def detect\_face(self, img, minsize):

# im: input image

# minsize: minimum of faces' size

if(self.scale\_factor > 1):

img = cv2.resize(img,(int(len(img[0])/self.scale\_factor), int(len(img)/self.scale\_factor)))

factor\_count = 0

total\_boxes = np.empty((0, 9))

points = []

h = img.shape[0]

w = img.shape[1]

minl = np.amin([h, w])

m = 12.0 / minsize

minl = minl \* m

# creat scale pyramid

scales = []

while minl >= 12:

scales += [m \* np.power(self.factor, factor\_count)]

minl = minl \* self.factor

factor\_count += 1

# first stage

for j in range(len(scales)):

scale = scales[j]

hs = int(np.ceil(h \* scale))

ws = int(np.ceil(w \* scale))

im\_data = imresample(img, (hs, ws))

im\_data = (im\_data - 127.5) \* 0.0078125

img\_x = np.expand\_dims(im\_data, 0)

img\_y = np.transpose(img\_x, (0, 2, 1, 3))

out = self.pnet(img\_y)

out0 = np.transpose(out[0], (0, 2, 1, 3))

out1 = np.transpose(out[1], (0, 2, 1, 3))

boxes, \_ = generateBoundingBox(out1[0, :, :, 1].copy(), out0[0, :, :, :].copy(), scale, self.threshold[0])

# inter-scale nms

pick = nms(boxes.copy(), 0.5, 'Union')

if boxes.size > 0 and pick.size > 0:

boxes = boxes[pick, :]

total\_boxes = np.append(total\_boxes, boxes, axis=0)

numbox = total\_boxes.shape[0]

if numbox > 0:

pick = nms(total\_boxes.copy(), 0.7, 'Union')

total\_boxes = total\_boxes[pick, :]

regw = total\_boxes[:, 2] - total\_boxes[:, 0]

regh = total\_boxes[:, 3] - total\_boxes[:, 1]

qq1 = total\_boxes[:, 0] + total\_boxes[:, 5] \* regw

qq2 = total\_boxes[:, 1] + total\_boxes[:, 6] \* regh

qq3 = total\_boxes[:, 2] + total\_boxes[:, 7] \* regw

qq4 = total\_boxes[:, 3] + total\_boxes[:, 8] \* regh

total\_boxes = np.transpose(np.vstack([qq1, qq2, qq3, qq4, total\_boxes[:, 4]]))

total\_boxes = rerec(total\_boxes.copy())

total\_boxes[:, 0:4] = np.fix(total\_boxes[:, 0:4]).astype(np.int32)

dy, edy, dx, edx, y, ey, x, ex, tmpw, tmph = pad(total\_boxes.copy(), w, h)

numbox = total\_boxes.shape[0]

if numbox > 0:

# second stage

tempimg = np.zeros((24, 24, 3, numbox))

for k in range(0, numbox):

tmp = np.zeros((int(tmph[k]), int(tmpw[k]), 3))

tmp[dy[k] - 1:edy[k], dx[k] - 1:edx[k], :] = img[y[k] - 1:ey[k], x[k] - 1:ex[k], :]

if tmp.shape[0] > 0 and tmp.shape[1] > 0 or tmp.shape[0] == 0 and tmp.shape[1] == 0:

tempimg[:, :, :, k] = imresample(tmp, (24, 24))

else:

return np.empty()

tempimg = (tempimg - 127.5) \* 0.0078125

tempimg1 = np.transpose(tempimg, (3, 1, 0, 2))

out = self.rnet(tempimg1)

out0 = np.transpose(out[0])

out1 = np.transpose(out[1])

score = out1[1, :]

ipass = np.where(score > self.threshold[1])

total\_boxes = np.hstack([total\_boxes[ipass[0], 0:4].copy(), np.expand\_dims(score[ipass].copy(), 1)])

mv = out0[:, ipass[0]]

if total\_boxes.shape[0] > 0:

pick = nms(total\_boxes, 0.7, 'Union')

total\_boxes = total\_boxes[pick, :]

total\_boxes = bbreg(total\_boxes.copy(), np.transpose(mv[:, pick]))

total\_boxes = rerec(total\_boxes.copy())

numbox = total\_boxes.shape[0]

if numbox > 0:

# third stage

total\_boxes = np.fix(total\_boxes).astype(np.int32)

dy, edy, dx, edx, y, ey, x, ex, tmpw, tmph = pad(total\_boxes.copy(), w, h)

tempimg = np.zeros((48, 48, 3, numbox))

for k in range(0, numbox):

tmp = np.zeros((int(tmph[k]), int(tmpw[k]), 3))

tmp[dy[k] - 1:edy[k], dx[k] - 1:edx[k], :] = img[y[k] - 1:ey[k], x[k] - 1:ex[k], :]

if tmp.shape[0] > 0 and tmp.shape[1] > 0 or tmp.shape[0] == 0 and tmp.shape[1] == 0:

tempimg[:, :, :, k] = imresample(tmp, (48, 48))

else:

return np.empty()

tempimg = (tempimg - 127.5) \* 0.0078125

tempimg1 = np.transpose(tempimg, (3, 1, 0, 2))

out = self.onet(tempimg1)

out0 = np.transpose(out[0])

out1 = np.transpose(out[1])

out2 = np.transpose(out[2])

score = out2[1, :]

points = out1

ipass = np.where(score > self.threshold[2])

points = points[:, ipass[0]]

total\_boxes = np.hstack([total\_boxes[ipass[0], 0:4].copy(), np.expand\_dims(score[ipass].copy(), 1)])

mv = out0[:, ipass[0]]

w = total\_boxes[:, 2] - total\_boxes[:, 0] + 1

h = total\_boxes[:, 3] - total\_boxes[:, 1] + 1

points[0:5, :] = np.tile(w, (5, 1)) \* points[0:5, :] + np.tile(total\_boxes[:, 0], (5, 1)) - 1

points[5:10, :] = np.tile(h, (5, 1)) \* points[5:10, :] + np.tile(total\_boxes[:, 1], (5, 1)) - 1

if total\_boxes.shape[0] > 0:

total\_boxes = bbreg(total\_boxes.copy(), np.transpose(mv))

pick = nms(total\_boxes.copy(), 0.7, 'Min')

total\_boxes = total\_boxes[pick, :]

points = points[:, pick]

# convert to int before return

# multiply conf 100 time to return a int

total\_boxes[:, 4] = total\_boxes[:, 4] \* 100

total\_boxes = np.array((total\_boxes), dtype=int)

points = np.array((points), dtype=int)

return total\_boxes \* self.scale\_factor, points \* self.scale\_factor

def layer(op):

'''Decorator for composable network layers.'''

def layer\_decorated(self, \*args, \*\*kwargs):

# Automatically set a name if not provided.

name = kwargs.setdefault('name', self.get\_unique\_name(op.\_\_name\_\_))

# Figure out the layer inputs.

if len(self.terminals) == 0:

raise RuntimeError('No input variables found for layer %s.' % name)

elif len(self.terminals) == 1:

layer\_input = self.terminals[0]

else:

layer\_input = list(self.terminals)

# Perform the operation and get the output.

layer\_output = op(self, layer\_input, \*args, \*\*kwargs)

# Add to layer LUT.

self.layers[name] = layer\_output

# This output is now the input for the next layer.

self.feed(layer\_output)

# Return self for chained calls.

return self

return layer\_decorated

class Network(object):

def \_\_init\_\_(self, inputs, trainable=True):

# The input nodes for this network

self.inputs = inputs

# The current list of terminal nodes

self.terminals = []

# Mapping from layer names to layers

self.layers = dict(inputs)

# If true, the resulting variables are set as trainable

self.trainable = trainable

self.setup()

def setup(self):

'''Construct the network. '''

raise NotImplementedError('Must be implemented by the subclass.')

def load(self, data\_path, session, ignore\_missing=False):

'''Load network weights.

data\_path: The path to the numpy-serialized network weights

session: The current TensorFlow session

ignore\_missing: If true, serialized weights for missing layers are ignored.

'''

data\_dict = np.load(data\_path, encoding='latin1').item() # pylint: disable=no-member

for op\_name in data\_dict:

with tf.variable\_scope(op\_name, reuse=True):

for param\_name, data in iteritems(data\_dict[op\_name]):

try:

var = tf.get\_variable(param\_name)

session.run(var.assign(data))

except ValueError:

if not ignore\_missing:

raise

def feed(self, \*args):

'''Set the input(s) for the next operation by replacing the terminal nodes.

The arguments can be either layer names or the actual layers.

'''

assert len(args) != 0

self.terminals = []

for fed\_layer in args:

if isinstance(fed\_layer, string\_types):

try:

fed\_layer = self.layers[fed\_layer]

except KeyError:

raise KeyError('Unknown layer name fed: %s' % fed\_layer)

self.terminals.append(fed\_layer)

return self

def get\_output(self):

'''Returns the current network output.'''

return self.terminals[-1]

def get\_unique\_name(self, prefix):

'''Returns an index-suffixed unique name for the given prefix.

This is used for auto-generating layer names based on the type-prefix.

'''

ident = sum(t.startswith(prefix) for t, \_ in self.layers.items()) + 1

return '%s\_%d' % (prefix, ident)

def make\_var(self, name, shape):

'''Creates a new TensorFlow variable.'''

return tf.get\_variable(name, shape, trainable=self.trainable)

def validate\_padding(self, padding):

'''Verifies that the padding is one of the supported ones.'''

assert padding in ('SAME', 'VALID')

@layer

def conv(self,

inp,

k\_h,

k\_w,

c\_o,

s\_h,

s\_w,

name,

relu=True,

padding='SAME',

group=1,

biased=True):

# Verify that the padding is acceptable

self.validate\_padding(padding)

# Get the number of channels in the input

c\_i = int(inp.get\_shape()[-1])

# Verify that the grouping parameter is valid

assert c\_i % group == 0

assert c\_o % group == 0

# Convolution for a given input and kernel

convolve = lambda i, k: tf.nn.conv2d(i, k, [1, s\_h, s\_w, 1], padding=padding)

with tf.variable\_scope(name) as scope:

kernel = self.make\_var('weights', shape=[k\_h, k\_w, c\_i // group, c\_o])

# This is the common-case. Convolve the input without any further complications.

output = convolve(inp, kernel)

# Add the biases

if biased:

biases = self.make\_var('biases', [c\_o])

output = tf.nn.bias\_add(output, biases)

if relu:

# ReLU non-linearity

output = tf.nn.relu(output, name=scope.name)

return output

@layer

def prelu(self, inp, name):

with tf.variable\_scope(name):

i = int(inp.get\_shape()[-1])

alpha = self.make\_var('alpha', shape=(i,))

output = tf.nn.relu(inp) + tf.multiply(alpha, -tf.nn.relu(-inp))

return output

@layer

def max\_pool(self, inp, k\_h, k\_w, s\_h, s\_w, name, padding='SAME'):

self.validate\_padding(padding)

return tf.nn.max\_pool(inp,

ksize=[1, k\_h, k\_w, 1],

strides=[1, s\_h, s\_w, 1],

padding=padding,

name=name)

@layer

def fc(self, inp, num\_out, name, relu=True):

with tf.variable\_scope(name):

input\_shape = inp.get\_shape()

if input\_shape.ndims == 4:

# The input is spatial. Vectorize it first.

dim = 1

for d in input\_shape[1:].as\_list():

dim \*= int(d)

feed\_in = tf.reshape(inp, [-1, dim])

else:

feed\_in, dim = (inp, input\_shape[-1].value)

weights = self.make\_var('weights', shape=[dim, num\_out])

biases = self.make\_var('biases', [num\_out])

op = tf.nn.relu\_layer if relu else tf.nn.xw\_plus\_b

fc = op(feed\_in, weights, biases, name=name)

return fc

"""

Multi dimensional softmax,

refer to https://github.com/tensorflow/tensorflow/issues/210

compute softmax along the dimension of target

the native softmax only supports batch\_size x dimension

"""

@layer

def softmax(self, target, axis, name=None):

max\_axis = tf.reduce\_max(target, axis, keep\_dims=True)

target\_exp = tf.exp(target - max\_axis)

normalize = tf.reduce\_sum(target\_exp, axis, keep\_dims=True)

softmax = tf.div(target\_exp, normalize, name)

return softmax

class PNet(Network):

def setup(self):

(self.feed('data') # pylint: disable=no-value-for-parameter, no-member

.conv(3, 3, 10, 1, 1, padding='VALID', relu=False, name='conv1')

.prelu(name='PReLU1')

.max\_pool(2, 2, 2, 2, name='pool1')

.conv(3, 3, 16, 1, 1, padding='VALID', relu=False, name='conv2')

.prelu(name='PReLU2')

.conv(3, 3, 32, 1, 1, padding='VALID', relu=False, name='conv3')

.prelu(name='PReLU3')

.conv(1, 1, 2, 1, 1, relu=False, name='conv4-1')

.softmax(3, name='prob1'))

(self.feed('PReLU3') # pylint: disable=no-value-for-parameter

.conv(1, 1, 4, 1, 1, relu=False, name='conv4-2'))

class RNet(Network):

def setup(self):

(self.feed('data') # pylint: disable=no-value-for-parameter, no-member

.conv(3, 3, 28, 1, 1, padding='VALID', relu=False, name='conv1')

.prelu(name='prelu1')

.max\_pool(3, 3, 2, 2, name='pool1')

.conv(3, 3, 48, 1, 1, padding='VALID', relu=False, name='conv2')

.prelu(name='prelu2')

.max\_pool(3, 3, 2, 2, padding='VALID', name='pool2')

.conv(2, 2, 64, 1, 1, padding='VALID', relu=False, name='conv3')

.prelu(name='prelu3')

.fc(128, relu=False, name='conv4')

.prelu(name='prelu4')

.fc(2, relu=False, name='conv5-1')

.softmax(1, name='prob1'))

(self.feed('prelu4') # pylint: disable=no-value-for-parameter

.fc(4, relu=False, name='conv5-2'))

class ONet(Network):

def setup(self):

(self.feed('data') # pylint: disable=no-value-for-parameter, no-member

.conv(3, 3, 32, 1, 1, padding='VALID', relu=False, name='conv1')

.prelu(name='prelu1')

.max\_pool(3, 3, 2, 2, name='pool1')

.conv(3, 3, 64, 1, 1, padding='VALID', relu=False, name='conv2')

.prelu(name='prelu2')

.max\_pool(3, 3, 2, 2, padding='VALID', name='pool2')

.conv(3, 3, 64, 1, 1, padding='VALID', relu=False, name='conv3')

.prelu(name='prelu3')

.max\_pool(2, 2, 2, 2, name='pool3')

.conv(2, 2, 128, 1, 1, padding='VALID', relu=False, name='conv4')

.prelu(name='prelu4')

.fc(256, relu=False, name='conv5')

.prelu(name='prelu5')

.fc(2, relu=False, name='conv6-1')

.softmax(1, name='prob1'))

(self.feed('prelu5') # pylint: disable=no-value-for-parameter

.fc(4, relu=False, name='conv6-2'))

(self.feed('prelu5') # pylint: disable=no-value-for-parameter

.fc(10, relu=False, name='conv6-3'))

# function [boundingbox] = bbreg(boundingbox,reg)

def bbreg(boundingbox, reg):

# calibrate bounding boxes

if reg.shape[1] == 1:

reg = np.reshape(reg, (reg.shape[2], reg.shape[3]))

w = boundingbox[:, 2] - boundingbox[:, 0] + 1

h = boundingbox[:, 3] - boundingbox[:, 1] + 1

b1 = boundingbox[:, 0] + reg[:, 0] \* w

b2 = boundingbox[:, 1] + reg[:, 1] \* h

b3 = boundingbox[:, 2] + reg[:, 2] \* w

b4 = boundingbox[:, 3] + reg[:, 3] \* h

boundingbox[:, 0:4] = np.transpose(np.vstack([b1, b2, b3, b4]))

return boundingbox

def generateBoundingBox(imap, reg, scale, t):

# use heatmap to generate bounding boxes

stride = 2

cellsize = 12

imap = np.transpose(imap)

dx1 = np.transpose(reg[:, :, 0])

dy1 = np.transpose(reg[:, :, 1])

dx2 = np.transpose(reg[:, :, 2])

dy2 = np.transpose(reg[:, :, 3])

y, x = np.where(imap >= t)

if y.shape[0] == 1:

dx1 = np.flipud(dx1)

dy1 = np.flipud(dy1)

dx2 = np.flipud(dx2)

dy2 = np.flipud(dy2)

score = imap[(y, x)]

reg = np.transpose(np.vstack([dx1[(y, x)], dy1[(y, x)], dx2[(y, x)], dy2[(y, x)]]))

if reg.size == 0:

reg = np.empty((0, 3))

bb = np.transpose(np.vstack([y, x]))

q1 = np.fix((stride \* bb + 1) / scale)

q2 = np.fix((stride \* bb + cellsize - 1 + 1) / scale)

boundingbox = np.hstack([q1, q2, np.expand\_dims(score, 1), reg])

return boundingbox, reg

# function pick = nms(boxes,threshold,type)

def nms(boxes, threshold, method):

if boxes.size == 0:

return np.empty((0, 3))

x1 = boxes[:, 0]

y1 = boxes[:, 1]

x2 = boxes[:, 2]

y2 = boxes[:, 3]

s = boxes[:, 4]

area = (x2 - x1 + 1) \* (y2 - y1 + 1)

I = np.argsort(s)

pick = np.zeros\_like(s, dtype=np.int16)

counter = 0

while I.size > 0:

i = I[-1]

pick[counter] = i

counter += 1

idx = I[0:-1]

xx1 = np.maximum(x1[i], x1[idx])

yy1 = np.maximum(y1[i], y1[idx])

xx2 = np.minimum(x2[i], x2[idx])

yy2 = np.minimum(y2[i], y2[idx])

w = np.maximum(0.0, xx2 - xx1 + 1)

h = np.maximum(0.0, yy2 - yy1 + 1)

inter = w \* h

if method is 'Min':

o = inter / np.minimum(area[i], area[idx])

else:

o = inter / (area[i] + area[idx] - inter)

I = I[np.where(o <= threshold)]

pick = pick[0:counter]

return pick

# function [dy edy dx edx y ey x ex tmpw tmph] = pad(total\_boxes,w,h)

def pad(total\_boxes, w, h):

# compute the padding coordinates (pad the bounding boxes to square)

tmpw = (total\_boxes[:, 2] - total\_boxes[:, 0] + 1).astype(np.int32)

tmph = (total\_boxes[:, 3] - total\_boxes[:, 1] + 1).astype(np.int32)

numbox = total\_boxes.shape[0]

dx = np.ones((numbox), dtype=np.int32)

dy = np.ones((numbox), dtype=np.int32)

edx = tmpw.copy().astype(np.int32)

edy = tmph.copy().astype(np.int32)

x = total\_boxes[:, 0].copy().astype(np.int32)

y = total\_boxes[:, 1].copy().astype(np.int32)

ex = total\_boxes[:, 2].copy().astype(np.int32)

ey = total\_boxes[:, 3].copy().astype(np.int32)

tmp = np.where(ex > w)

edx.flat[tmp] = np.expand\_dims(-ex[tmp] + w + tmpw[tmp], 1)

ex[tmp] = w

tmp = np.where(ey > h)

edy.flat[tmp] = np.expand\_dims(-ey[tmp] + h + tmph[tmp], 1)

ey[tmp] = h

tmp = np.where(x < 1)

dx.flat[tmp] = np.expand\_dims(2 - x[tmp], 1)

x[tmp] = 1

tmp = np.where(y < 1)

dy.flat[tmp] = np.expand\_dims(2 - y[tmp], 1)

y[tmp] = 1

return dy, edy, dx, edx, y, ey, x, ex, tmpw, tmph

# function [bboxA] = rerec(bboxA)

def rerec(bboxA):

# convert bboxA to square

h = bboxA[:, 3] - bboxA[:, 1]

w = bboxA[:, 2] - bboxA[:, 0]

l = np.maximum(w, h)

bboxA[:, 0] = bboxA[:, 0] + w \* 0.5 - l \* 0.5

bboxA[:, 1] = bboxA[:, 1] + h \* 0.5 - l \* 0.5

bboxA[:, 2:4] = bboxA[:, 0:2] + np.transpose(np.tile(l, (2, 1)))

return bboxA

def imresample(img, sz):

im\_data = cv2.resize(img, (sz[1], sz[0]), interpolation=cv2.INTER\_AREA) # @UndefinedVariable

return im\_data

**tf\_graph:**

'''

Load pretrain models and create a tensorflow session to run them

'''

import tensorflow as tf

class FaceRecGraph(object):

def \_\_init\_\_(self):

'''

There'll be more to come in this class

'''

self.graph = tf.Graph();

**Chapter 7**

**Software Testing**

We'll talk about testing and evaluating our method in this chapter, and we'll use various testing techniques to do so. Testing is an essential part of the system development process because it aids in the detection of errors and ensures that the system is performing as intended. Testing also aids in ensuring that systems meet the required specifications. We can't treat testing as a step in our method, so we started testing from the beginning of the project because we use an iterative model.

## 7.1 GUI Testing

The graphical user interface (GUI) is the part of an application that the consumer sees. We check the functionality of buttons, menus, icons, and other elements in GUI testing to ensure that they operate as intended.

## Log-in interface test case

This test case identifies the login interface's non-working and working features.

|  |  |  |  |
| --- | --- | --- | --- |
| TC\_FUNCT\_01 | | | |
| Tests the Log-in Screen | | | |
| Spyder | | | |
| REQ\_FUNCT\_01 | | | |
| Spyder should be installed on system. | | | |
| Step | Task | Expected Result | Actual Resullt |
| 1 | Open the log-in screen | Pass/Fail | Pass |
| 2 | Check if the login screen is fully visible | Pass/Fail | Pass |
| 3 | Enter User-name and password. | Pass/Fail | Pass |
| 4 | Verify that the User-name can be entered. | Pass/Fail | Pass |
| 5 | Verify that the password is not showing and is entering | Pass/Fail | Pass |
| 6 | Verify that a submit and reset buttons are visible on the screen | Pass/Fail | Pass |
| 7 | Verify the functionality of each item on the screen. | Pass/Fail | Fail |

**Table 6.1: Login screen test case**

## Student Registration test case

## This test case helps identify that student is registered successfully and all the required information is collected

|  |  |  |  |
| --- | --- | --- | --- |
| TC\_FUNCT\_02 | | | |
| Student Registration test case | | | |
| Spyder | | | |
| REQ\_FUNCT\_02 | | | |
| Spyder should be installed on system. | | | |
| Step | Task | Expected Result | Actual Resullt |
| 1 | Click on the registered student button | Pass/Fail | Pass |
| 2 | Check that the information provided in valid | Pass/Fail | Pass |
| 3 | Verify that on clicking “camera” button camera turned ON | Pass/Fail | Pass |
| 4 | Make sure camera is capturing according to the requirement e.g. at different angles | Pass/Fail | Pass |
| 5 | Check that after pressing submit button data is successfully stored in the database. | Pass/Fail | Pass |

**Table 6.2: Student registration test case**

## Test Case for Home Page of Web Application:

|  |  |  |  |
| --- | --- | --- | --- |
| TC\_FUNCT\_03 | | | |
| Test Case for Home Page of Desktop Application | | | |
| Spyder | | | |
| REQ\_FUNCT\_03 | | | |
| Spyder should be installed on system. | | | |
| Step | Task | Expected Result | Actual Resullt |
| 1 | Make sure Home page of the application loaded successfully | Pass/Fail | Pass |
| 2 | Verify that the menus and function are shown properly on the home page of the application | Pass/Fail | Pass |
| 3 | Verify that on clicking “camera” button camera turned ON | Pass/Fail | Pass |
| 4 | Make sure that menu items on the page work properly | Pass/Fail | Pass |
| 5 | Check that application navigate to the respective page of the menu item. | Pass/Fail | Pass |

**Table 6.3: Home page test case**

## Student attendance test case

|  |  |  |  |
| --- | --- | --- | --- |
| TC\_FUNCT\_04 | | | |
| Student Attendance test case | | | |
| Spyder | | | |
| REQ\_FUNCT\_04 | | | |
| Spyder should be installed on system. | | | |
| Step | Task | Expected Result | Actual Resullt |
| 1 | Verify initiating attendance button works properly | Pass/Fail | Pass |
| 2 | Verify that camera is started and taking pictures | Pass/Fail | Pass |
| 3 | Make sure that after processing present students will be marked | Pass/Fail | Pass |
| 4 | Verify that faculty receives message which provide confirmation that attendance is marked. | Pass/Fail | Pass |
| 5 | Make sure that faculty logged out successfully | Pass/Fail | Pass |

**Table 6.4: Student attendance test case**

## Invigilation system test case

|  |  |  |  |
| --- | --- | --- | --- |
| TC\_FUNCT\_05 | | | |
| Invigilation system test case | | | |
| Spyder | | | |
| REQ\_FUNCT\_05 | | | |
| Spyder should be installed on system. | | | |
| Step | Task | Expected Result | Actual Resullt |
| 1 | Verify that “Start invigilation” button works properly. | Pass/Fail | Pass |
| 2 | Verify that camera is started and taking pictures | Pass/Fail | Pass |
| 3 | Make sure that an alert is generated when suspicious body movement is detected | Pass/Fail | Pass |
| 4 | Make sure that the student name and id is not wrong | Pass/Fail | Pass |
| 5 | Check that system saves the names and ids of the students that cause trouble automatically after exiting the module. | Pass/Fail | Pass |

**Table 6.5: Invigilation system test case**

## Usability Testing

Device users perform usability testing. In this type of research, we ask the system's user to run the programme and perform its functions so that we can determine how simple or difficult it is to use the system. Furthermore, we are aware that the device complies with the requirements.

## Usability Test case for login

|  |  |  |  |
| --- | --- | --- | --- |
| TC\_FUNCT\_06 | | | |
| Login Screen test case | | | |
| Spyder | | | |
| REQ\_FUNCT\_06 | | | |
| Spyder should be installed on system. | | | |
| Step | Task | Expected Result | Actual Resullt |
| 1 | Check that application is running perfectly. | Pass/Fail | Pass |
| 2 | Verify that the login button work flawlessly | Pass/Fail | Pass |
| 3 | Check user logged in successfully | Pass/Fail | Pass |

**Table 6.6: Login Screen test case**

## Usability test case for create, update and delete

|  |  |  |  |
| --- | --- | --- | --- |
| TC\_FUNCT\_07 | | | |
| Insert, update test case | | | |
| Spyder | | | |
| REQ\_FUNCT\_07 | | | |
| Spyder should be installed on system. | | | |
| Step | Task | Expected Result | Actual Resullt |
| 1 | Make sure that registration module is opening correctly. | Pass/Fail | Pass |
| 2 | Verify new student is registered in database successfully | Pass/Fail | Pass |
| 3 | Check update button is available on the registration first display. | Pass/Fail | Pass |
| 4 | Verify that student record changes in the database after the updating | Pass/Fail | Pass |
| 5 | Make sure “Delete Student” button is present on screen and on clicking it deletes the intended student record | Pass/Fail | Pass |

**Table 6.7: Insert, update test case**

## Exception handling testing

## We use exception handling in our system to deal with any issues that might arise, and we evaluate it by providing inputs that trigger the exception handle to ensure that it functions properly.

## Performance testing

## We conduct performance testing to see how well our device responds to the tasks we give it. We would be able to identify weak points when the device becomes inefficient by performing performance testing.

## Test case for system performance

|  |  |  |  |
| --- | --- | --- | --- |
| TC\_FUNCT\_08 | | | |
| System performance test case | | | |
| Spyder | | | |
| REQ\_FUNCT\_08 | | | |
| Spyder should be installed on system. | | | |
| Step | Task | Expected Result | Actual Resullt |
| 1 | Run application | Pass/Fail | Pass |
| 2 | Login to application | Pass/Fail | Pass |
| 3 | Click on different modules of application separately and checking how efficiently they are working | Pass/Fail | Pass |
| 4 | Initiate attendance module | Pass/Fail | Pass |
| 5 | Initiate invigilation module | Pass/Fail | Pass |
| 6 | Check create, update and delete button functionalities. | Pass/Fail | Pass |

**Table 6.8: System performance test case**

**Chapter 8**

**Conclusion**

## Conclusion

## Smart Attendance System is a desktop programme that was created to replace the conventional attendance and invigilation system, which has several drawbacks such as time waste, proxy attendance, and invigilation deception. To solve all bugs or challenges, this device employs the most up-to-date artificial intelligence techniques. Once installed, the device requires very little human interference to function. Furthermore, the user-friendly interface facilitates communication between the system and its stakeholders. During the creation of the system, we experiment with a variety of new technologies and methods to see how far we can push artificial intelligence. We are sure that with time this application will be more mature and get perfection and more accuracy. We are confident that as time passes, this application will become more refined and accurate.

## Future Enhancements

## Our system can currently register students, mark attendance through face detection and recognition, and invigilate by detecting head and gaze movement that is not permitted in the examination hall; however, in the current system, we ignore some special cases that may arise, such as human face changes from time to time, causing the camera to ignore these faces and mark absent; additionally, since we live in a Muslim society, girls wear niqab, we ignore some special cases that may arise, such as time to time human face changes, causing the camera to ignore these faces and mark. To address these concerns, we are considering expanding our project by incorporating another Artificial Intelligence technique to detect the student's retina. We will mark the student's attendance by identifying its face or retina, and if one of them matches, we will mark the attendance, increasing the accuracy of the attendance module. We have considered modifying some of the system's features to use it for protection in institutes.

# Reference

1. N.Sudhakar Reddy, M.V.Sumanth, S.Suresh Babu, "A Counterpart Approach to Attendance and Feedback System using Machine Learning Techniques",Journal of Emerging Technologies and Innovative Research (JETIR), Volume 5, Issue 12, Dec 2018.
2. 2. Dan Wang, Rong Fu, Zuying Luo, "Classroom Attendance Auto-management Based on Deep Learning",Advances in Social Science, Education and Humanities Research, volume 123,ICESAME 2017.
3. 3. Akshara Jadhav, Akshay Jadhav, Tushar Ladhe, Krishna Yeolekar, "Automated Attendance System Using Face Recognition", International Research Journal of Engineering and Technology (IRJET), Volume 4, Issue 1, Jan 2017.
4. 4. B Prabhavathi, V Tanuja, V Madhu Viswanatham and M Rajashekhara Babu, "A smart technique for attendance system to recognize faces through parallelism", IOP Conf. Series: Materials Science and Engineering 263, 2017
5. Michel Owayjan, Amer Dergham, Gerges Haber, Nidal Fakih, Ahmad Hamoush, Elie Abdo “Face Recognition Security System”
6. Akanksha Student, Jashanpreet Kaur Student, Harjeet Singh “Face detection and Recognition: A review” in: 6th International Conference on Advancements in Engineering & Technology (ICAET-2018), Feb. 23-24, 2018, Sangrur
7. Anil K Jain, Lin Hong, Sharath Pankanti, and Ruud Bolle, Biometric Identification. IEEE, 2004
8. N. Tom, Face Detection, Near Infinity - Podcasts, 2007.
9. T. Kanade, Computer recognition of human faces. Basel [etc.]: Birkhäuser, 1977.
10. .. A. L. Rekha and H. K. Chethan, “Automated Attendance System using face Recognition through Video Surveillance,” Int. J. Technol. Res. Eng., vol. 1, no. 11, pp. 1327–1330, 2014.
11. I. Kim, J. H. Shim, and J. Yang, “Face detection,” Face Detect. Proj. EE368 Stanf. Univ., vol. 28, 2003.
12. E. Shervin, “OpenCV Computer Vision,” 03-Oct-2010.
13. T. Matthew and A. Pentland, Eigenfaces for Recognition, vol. 3, Volume 3, Number 1 vols.

Vision and Modelling Group, The Media Laboratory, MIT: Journal of Cognitive Neuroscience, 1991.

1. Y.-Q. Wang, “An Analysis of the Viola-Jones Face Detection Algorithm,” Image Process.

Line, vol. 4, pp. 128–148, Jun. 2014.

1. Y. Freund, R. Schapire, and N. Abe, “A short introduction to boosting,” J.-Jpn. Soc. Artif.

Intell., vol. 14, no. 771–780, p. 1612, 1999.

1. P. Viola and M. J. Jones, “Robust real-time face detection,” Int. J. Comput. Vis., vol. 57, no. 2, pp. 137–154, 2004.
2. N. Mahvish, “Face Detection and Recognition,” Few Tutorials, 2014.
3. P. Viola and M. J. Jones, “Robust real-time face detection,” Int. J. Comput. Vis., vol. 57, no. 2, pp. 137–154, 2004
4. Arun Katara, Mr. Sudesh V. Kolhe, Mr. Amar P. Zilpe, Mr. Nikhil D. Bhele, Mr. Chetan

J. Bele. (2017). “Attendance System Using Face Recognition and Class Monitoring System”, International Journal on Recent and Innovation Trends in Computing and Communication, V5

1. Akshara Jadhav. (2017) .Automated Attendance System Using Face Recognition.

International Research Journal of Engineering and Technology.V4

1. Rahul V. Patil and S. B.Bangar. (2017). Video Surveillance Based Attendance system.

IJARCCE, 6(3), pp.708-713.

1. Varsha Gupta, Dipesh Sharma. (2014), “A Study of Various Face Detection Methods”, International Journal of Advanced Research in Computer and Communication Engineering), vol.3, no. 5.
2. S. Aanjanadevi, Dr. V. Palanisamy, R. Anandha Jothi. (2017). A Study on Secure Online

Voting System using Biometrics Face Detection and Recognition Algorithms. International Journal for Modern Trends in Science and Technology, V3

1. https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts- 17606c328184
2. Mustamin Anggo, La Arapu. (2018) Face Recognition Using Fisherface Method Iin 2nd International Conference on Statistics, Mathematics, Teaching, and Research
3. Sharma, N., & Kaur, R. (2016). Review of Face Recognition Techniques, 6(7), 29–37
4. Eng. Mohannad M. Ayash “Research Methodologies in Computer Science and Information Systems”

# Glossary

**Keras:** Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load.

**DFD**: Data Flow Diagram Diagram

**RFID:** Radio-frequency identification uses electromagnetic fields to automatically identify and track tags attached to objects. An RFID system consists of a tiny radio transponder; a radio receiver and transmitter.

**Spyder**: Spyder is an open source cross-platform integrated development environment for scientific programming in the Python language.

**SDLC:** Software development life cycle.

**Python:** Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace.