STUDENT ATTENDANCE PROGRAM WITH FACE RECOGNITION

by
Alp Şahin

Engineering Project Report

Yeditepe University
Faculty of Engineering
Department of Computer Engineering
2021

STUDENT ATTENDANCE PROGRAM WITH FACE RECOGNITION

APPROVED BY:

Assist. Prof. Dr. Dionysis Goularas (Supervisor)	. . . . .
Prof. Dr. Emin Erkan Korkmaz	. . . . .
Assist. Prof. Dr. Mustafa B. Mutluoğlu	. . . . .

DATE OF APPROVAL: .../.../2021

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Dionysis Goularas for his guidance and support throughout my project.

Also I would like to thank my parents for their support and encouragement throughout my education up to the present.

ABSTRACT

STUDENT ATTENDANCE PROGRAM WITH FACE RECOGNITION

Face recognition is one of the most effective image processing applications and it is in an important place among the technical areas. Face recognition is a problem for authentication purposes especially for students. Face recognition attendance system is a procedure of recognizing students by using face biostatistics based on high definition monitoring and other computer technologies. The development of this system is aimed to achieve taking place of the traditional system of taking attendance by calling names or signing a paper. These techniques are boring, time-consuming, and not reliable due to proxy signs. Therefore, this project focuses on all those problems, by explaining several existing approaches and demonstrating the implementation of them.

ÖZET

YÜZ TANIMALI ÖĞRENCİ YOKLAMA PROGRAMI

Yüz tanıma en etkili görüntü işleme uygulamalarından birisidir ve teknik alanlar arasında önemli bir yer tutmaktadır. Yüz tanıma, özellikle öğrenciler için kimlik doğrulama amaçlı bir sorundur. Yüz tanımalı yoklama sistemi, yüksek çözünürlüklü izleme ve diğer bilgisayar teknolojilerine dayalı yüz biyoistatistiklerini kullanarak öğrencileri tanıma prosedürüdür. Bu sistemin geliştirilmesinde ki amaç, isim okuyarak veya bir kağıt imzalayarak gerçekleştirilen geleneksel yoklama alma sisteminin yerini almasını sağlamaktır. Bu teknikler sıkıcı, zaman alıcı ve sahte imzalar sebebiyle güvenilir değildir. Bu nedenle, bu proje mevcut birkaç yaklaşımı açıklayarak ve bunların uygulanmasını göstererek tüm bu sorunlara odaklanmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

MAS	Manual Attendance System
AAS	Automated Attendance System
RFID	Radio Frequency Identification
DL	Deep Learning
ML	Machine Learning
NN	Neural Network
OpenCV	Open Computer Vision Library
GUI	Graphical User Interface
SQL	Structured Query Language
HCC	Haar Cascades Classifiers
LBPH	Local Binary Pattern Histogram
VJ	Viola Jones
CNN	Convolutional Neural Network
HOG	Histogram of Oriented Gradients
DB	Database
E-R	Entity–Relationship

1. INTRODUCTION

The technology aims to offer great information about the technical innovations these days. Deep learning is one of the most attractive branches that authorizes the machine to train itself. A lot of datasets used for training as an input and provides a suitable output thanks to learning algorithms. New and more efficient algorithms are added to these algorithms every day. In this way, deep learning technologies started to take place in every corner of our lives.

1.1. Problem Definition

Every course requires a strong and stable system to record the attendance of students. However, attendance is a problem for educational organizations and it affects both students as well as the instructors nowadays. Attendance participation frequency is showed below in figure 1.1 [1]. With the development of deep learning, the system automatically determines the attendance which is in a specific time period of students. In general, the student attendance system divided into two parts:

- Manual Attendance System (MAS)
- Automated Attendance System (AAS)

All instructors have different methods to maintain the attendance system. There are two the most used methods for manual attendance systems. First is giving a sheet of paper for signing student's names during lecture hours. However, it is not completely safe because of forged signatures by unauthorized students. The second is manually calling the names of the students. These processes might take 10-15 minutes of the entire session. In this way, these traditional techniques of taking attendance are not too efficient. Therefore, many institutes started performing the new techniques for automated attendance systems. Such as Radio Frequency Identification (RFID) card reader, biometric fingerprint recognition, iris recognition, and so on [2]. However, also these new techniques are time consuming because they are queue based systems and are intrusive in nature. For the RFID card system, there are possibilities of card loss or card reading by non-official persons.

Face recognition is the smartest method among the other attendance methods. Especially, on the topics of accuracy, reliability, and speed. Thanks to these features, it reduces the time and the chance of proxy attendance. Face recognition works passively to identify and record

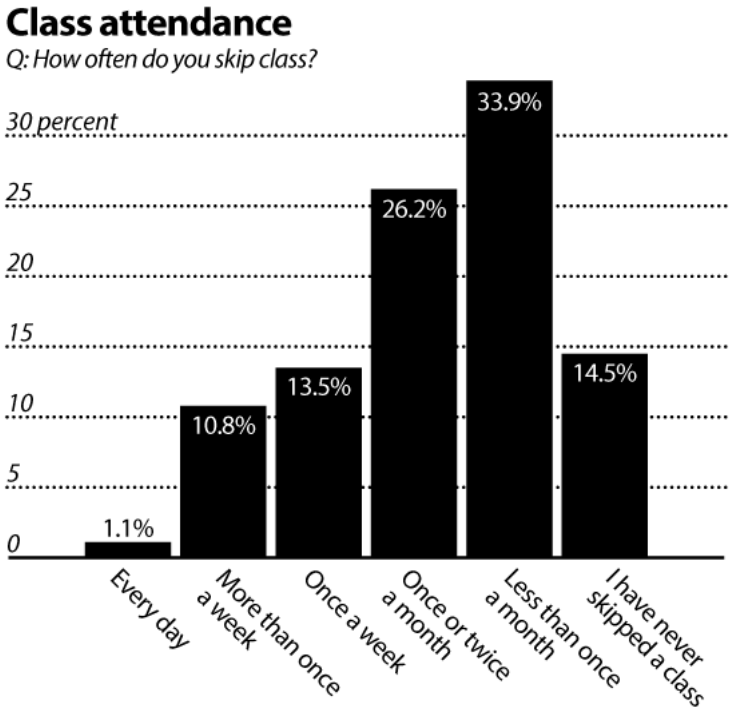


Figure 1.1. Class Attendance Participation Rate

the students. Thus, the student does not do anything after those processes, the system works automatically. Consequently, this project aims to solve problems in the other attendance systems and aims to make the face based attendance systems permanent in our lives.

Face recognition systems consist of two steps, verification and face identification. The first part consists of the detection of faces and the second part consists of the identification of those detected face images with the existing database. There are a lot of special face detection and face recognition techniques. Usually, appearance based systems preferred, they cover the whole face of a person. Other techniques work feature based, they cover the geometric features on the face such as eyes, nose, eyebrows, and cheeks. Recently, mask feature based face recognition often used due to COVID-19.

1.2. Requirements

In this subsection, explained **what** and **how** the project should do. Firstly, the system requires a computer camera that captures the images of the students and gives them to the image trainer method. After training, the face recognition module works, and then the attendance is recorded on the database server. The algorithm compares all training images and detected

images one by one with the student database. Three databases required for this task: Student database, lecture database, and attendance database. In order to ensure correct working conditions, the recognition should not be done at far distances, and in dark or bright lights. Also, the number of images in the dataset and the quality of the used camera are influencing factors. In the test and results part, the effects of these conditions are analyzed and result values shared. After these descriptions, the technological requirements for this project are identified.

1.2.1. Hardware Requirements

- A computer has to be installed in the classroom where the program is to be located.
- The computer should include a high quality camera.
- A powerful processor needs to be used due to the need for a high CPU.
- A memory that has a big capacity is required for software, images, and databases.

1.2.2. Software Requirements

- Windows 10: Windows 10 operating system is required for installation of all necessary up-to-date software.
- OpenCV: In this project, OpenCV via Anaconda is used for installing machine learning software libraries.
- Keras: Keras is a NN (Neural Network) library, it is required for data augmentation.
- Python: Python should be preferred for this project. Because it is the most prevalent and well-supported programming language for machine learning.
- NumPy: NumPy open source library is used in order to enable numerical computing as the arrays and the linear algebra functions for the face recognition with Python.
- Tkinter: Tkinter is a standard library for Python. It is required to build GUI (Graphical User Interface) on the project.
- Anaconda: Anaconda is the most prevalent Python distribution platform. It provides an environment for this project.
- SQLiteStudio: SQLiteStudio manages the database processes. It is used to accomplish diverse queries, insert data, and read data from the database.

1.3. Aims and Objectives

The objective [3] of this final project is to develop a student attendance system with face recognition. The expected achievements to reach the goals are:

- Detecting the face segment from the video frame.
- Extracting the useful features from the face detected.
- Classifying the features to recognize the face detected.
- Saving the attendance of the identified student.

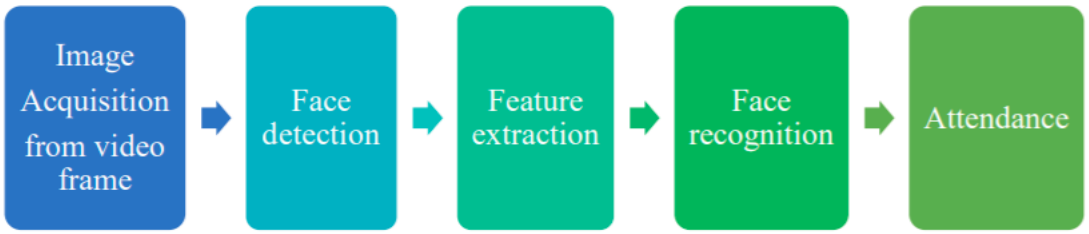


Figure 1.2. Block Diagram of the General Framework

2. RELATED WORKS

One of the most important secrets of success is to examine the projects that have been done before I start my own project and to get ideas from them. In this chapter, three related works that were developed to take attendance with face recognition are examined together with the plus points and minus points of each work.

2.1. Auto Attendance Using Face Recognition

One of the auto based attendance applications is "Auto Attendance Using Face Recognition" [4] which was developed by "Mahvish Nasir". In this project, there are two users accountable for the system. One of the users is the instructor who takes the attendance regularly and the other user is the admin who manages the faces of the students in the student database. Admin uses a camera to capture, collect and save the images to the database. When the collection and saving of the transaction finish, the trainingSet manager who called admin starts to eject faces from the image. Eventually, the admin is going to add the extracted image of students to the class or the trainingSet. Attendance is marked down by recognizing the faces owing to a computer camera and automatically recorded in the attendance sheet. The instructor enters the course name and the class name to begin the attending process. Screenshots of the Auto Attendance Program as shown in figure 2.1 and figure 2.2 [4].

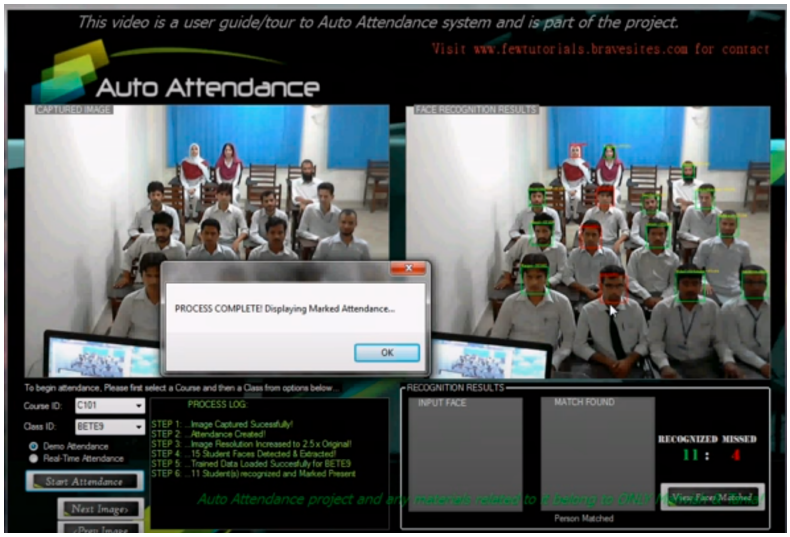


Figure 2.1. Auto Attendance System's Interface

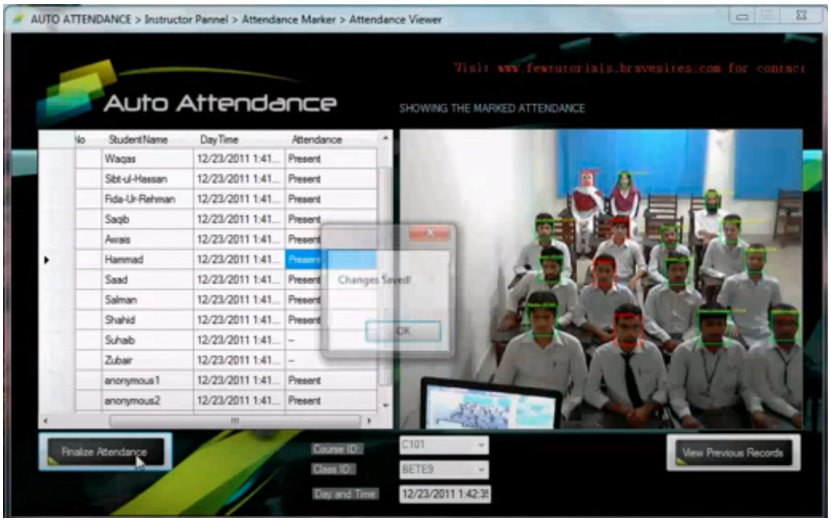


Figure 2.2. Displaying the Attendance List

Advantages:

- The system stores the faces of students in the database, detecting and marking automatically come true.
- Ease of use and system allows the only authorized access.
- Ensure the methods to increase the number of extracted faces from an image.

Disadvantages:

- The accuracy of the system is not 100%.
- Face detection and loading the training data processes take a small time.
- The instructor and admin still do some parts manually.

2.2. Attendance System on Face Detection

Another application is “Attendance System on Face Detection” which was developed by “NevonProjects” [4]. In this project, the system is designed for employing a simple and secure technique of creating an attendance list. Firstly, the application captures a photo of all the authorized people. Then, the program stores the photos by mapping them into a face

check structure. The application will recognize the registered employee and record her/him into the database. Screenshots of the program are shown in figure 2.3 and figure 2.4 [4].

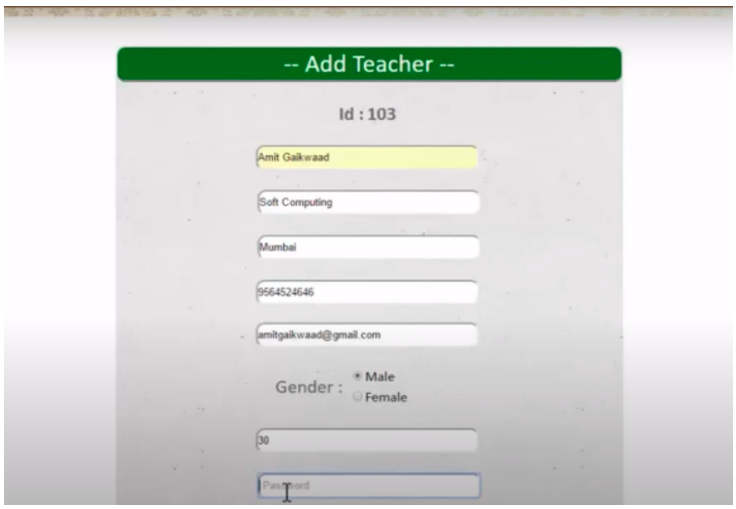


Figure 2.3. Adding User Information

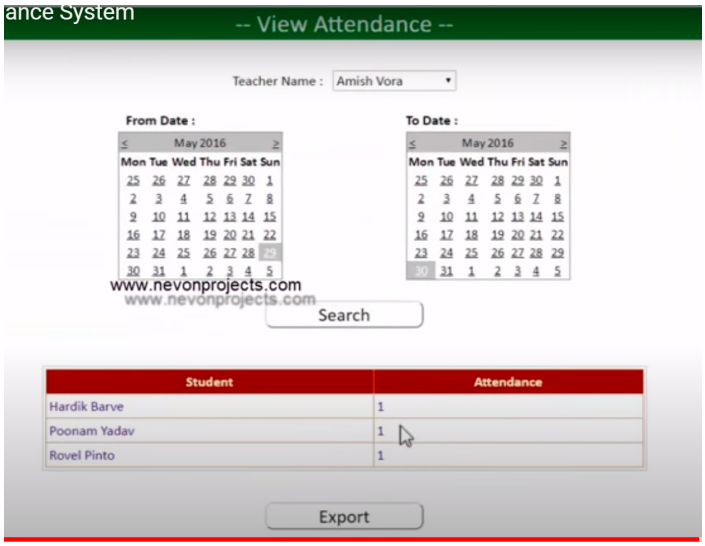


Figure 2.4. Viewing the Attendance List

Advantages:

- The program stores the detected faces of employees and automatically saves them into the database.
- The system manipulates and recognizes the faces in a real time video camera.

- It saves time and effort.
- Ease of use.
- Multiple face detection.
- Can be used in different places.

Disadvantages:

- The application may not recognize the faces properly at a distance and in the dark.
- The accuracy of the system is not 100%.
- It may not capture again the live video for missed faces.

2.3. Churchix Face Recognition Software

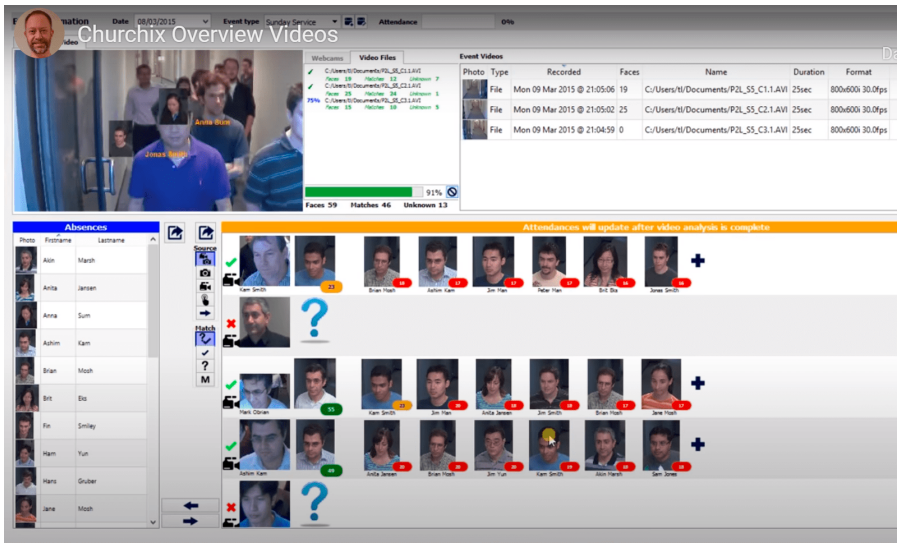


Figure 2.5. Face Recognition and Creating the Attendance List

[5] The last project of this chapter is Churchix. A screenshot of the Churchix is shown in above figure 2.5. It introduces a useful face recognition program, which takes the person's face owing to a picture or a video. After that, the system compares it with the contents of the database then the system identifies the person. Furthermore, Churchix has an application for Android devices. In order to use it, people focus the phone camera on a person who wanted to recognize. In addition to this, The Churchix Application allows enrolling people from the

photo album. For these reasons, the program is becoming more widespread every day.

Advantages:

- The program takes the faces with videos and photos.
- High quality photos.
- High accuracy.

Disadvantages:

- Some people see this program as a threat to privacy.
- Deficiency to capture all the faces in the video.
- Restart the video several times to capture the all faces.

3. ANALYSIS AND DESIGN

The analysis is the process of breaking a complex subject or material into smaller parts in order to get a better understanding of it. After the system analysis, the design of the system should be designed. These processes should be done before implementation. If the analysis and design of the system are done poorly, problems will occur in the implementation of the system.

3.1. Analysis of the System Requirements

In order to use the system very well and efficiently, the system has some requirements for users and also for hardware. With those requirements, the system becomes more user-satisfied, easy to use, and useful. There is one role in this system which is the user.

3.1.1. User Requirements

In this project, users are both students and instructors. In order to work the attendance system, students should implement some stages. Firstly, the button which is named ”Add a Student (Create Face Data)” should be clicked. After this, students should be able to register to the system by entering their name, id, age, and gender. Then, the camera appears on the screen and starts to capture the photos of students’ faces. If all students are registered in the system, should be clicked to the button which is named ”Train Dataset” for training the images. Then instructor clicks the ”Start Lecture (Face Recognizer)” button just before the lecture. This event provides the recording attendance. However, two questions should be selected by the instructor. These are the course name and course date. The instructor selects these from the combo box and calendar. When the instructor wants to see the attendance list, they should click on the ”Attendance List” button. Also here instructor selects the course name from the combo box and selects the course date that want to see from the calendar. All information is displayed on the screen as student name, student id, lecture name, and lecture date.

3.1.2. Hardware and Software Requirements

In this project, various types of deep learning and image processing are used. Therefore, the system needs some required algorithms and environments. In addition, a database is necessary to keep the students’ personal information, course information, and students’

attendance information. Lastly, the system requires a computer and a camera.

3.2. Overview of the Architecture

In order to solve the problems which were mentioned before, the attendance system consists of four parts. They are camera face capture, student image database, face recognition, and attendance recording. The system architecture is shown in figure 3.1 [6].

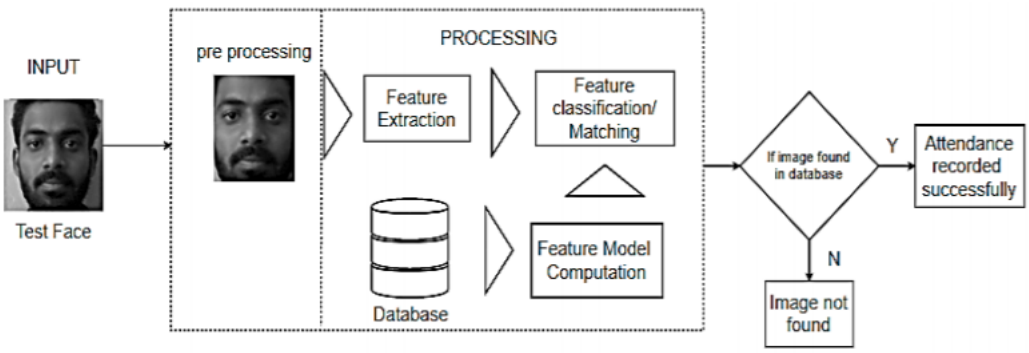


Figure 3.1. Proposed System Architecture Diagram

Initially, an input image is captured by the system. Before the processing, made preprocessing on the image. In the processing part, feature extraction and feature model computation techniques are used. However, the last action is determined according to the condition of the database. There are two possibilities. First is that if the image is found in the database, attendance is recorded successfully. Second, if the images are not found in the database, attendance is not recorded.

3.3. Modeling of the System

Figure 3.2 [4] shows a use case diagram of the student attendance with face recognition. In the system, the scenario begins with the running of the application. All of the users have the authority to run the system. With this, the program starts to activate the camera for student attendance. The camera has two processes, recognizing faces and detecting faces. Admin has authority for running applications and managing students. Lecturer except those two can view the student attendance. However, students only run applications and see their own attendance.

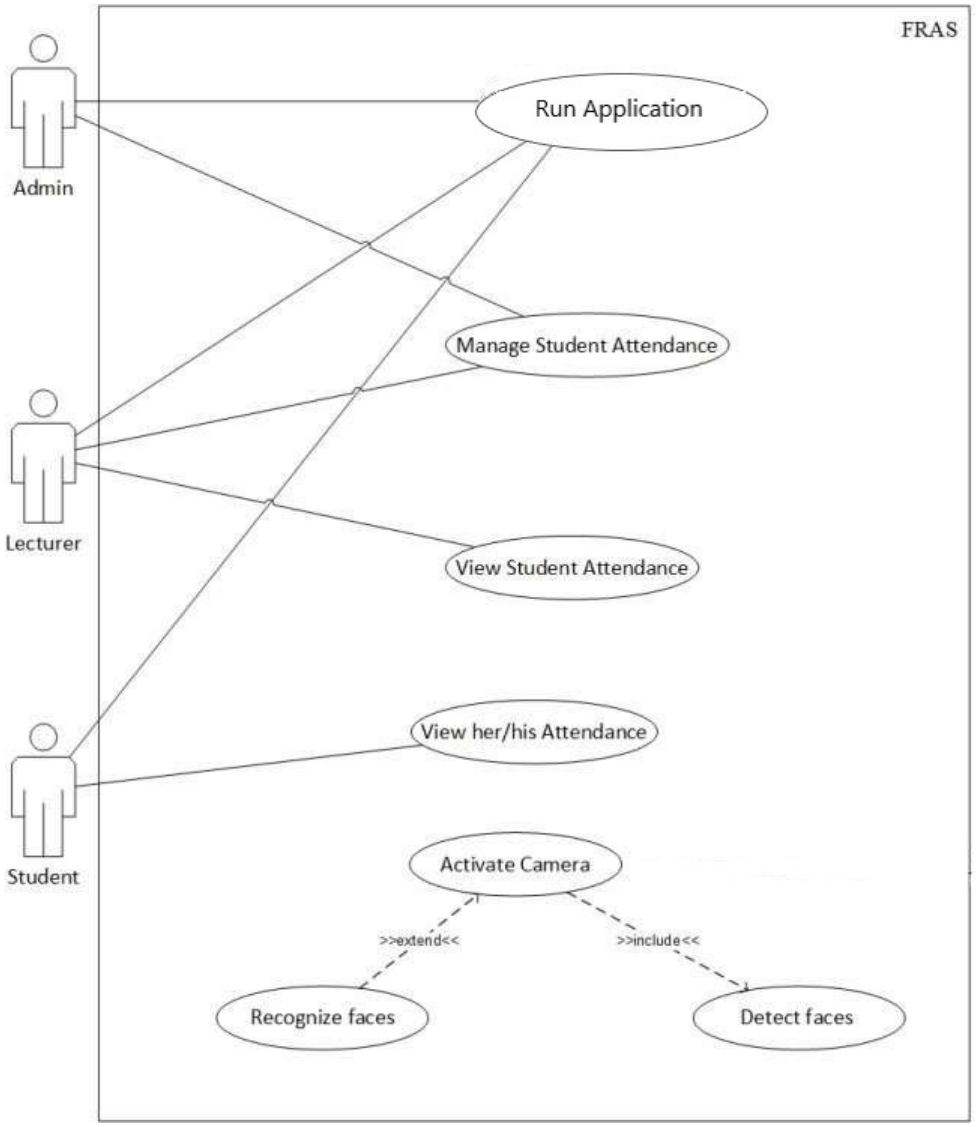


Figure 3.2. Access Permissions of Users

The sequence diagram of the system is shown in figure 3.3 [7]. Firstly, the faculty starts the capturing module for students. Meanwhile, other lifelines do not start until the end of the session record. After the captured students, those images were sent to the processing module in order to compare. The omparison module acts as temp storage. After the process, compared images sent to the database. If the database has images of those students, attendance is filled up.

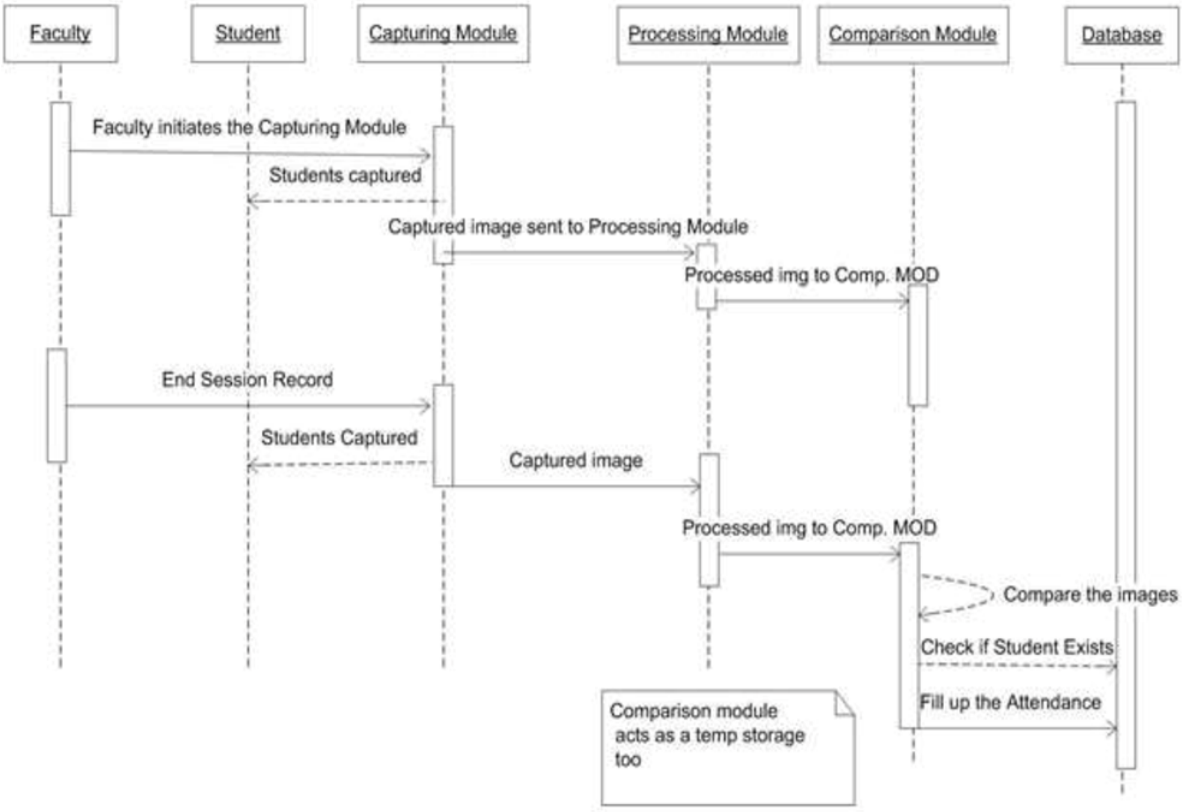


Figure 3.3. Object Interactions Arranged in Time Sequence

Activity diagram is essentially an advanced version of the flow chart that modeling the flow from one activity to another activity. Therefore, the system of this project is explained with the activity diagram in figure 3.4 [8]. The system is separated into three working modules. They are face representation, feature extraction, and classification [9]. The first process is modeling the faces of students. The image acquired is transformed to match the positions of images already present. In the feature extraction, the features of the face are mapped as histograms. Lastly, the system compares the images on the camera thanks to the database.

E-R (Entity - Relationship) model is another important behavioral diagram among the UML diagrams to describe aspects of the system. In figure 3.5 [10], the E-R diagram of the system is shown. There are three types of relations as (1-1), (1-M), and (M-M). In the diagram, classroom entities have only one relationship which is named ”belongs to” one to many relationships with student entities. However, the student has relationships with all entities such as capture image, train image, recognize image and attendance sheet. System and recognize image entities have one relationship as the classroom, but others have many relationships with the student. Lastly, the student has an attribute which is named id.

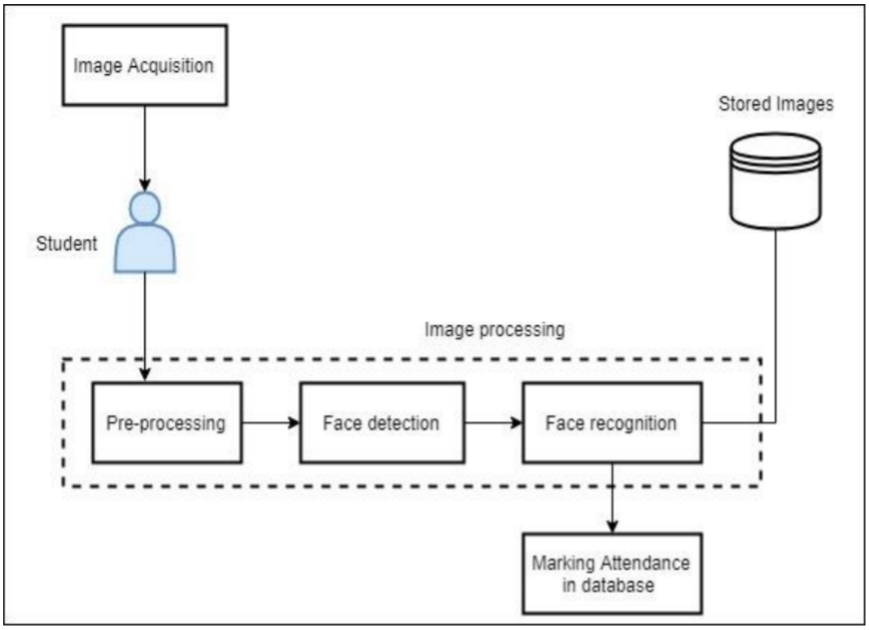


Figure 3.4. System's Behaviour

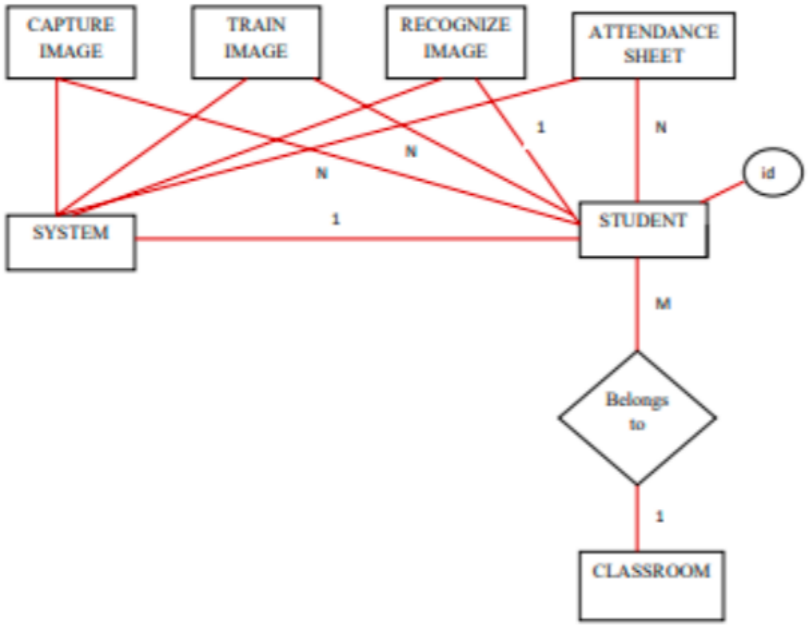


Figure 3.5. Relationships of Entity Sets

3.4. Design Details

For this display, the flowchart diagram is used. Because flowchart transforms into clear, easy-to-understand diagrams that allow developing of complex processes. In the beginning, the user runs the program and the camera is turned on. The program tries to scanning the face and creates an input dataset. This dataset is recorded to the database for comparison. If face recognition is successful, the program goes to the attendance process for the second comparison operation. However, if face recognition is not achieved, the system goes to the first step again like a loop. After a successful matching, the system records the attendance of students. Lastly, the system is closed. Stages of the design the student attendance with face recognition is showed in figure 3.6 [11].

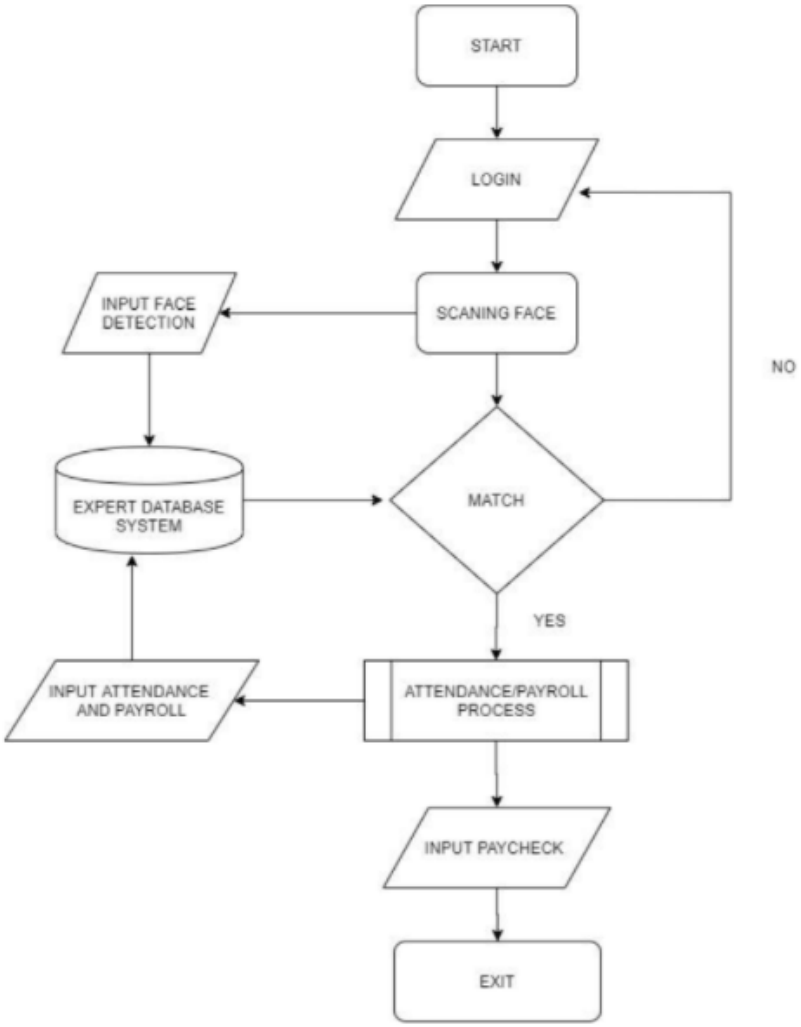


Figure 3.6. Flow of the Model

3.5. The Overview of the Algorithm

This section, described what types of algorithms are used in this project. Some algorithms are used for recognizing faces. For example Haar Cascade Classifiers (HCC), Viola-Jones, LBPH (Local Binary Patterns Histogram), and HOG (Histogram of Oriented Gradients).

3.5.1. Viola Jones and Haar-Feature Algorithm

Viola-Jones *face detection* algorithm is named after two scientists who developed the method in 2001, Paul Viola and Michael Jones. Name of their article, "Rapid Object Detection using a Boosted Cascade of Simple Features" [12]. Viola-Jones uses Haar-like features to detect faces in this algorithm. A Haar-Feature algorithm is a machine learning-based approach where a cascade function is trained from the numerous positive and negative images similar to other machine learning algorithms. A Haar-Feature is similar to the kernel in CNN (Convolutional Neural Networks), but there is a difference. In a CNN, the values of the kernel are defined by training, but a Haar-Feature is manually defined. It is afterward used to detect objects in other images. The Haar-Feature algorithm consists of four parts: Haar feature selection, creating integral images, AdaBoost training, and cascading classifiers. The first stage is to collect the Haar features. A Haar feature is actually calculations that are performed on contiguous rectangular regions at a certain location. The calculation includes the summing of the pixel intensities and calculating the differences between the sums. In figure 3.7 [13] is shown an example.

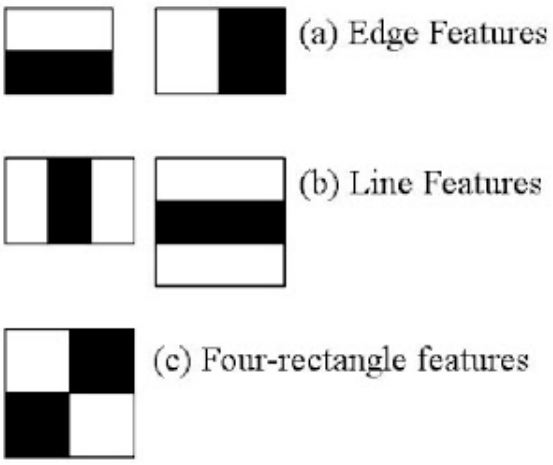


Figure 3.7. Types of Haar Features

The second stage is creating integral images. Instead of calculating at every pixel, it creates sub-rectangles and array references for each of those sub-rectangles. These are then used to compute the Haar features. It is showed in figure 3.8 [13].

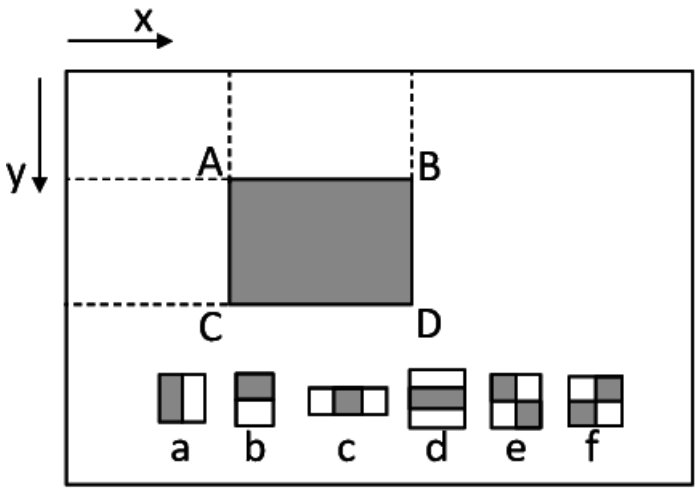


Figure 3.8. Illustration For How an Integral Image Works

After those steps, Adaboost comes. Adaboost actually prefers the best features and trains the classifiers to use them. It uses an integration of "weak classifiers" to create a "strong classifier" in order to detect objects. As shown in figure 3.9 [13].

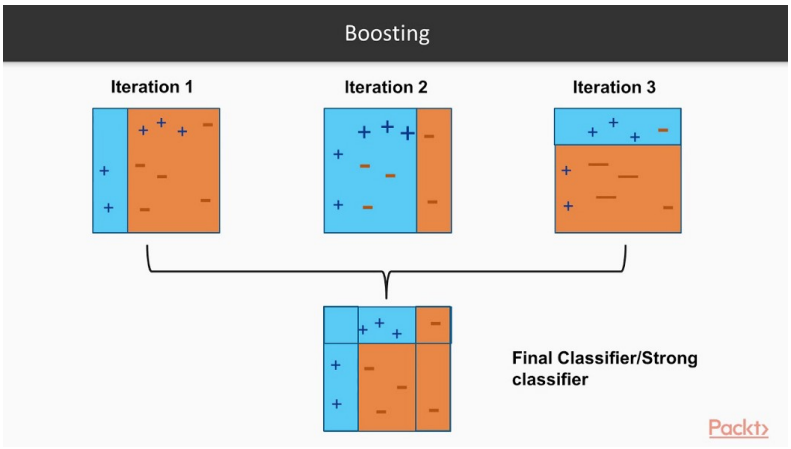


Figure 3.9. Adaboost Training the Classifiers

In the last step and the last figure 3.10 [13], the cascade classifier consist of a group of stages, where each stage is a collection of weak learners. Weak learners are trained thanks to boosting, which allows for a highly accurate classifier.

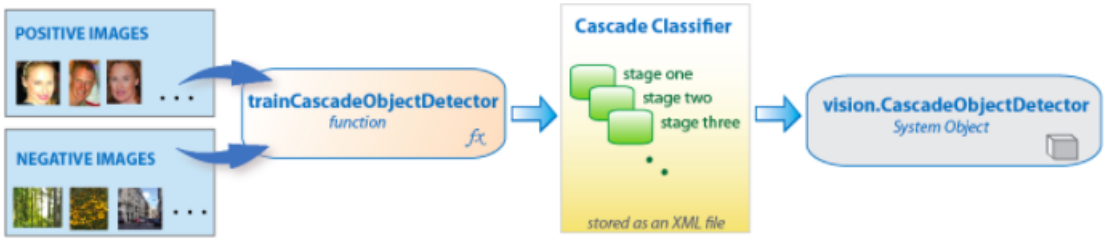


Figure 3.10. A Flowchart of Cascade Classifiers

3.5.2. LBPH and HOG Algorithm

HOG (Histograms of Oriented Gradients) is an efficient descriptor for object recognition and detection. These descriptors are very successful to detect faces with occlusions, pose, and illumination changes. LBP is a simple and very efficient texture face recognition operator which labels the pixels of an image by thresholding the neighborhood of each pixel and processes the result as a binary number [14]. It was first put forward in 1994 and has been analyzed to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptors, it increases the detection performance highly on some datasets. The LBP, together with the cascaded classifiers. It was used first in object recognition, where it was employed for face recognition. LBP is used for face recognition unlike, Viola-Jones. If we explain the difference between LBP and HOG, LBP uses all 8 directions for each pixel, but HOG only uses 1 direction for each pixel. HOG is accomplished at capturing edges and corners in the images. On the other hand, LBP accomplished at the local patterns. Actually, HOG and LBP can be assembled together in face recognition. Applying LBP is showed in figure 3.11 [14].

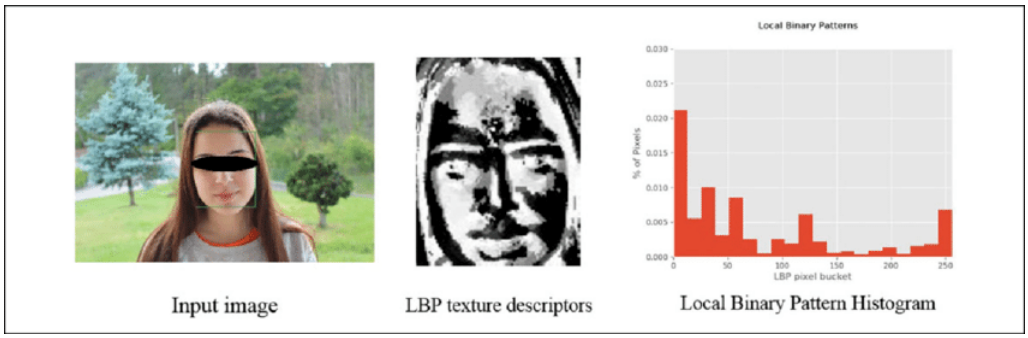
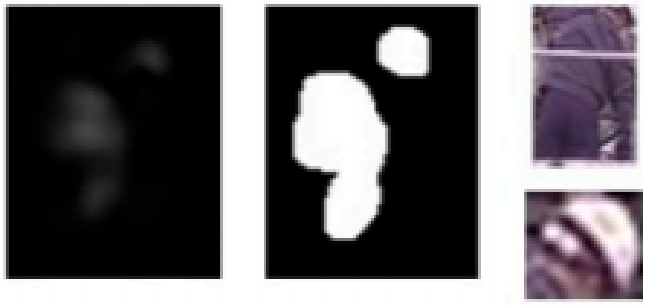


Figure 3.11. Applying LBP Operation

(a) First Background and Current Frame Images

(b) Gray Scale and Gaussian Blur Images



(c) Frame Subtraction, Threshold Image, and Result Image

Figure 3.12. Applying Haar Cascades on the Images

Figure 3.12 [15] above, shows the process of applying haar cascades on the images. The background image and the current frame image are shown in figure 3.12(a). Applying the grayscale and the gaussian blur on the last image is shown respectively in figure 3.12(b). Lastly, the frame subtraction using cv2.absdiff image, the threshold image, and the result image are showed in figure 3.12(c).

In the chart below [16], time consumption between Haar-Cascade and HOG algorithm is displayed. For measuring these values, eight different videos were used. According to the chart, the HOG algorithm consumes generally 1 second more than the Haar-Cascade algorithm. In this case, Haar-Cascade works faster than HOG.

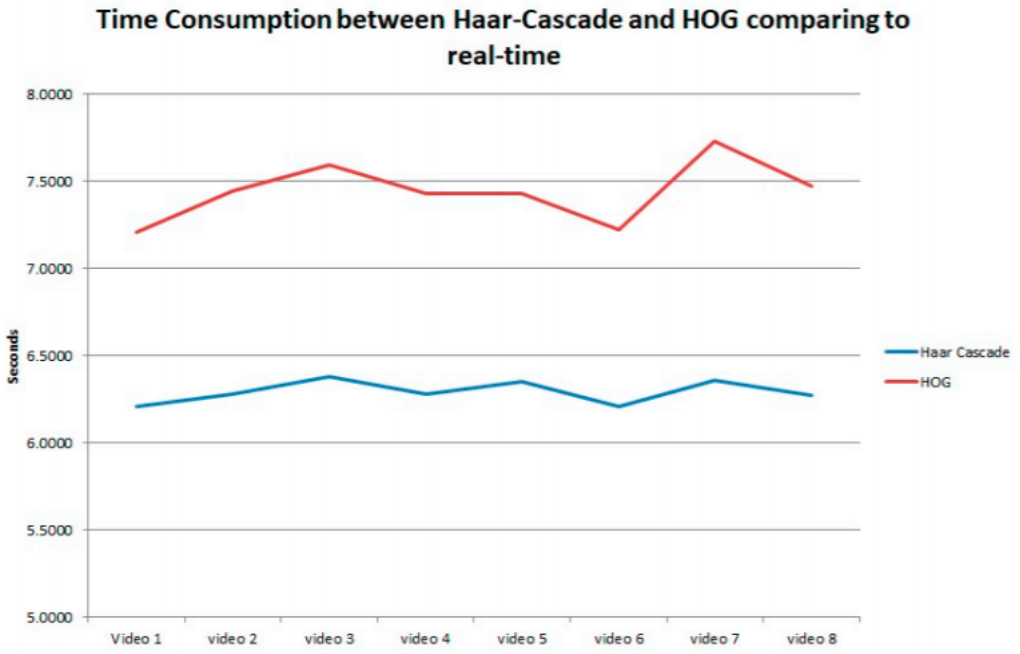


Figure 3.13. Comparing Haar-Cascade and HOG to Speed

3.6. The Algorithm of the System

The last step before implementation is to write the system's algorithm with pseudocode. In this way, algorithms and their operations are examined more detail. Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program. Pseudocode summarizes a program's flow but excludes underlying details. System designers write pseudocode to ensure that programmers understand a software project's requirements and align code accordingly. Pseudocode is not an actual programming language. So it cannot be compiled into an executable program. It uses short terms or simple English language syntaxes to write code for programs before it is actually converted into a specific programming language. This is done to identify top-level flow errors and understand the programming data flows that the final program is going to use. This definitely helps save time during actual programming as conceptual errors have been already corrected. In figure 3.14(a) [15][17], defined the pseudocode of Haar-cascade and Viola-Jones algorithm. The pseudocode of the HOG and the LBP algorithm is shown in figure 3.14(b) [18][19].

- Pick f (maximum acceptable false positive rate per layer) and d (minimum acceptable detection rate per layer)
- Lets F_{target} is target overall false positive rate
- Lets P is a set of positive examples
- Lets N is a set of negative examples
- Lets $F_0 = 1$, $D_0=1$, and i=0 (F_0: overall false positive rate at layer 0, D_0: acceptable detection rate at layer 0, and i: is the current layer)
- While $F_i > F_{target}$ (F_i: overall false positive rate at layer i):
  - i++ (layer increasing by 1)
  - $n_i=0$; $F_i = F_{i-1}$ (n_i: negative example i):
  - While $F_i > f*F_{i-1}$:
    - n_i ++ (check a next negative example)
    - Use P and N to train with AdaBoost to make a xml (classifier)
    - Check the result of new classifier for F_i and D_0
    - Decrease threshold for new classifier to adjust detection rate r >= $d*F_{i-1}$
  - N = empty
  - If $F_i > F_{target}$, use the current classifier and false detection to set N

Algorithm: Viola-Jones Face Detection Algorithm

```
Input: original test image
Output: image with face indicators as rectangles
for i ← 1 to num of scales in pyramid of images do
    Downsample image to create image_i
    Compute integral image, image_ii
    for j ← 1 to num of shift steps of sub-window do
        for k ← 1 to num of stages in cascade classifier do
            for l ← 1 to num of filters of stage k do
                Filter detection sub-window
                Accumulate filter outputs
            end for
            if accumulation fails per-stage threshold then
                Reject sub-window as face
                Break this k for loop
            end if
        end for
        if sub-window passed all per-stage checks then
            Accept this sub-window as a face
        end if
    end for
end for
```

(a) Haar-like Feature and Viola-Jones Algorithms

```
1: sds_alloc (RGB, Gray, Grad_x, Grad_y, Mag, Orient, Hist, binHist)
2: RGB2Gray(RGB, Gray)
3: Gradient(Gray, Grad_x, Grad_y)
4: Magnitude(Grad_x, Grad_y, Mag)
5: Orientation(Grad_x, Grad_y, Orient)
6: Histogram(Orient, Mag, Hist)
7: Normalization(Hist, normHist)
8: sds_free(RGB, Gray, Grad_x, Grad_y, Mag, Orient, Hist, normHist)
```

```
00: Load XI
01: for i=1 to W-2 do
02:     for j=1 to H-2 do
03:         block = XI(i: i + 2, j: j + 2); // 3 x 3 sized block division.
04:         Use Eqs. 1-2 to calculate values.
05:     end for j
06: end for i
07: Extract histogram of the value. // In here, the LBP image is coded as 8-bit. Therefore, the size of the histogram is calculated as 2^8 = 256. This histogram is utilized as feature vector (feat)
```

(b) HOG and LBP algorithms

Figure 3.14. Descriptions of the Steps in the Algorithms

In the Haar-like algorithm, some operations are made for face detection. These are Haar feature selection, creating some integral images, AdaBoost training, and cascading classifiers. Meanwhile, rectangular feature and cascading AdaBoost are operations in the Viola-Jones algorithm.

Preprocessing, calculating the gradient images and histogram of gradients, block normalization, and feature vector are operated in the HOG algorithm. Lastly, creating datasets, face acquisition, feature extraction, and classification operations are made in the LBP algorithm for face recognition.

4. IMPLEMENTATION

After the analysis and design part of a project's life cycle is completed, it moves on to the implementation part. At this stage, analyzes and calculations made on paper are now put into practice in order to give concrete shape to the project.

4.1. Programming Languages and Libraries

It's time for the implementation of the project. Before starting the implementation, language, tools, and libraries are defined [20].

4.1.1. Python

Python is an interpreted, object-oriented, and high-level programming language with dynamic semantics. It is high-level built-in data structures, combined with dynamic typing and dynamic binding. These features make Python very engaging for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python is a very simple programming language due to its easy syntax structure and being readability. Thanks to those features, the cost of program maintenance highly decreases. Python supports a lot of modules and packages, which encourages program modularity and code reuse. The Python interpreter and the wide standard library enable deep learning and image processing especially. Lastly, the Python programming language is available in the source so that can be used freely [21].

4.1.2. OpenCV

OpenCV (Open Source Computer Vision Library) [22] is a library of programming functions at real-time computer vision. It was developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source Apache 2 License. OpenCV has more than 2500 optimized algorithms. Those algorithms are used for especially detecting and recognizing faces and classifying human actions in videos.

4.1.3. Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit and is Python's important GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. Tkinter is a combining of the Tk interface and it was developed by Fredrik Lundh [23]. Tkinter library is available in the source so that can be used freely.

4.2. Introduction to Implementation

First of all, the system starts by getting camera permissions by going to the create_dataset and Detector files. Secondly, it is necessary to design how the application will look. "app-gui.py" file opened in order to design GUI of application. For running of the program and printing the error or warning messages from all classes, Anaconda Prompt is needed. Lastly, necessary libraries are imported. Required procedures and required libraries are listed below.

- cv2: Library is used to reach OpenCV features.
- tk: Library is used to reach Tkinter features.
- numpy: Library is used to reach numpy features.
- keras: Library is used to reach keras features.
- recognizer: Library is used to recognize the face.
- VideoCapture(): This method is used for running the camera and send the frames to the dataset.
- cam.set(): Method is used for setting the camera height and width.
- haarcascade_frontalface_default.xml: This file is a cascade type and used for face detection.
- train(): Method is used for training data.
- LBPHFaceRecognizer_create(): Method is used for creating a connection with trained images.
- trainner.yml: Trained images kept in this file with a special type of naming.
- ImageDataGenerator(): Is used for data augmentation.

- imwrite(): This method is used for write something into a file.
- detectMultiScale():Method is used for face detection.
- predict(): Method is used to determine accuracy.
- getProfile(): This method is used for getting the given attributes of the various faces.
- sqlite3.connect(): Method is used for connecting to the database.
- INSERT: Query is used for inserting data into the database.
- SELECT: Query is used for selecting data from the database.
- __init__: It is a constructor and it allows to initialize the attributes of a class.
- Label(): Method creates a label for texts.
- grid(): Method is used to set the location of labels, buttons, or other items.

4.3. Screenshots of the Application

Views of the main menu of the application are shown in figure 4.1. There are three buttons. For adding a new student to the lecture, the ”Add a Student” button is used. Before just starting the lecture, the ”Start Lecture” button is used. Lastly, the ”Attendance List” button is for seeing the attendance table.

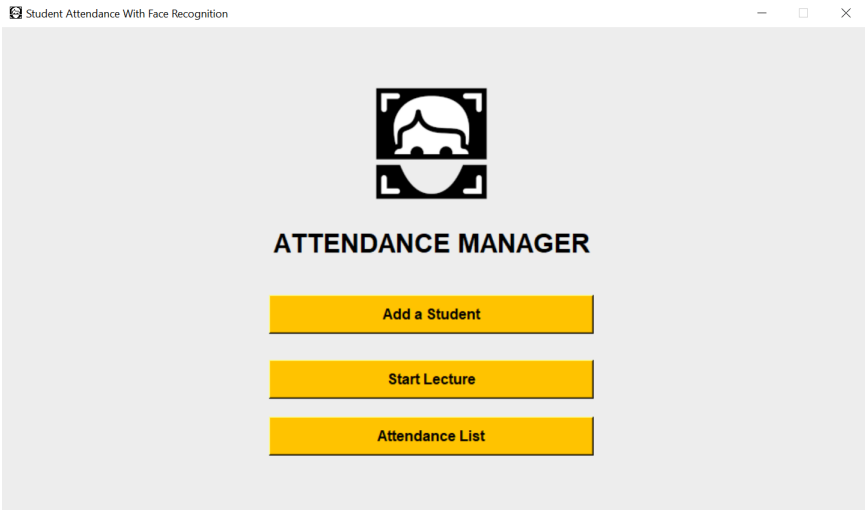


Figure 4.1. Main Menu

Figure 4.2 shows the result of the ”Add a Student” button. Some personal information such as student’s name, id, age, and gender filled by the student. After pressed ”Add User” button, the student was added to the database. The ”Train Dataset” button is pressed in order to train face images. When the system captures the images of the face is shown in figure 4.3.

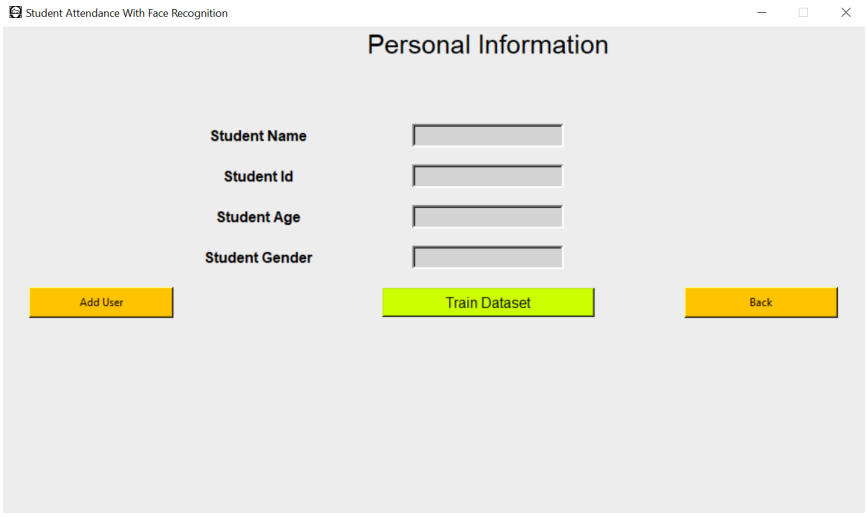


Figure 4.2. ”Add a Student” Menu

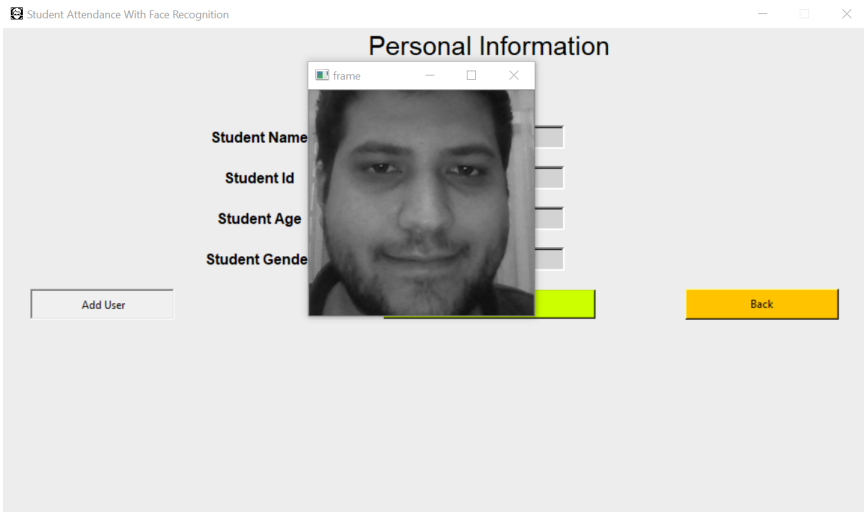


Figure 4.3. During Shooting for Register

Figure 4.4 shows the result of the ”Start Lecture” button. The instructor selects lecture from the combo box list and selects a date from the calendar. Finally, the ”Next” button runs the camera for recording attendance. Taking attendance process is showed in figure 4.5.

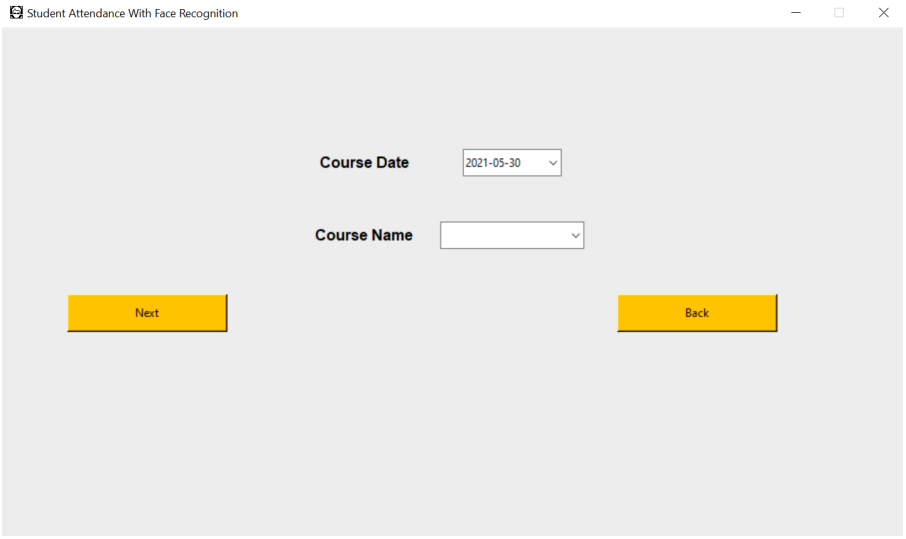


Figure 4.4. ”Start Lecture” Menu

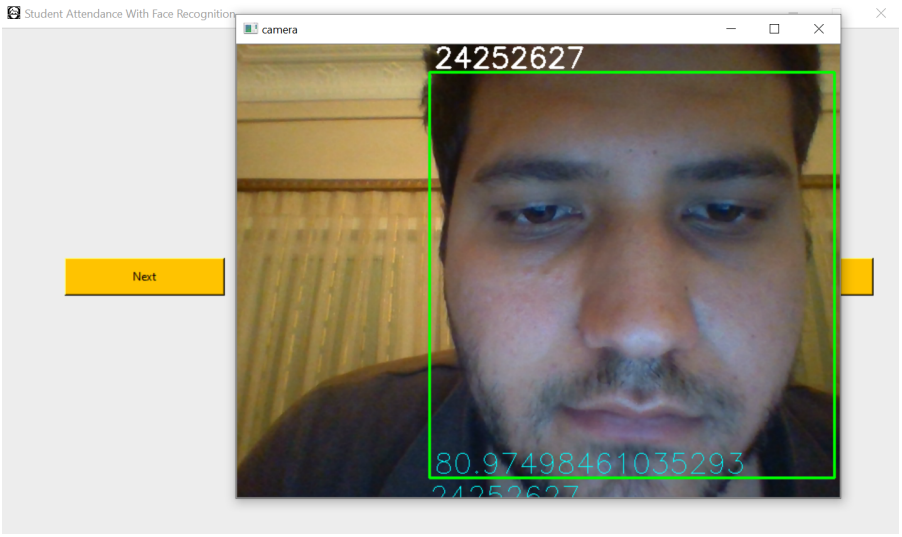


Figure 4.5. During Shooting for Attendance

The last part of the menu is the ”Attendance List” page. For showing attendance list, instructor or student select lecture name, and date which wants to see from combo box and calendar. Then press the ”Attendance List” button and the list is seen on another page. Thanks to the ”Next” and ”Prev” buttons, pages can be changed. Shown the steps in figures 4.6, 4.7, and 4.8.

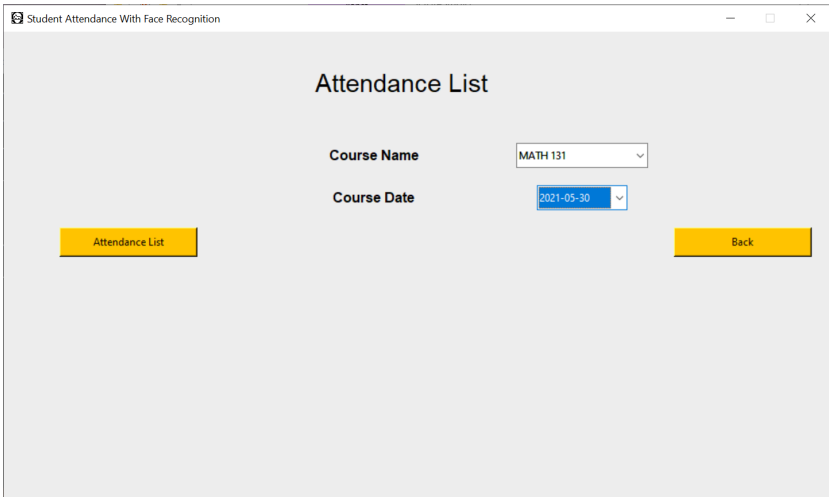


Figure 4.6. ”Attendance List” Menu

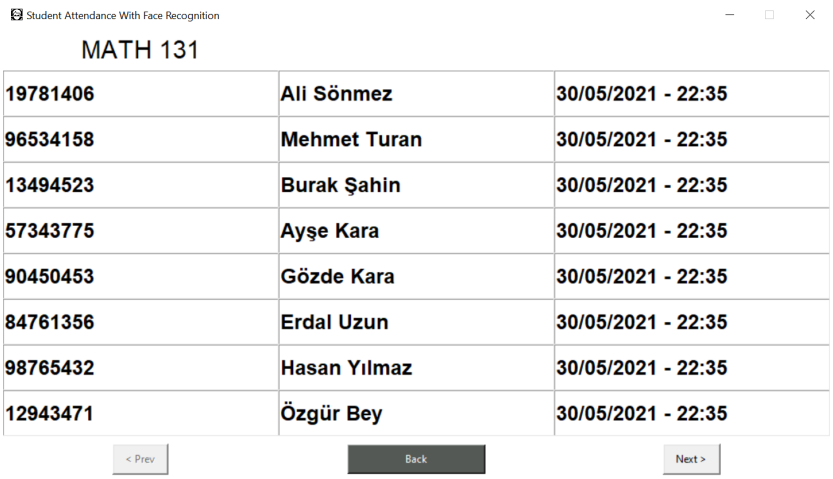
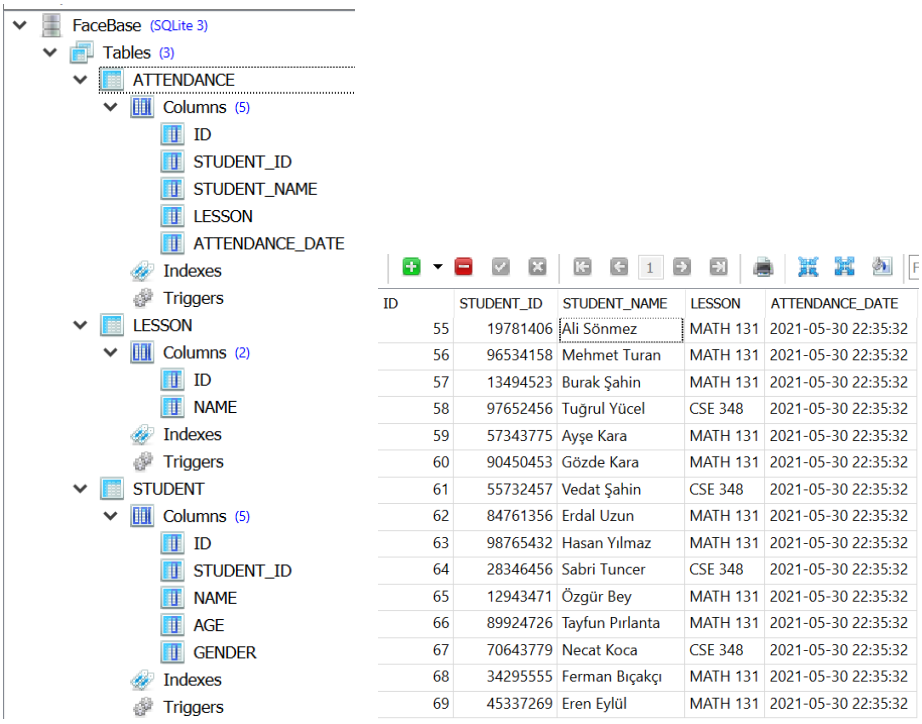


Figure 4.7. Records in the Attendance List

Figure 4.8. Continuation of the Attendance List



ID	STUDENT_ID	STUDENT_NAME	LESSON	ATTENDANCE_DATE
55	19781406	Ali Sönmez	MATH 131	2021-05-30 22:35:32
56	96534158	Mehmet Turan	MATH 131	2021-05-30 22:35:32
57	13494523	Burak Şahin	MATH 131	2021-05-30 22:35:32
58	97652456	Tuğrul Yücel	CSE 348	2021-05-30 22:35:32
59	57343775	Ayşe Kara	MATH 131	2021-05-30 22:35:32
60	90450453	Gözde Kara	MATH 131	2021-05-30 22:35:32
61	55732457	Vedat Şahin	CSE 348	2021-05-30 22:35:32
62	84761356	Erdal Uzun	MATH 131	2021-05-30 22:35:32
63	98765432	Hasan Yılmaz	MATH 131	2021-05-30 22:35:32
64	28346456	Sabri Tuncer	CSE 348	2021-05-30 22:35:32
65	12943471	Özgür Bey	MATH 131	2021-05-30 22:35:32
66	89924726	Tayfun Pırlanta	MATH 131	2021-05-30 22:35:32
67	70643779	Necat Koca	CSE 348	2021-05-30 22:35:32
68	34295555	Ferman Bıçakçı	MATH 131	2021-05-30 22:35:32
69	45337269	Eren Eylül	MATH 131	2021-05-30 22:35:32

(a) Tables and Columns (b) Data in the Attendance Table

Figure 4.9. Data in Tables and Columns

Finally, the database screenshots are shown in figure 4.9(a) and 4.9(b). In the database, attendance, lesson, and student tables are created. The attendance table includes five columns: id, student id, student name, lesson, and attendance date. The lesson table contains two columns, id, and name. They are defined initially as Math 131 and Cse 348. Lastly, the student table includes student's information in five columns as the attendance table. They are id, student id, name, age, and gender. In the right figure, example of the student registers are shown.

5. TEST AND RESULTS

After the implementation was completed, the tests were done. The purpose of these tests is to calculate the accuracy of the program. The performance of the face recognition algorithm has been evaluated by the set of tests collected. Since this program is crucial for faculty and instructors so the margin of error should be minimum [24]. Table 5.1. has information about the distance, light conditions, counts of test, and the accuracy percentages. This application was tested under 3 different distances and 3 different light types. This testing consists of a total of 90 tests, including 10 tests for each face recognition.

5.1. Test Results

Table 5.1. Tests.

Distance	Light Conditions	Number of Tests	Number of True Recognition	Number of False Recognition	Accuracy Rate
30 cm	Normal	10	10	0	100%
30 cm	Bright	10	9	1	90%
30 cm	Dark	10	7	3	70%
50 cm	Normal	10	9	1	90%
50 cm	Bright	10	8	2	80%
50 cm	Dark	10	6	4	60%
>50 cm	Normal	10	6	4	60%
>50 cm	Bright	10	6	4	60%
>50 cm	Dark	10	4	6	40%

Below, the results in the table are interpreted:

- Distances have been set by me before the tests. Close distance is 30 centimeters, normal distance is 50 centimeters and far distance is more than 50 centimeters.
- In the normal light condition, make use of sunlight in the room. In the bright, lamb is used in addition to normal light. However, in the dark light condition, the system was tested in the evening without any lamb.
- If the room is under normal light conditions, the accuracy of the program is the highest.

The program gives correct results mostly, independently of the distance.

- In the position that the computer is placed 30 centimeters from the person. Actually, this distance is the best position for recognizing. In total, 30 tests were carried out at 30 centimeters. The accuracy rate of it is very high as 90%.
- When the light is dark, the accuracy rates decrease. This accuracy ratio decreases as the computer move away from the face. **The camera used in the test was also effective in the decrease of this rate**. Because the quality of my computer's camera is low (0.9 MP). In addition, the camera does not have a night mode. Therefore, the camera performance is very poor.
- When the room is dark, closer distances to the camera give better results than at the far distances.
- In the tests performed in the dark, the application can give false results between different people so it shows that the program has difficulty detecting faces in the dark.
- In the close position, accuracy was high even in the dark because the camera was very close to my face. Since the light of the computer gives shine to my face so the program detects my face more easily. I also tried that situation by changing the brightness of the computer screen.

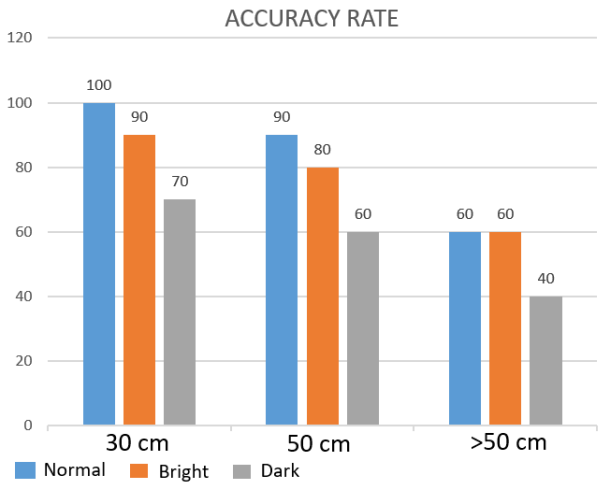
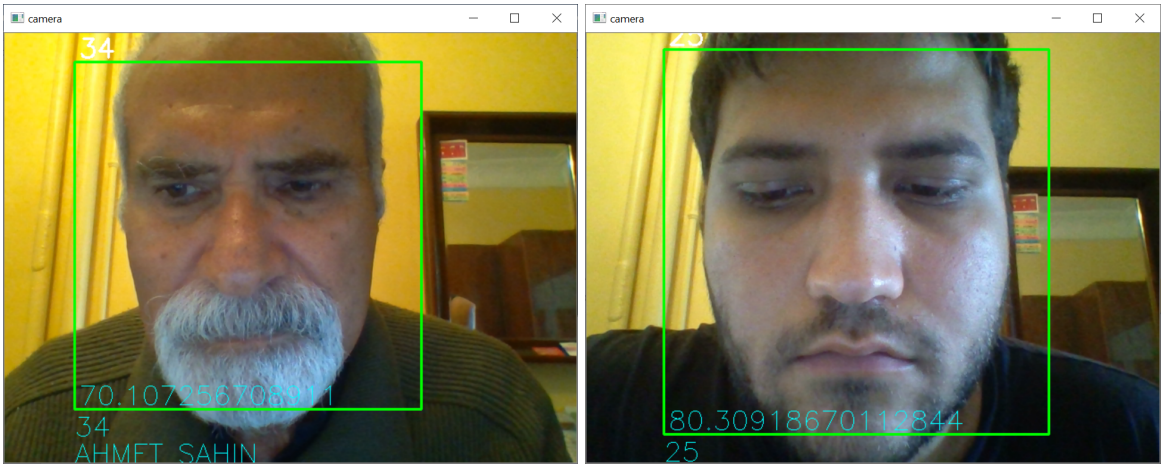


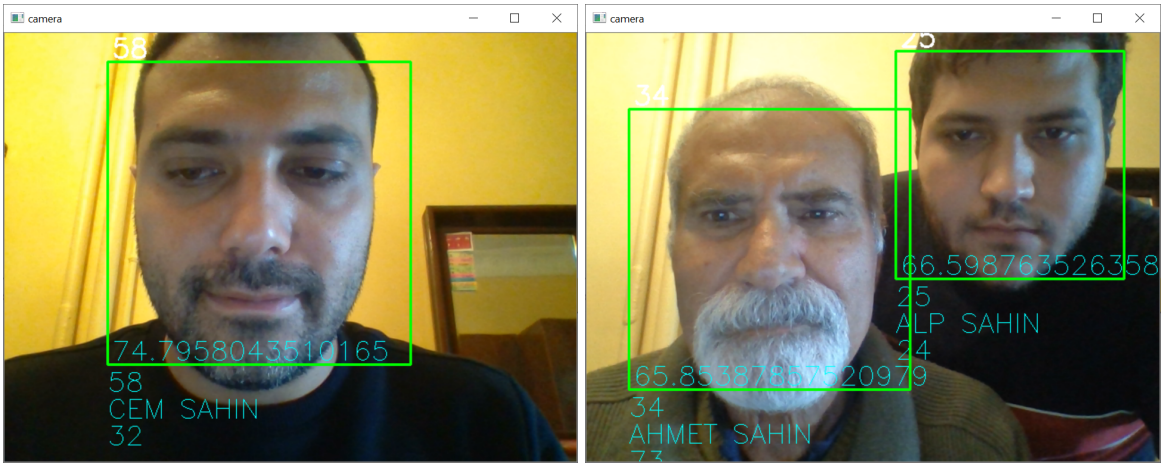
Figure 5.1. Graph of the Accuracy Rate

5.2. Test Screenshots

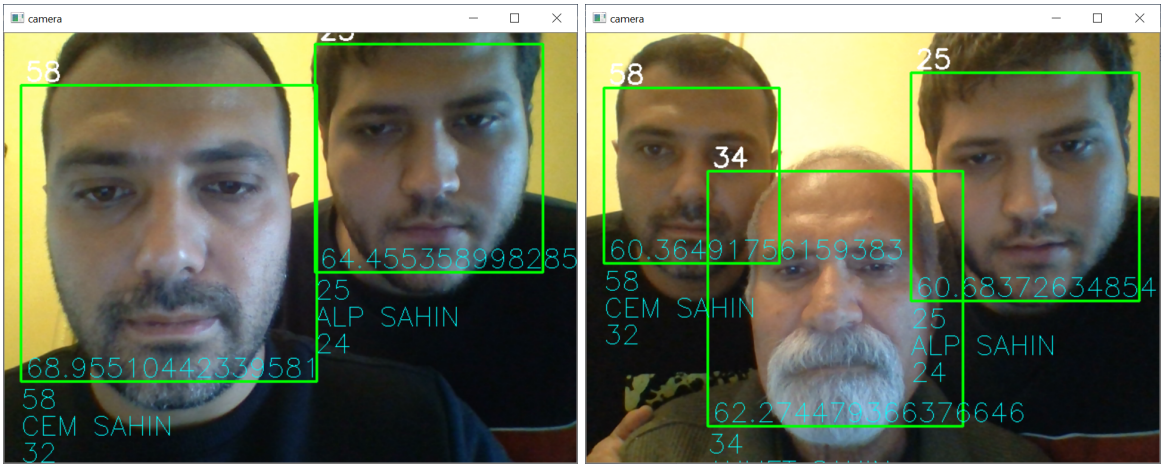
The test screenshots of moment of face recognition are shown in figure 5.2.



(a) Tests Done Alone

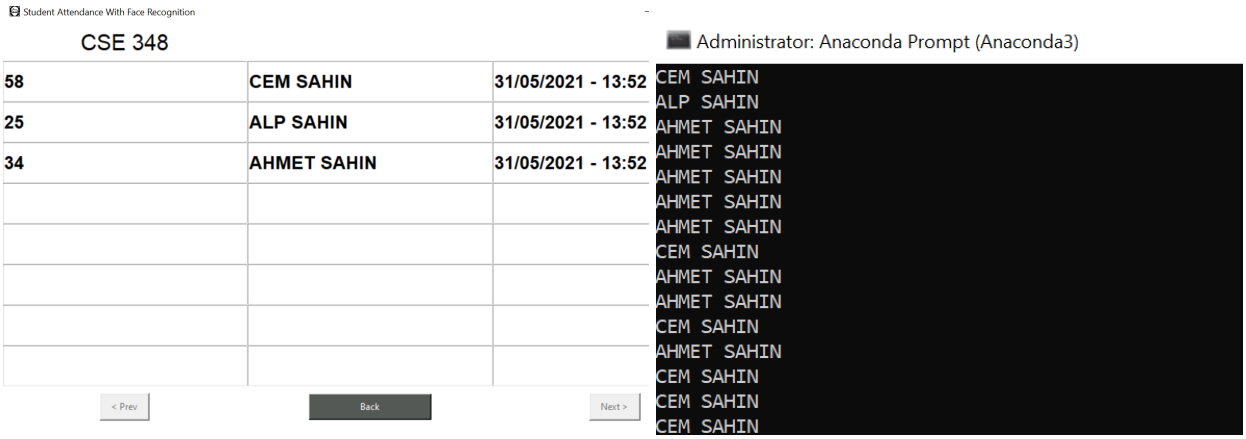


(b) Done Alone and Two-Person Tests

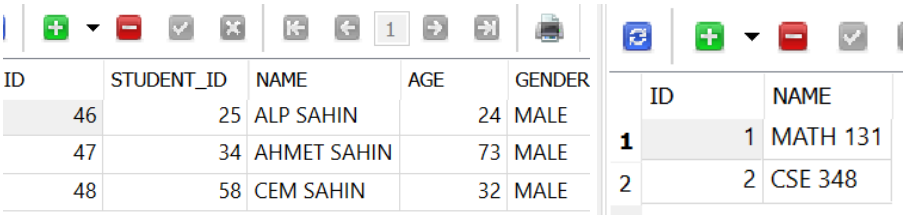


(c) Two-Person and Three-Person Tests

Figure 5.2. Face Recognition Tests With My Family



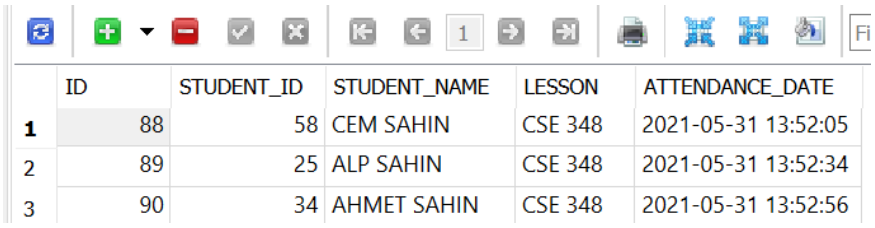
(a) Attendance List After the Tests and Camera Capture Moments in the Terminal



ID	STUDENT_ID	NAME	AGE	GENDER
46	25	ALP SAHIN	24	MALE
47	34	AHMET SAHIN	73	MALE
48	58	CEM SAHIN	32	MALE

	ID	NAME
1	1	MATH 131
2	2	CSE 348

(b) Student and Lecture Databases After the Tests



	ID	STUDENT_ID	STUDENT_NAME	LESSON	ATTENDANCE_DATE
1	88	58	CEM SAHIN	CSE 348	2021-05-31 13:52:05
2	89	25	ALP SAHIN	CSE 348	2021-05-31 13:52:34
3	90	34	AHMET SAHIN	CSE 348	2021-05-31 13:52:56

(c) Attendance Database After the Tests

Figure 5.3. Screenshots of the End of Face Recognition Tests

The test screenshots of the end of face recognition are shown above in figure 5.3.

6. CONCLUSION AND FUTURE WORK

This project's aim is to manage the attendance system with face recognition and make face-based attendance systems permanent in our lives. A desktop application has been developed for managing the attendance system. This application stores the student's faces and creates the student's attendance list. If the students are recorded in the database, the system assigns them to the list. As a result of the tests, a high level of accuracy was achieved. Tests were performed. No problem was found. Even 100% accuracy was achieved in tests with the 30 centimeters distance to the camera and under normal light.

This project can help in the development of future projects. The accuracy ratio will reach 100% even in the dark and far distance, by increasing the camera quality and algorithm quality. In the future, if the students enter the classroom without using this system, the new application can recognize the students without having to go in front of the camera. The system can take attendance automatically and periodically. As a result, the system can be more efficient and more useful.

Bibliography

[1] G. DEE. (2014). Busy schedules, boring lectures drive students to skip classes, [Online]. Available: https://www.browndailyherald.com/2014/04/17/busy-schedules-boring-lectures-drive-students-skip-classes/.

[2] A. Smitha Pavithra S Hegde, "Face recognition based attendance management system," *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT)*, vol. 09, no. 09, 2020.

[3] E. Thaman and R. Kaur, "Face detection using spatio-temporal segmentation and tracking for video security," 2015.

[4] UKEssays. (2018). Face recognition attendance system, [Online]. Available: https://www.ukessays.com/essays/education/recognition-attendance-system-6424.php?vref=1.

[5] R. L. SARFIN. (2015). Church events with facial recognition software, [Online]. Available: https://churchm.ag/facial-recognition-software/.

[6] PRANEETH. (2020). Face recognition based attendance system using deep learning techniques, [Online]. Available: http://www.jcreview.com/?mno=122643%20[Access:%20May%2031,%202021].

[7] A. Choudhary, A. D. Tripathi, A. Bajaj, M. Rathi, and B. Nandini, "Automatic attendance system using facial recognition," *International Journal of Modern Trends in Engineering and Research*, vol. 3, 2016.

[8] S. K. Jha, A. Tyag, K. Kumar, and M. Sharma, "Attendance management using facial recognition," *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT)*, vol. 8, 6 2019.

[9] S. Z. Li and A. K. Jain, *Handbook of Face Recognition*, 2nd. Springer Publishing Company, Incorporated, 2011, ISBN: 085729931X.

[10] A. Naskar, A. Sardar, K. Mondal, and R. Das. (2018). Face detection, [Online]. Available: https://www.rcciit.org/students_projects/projects/cse/2018/GR2.pdf.

[11] Espinosa, S. K. Kenneth, Marvin, D. Tanquiamco, C. Jandayan, and P. grace denilla. (2020). Analysis and design of employee attendance monitoring using face recognition system for archempress fruit corporation, [Online]. Available: https://ssrn.com/abstract=3636604.

[12] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

[13] A. Mittal. (2020). Haar cascades, explained, [Online]. Available: https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d.

[14] K. S. do Prado. (2018). Face recognition: Understanding lbph algorithm, [Online]. Available: https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b.

[15] L. T. H. Phuc, H. Jeon, N. T. N. Truong, and J. J. Hak, "Applying the haar-cascade algorithm for detecting safety equipment in safety management systems for multiple working environments," *Electronics*, vol. 8, no. 10, 2019, ISSN: 2079-9292. DOI: 10.3390/electronics8101079. [Online]. Available: https://www.mdpi.com/2079-9292/8/10/1079.

[16] J. E. C. Cruza1, E. H. Shiguemorib, and L. N. F. Guimar, "A comparison of haar-like, lbp and hog approaches to concrete and asphalt runway detection in high resolution imagery," *PAN-AMERICAN ASSOCIACION OF COMPUTATIONAL INTERDISCIPLINARY SCIENCES*, vol. 6, 3 2015.

[17] P. Irgens, C. Bader, T. Lé, D. Saxena, and C. Ababei, "An efficient and cost effective fpga based implementation of the viola-jones face detection algorithm," *HardwareX*, vol. 1, Mar. 2017. DOI: 10.1016/j.ohx.2017.03.002.

[18] J. Rettkowski, A. Boutros, and D. Göhringer, "Hw/sw co-design of the hog algorithm on a xilinx zynq soc," *Journal of Parallel and Distributed Computing*, vol. 109, pp. 50–62, 2017, ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2017.05.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731517301569.

[19] T. Tuncer, S. Dogan, and F. Ozyurt, "An automated residual exemplar local binary pattern and iterative relieff based covid-19 detection method using chest x-ray image," *Chemometrics and Intelligent Laboratory Systems*, vol. 203, p. 104 054, 2020, ISSN: 0169-7439. DOI: https://doi.org/10.1016/j.chemolab.2020.104054. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743920301970.

[20] A. K. Datta, M. Datta, and P. K. Banerjee, *Face Detection and Recognition: Theory and Practice*. Chapman and amp; Hall/CRC, 2015, ISBN: 1482226545.

[21] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[22] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[23] F. Lundh, "An introduction to tkinter," *pythonware.com/library/tkinter/introduction/index*, 1999.

[24] U. Clark, "Facial recognition test," in *Encyclopedia of Clinical Neuropsychology*, J. S. Kreutzer, J. DeLuca, and B. Caplan, Eds. New York, NY: Springer New York, 2011, pp. 1010–1012, ISBN: 978-0-387-79948-3. DOI: 10.1007/978-0-387-79948-3_1364. [Online]. Available: https://doi.org/10.1007/978-0-387-79948-3_1364.

APPENDIX A: PYTHON CODE

```
//create_dataset.py

import cv2
import sqlite3
import numpy as np

from keras.preprocessing.image import ImageDataGenerator,array_to_img, img_to_array,load_img


STUDENT_ID=""
NAME=""
AGE=""
GENDER=""

def insertOrUpdate(STUDENT_ID, NAME, AGE, GENDER):
    conn=sqlite3.connect("FaceBase.db")
    cmd="SELECT * FROM STUDENT WHERE ID="+str(STUDENT_ID)
    cursor=conn.execute(cmd)
    isRecordExist=0
    for row in cursor:
        isRecordExist=1
    if(isRecordExist==""):
        cmd="UPDATE STUDENT SET Name="+str(NAME)+"WHERE ID="+str(STUDENT_ID)
    else:
        cmd="INSERT INTO STUDENT(STUDENT_ID, NAME, AGE, GENDER)
        Values("+str(STUDENT_ID)+",'"+str(NAME)+"',"+str(AGE)+",
        '"+str(GENDER)+"')"
        conn.execute(cmd)
        conn.commit()
        conn.close()

"""STUDENT_ID=input('enter your student id')
NAME=input('enter your name')
AGE=input('enter your age')
GENDER=input('enter your GENDER')
insertOrUpdate(STUDENT_ID, NAME, AGE, GENDER)
"""
def startCreate(std_id, std_name, std_age, std_gender):
    STUDENT_ID=std_id
    NAME=std_name
    AGE=std_age
    GENDER=std_gender
    insertOrUpdate(std_id, std_name, std_age, std_gender)

    cam = cv2.VideoCapture(0)
    detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    sampleNum=0
    while(True):
        ret, img = cam.read()
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = detector.detectMultiScale(gray, 1.3, 5)
        for (x,y,w,h) in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
```

```
            sampleNum=sampleNum+1

            cv2.imwrite("dataSet/User."+STUDENT_ID+'.'+str(samplNum) + ".jpg", gray[y:y+h,x:x+w])
            datagen =ImageDataGenerator( rotation_range=40,
            width_shift_range=0.2, height_shift_range=0.2,     brightness_range=None,
            shear_range=0.2, zoom_range=0.2, channel_shift_range=0.0,
            fill_mode='nearest', cval=0.0, horizontal_flip=True,
            vertical_flip=False)
            img=load_img("dataSet/User."+STUDENT_ID +'.'+ str(sampleNum) + ".jpg")
            x=img_to_array(img)
            x=x.reshape((1,)+x.shape)
            i=0
            for batch in datagen.flow(x,batch_size=1,save_to_dir="dataSet",
            save_prefix="User."+STUDENT_ID +'._'+
            str(sampleNum),save_format="jpg"):
                i+=1
                if i>15:
                    break

            img = np.array(img)
            cv2.imshow('frame', img)

        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
        elif sampleNum>30:
            break
    cam.release()
    cv2.destroyAllWindows()

//create_classifier.py

import cv2,os
import numpy as np
from PIL import Image

global faces,Ids
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector=cv2.CascadeClassifier("haarcascade_frontalface_default.xml")


def getImagesAndLabels(path):
    #get the path of all the files in the folder
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    #create empth face list
    faceSamples=[]
    #create empty ID list
    Ids=[]

    for imagePath in imagePaths:

        # Updates in Code
        if(os.path.split(imagePath)[-1].split(".")[-1]!='jpg'):
            continue

        #loading the image and converting it to gray scale
        pilImage=Image.open(imagePath).convert('L')
```

```
        #Now we are converting the PIL image into numpy array
        imageNp=np.array(pilImage,'uint8')
        #getting the Id from the image
        Id=int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces=detector.detectMultiScale(imageNp)
        for (x,y,w,h) in faces:
            faceSamples.append(imageNp[y:y+h,x:x+w])
            Ids.append(Id)
    return faceSamples,Ids

def startTrainer():
    faces, Ids = getImagesAndLabels('dataSet')
    recognizer.train(faces, np.array(Ids))
    recognizer.save('trainner/trainner.yml')

//Detector.py

import cv2
import numpy as np
import os
import sqlite3
import time

COURSE = ""
nameList = []

conn=sqlite3.connect("FaceBase.db")

def __del__(self):
    conn.close()

def __init__(self):
    conn=sqlite3.connect("FaceBase.db")
    COURSE = ""
    nameList = []


def startRecognize(lesson, lecture_date):
    conn=sqlite3.connect("FaceBase.db")
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read('trainner/trainner.yml')
    cascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(cascadePath);
    font = cv2.FONT_HERSHEY_SIMPLEX
    COURSE = ""

    nameList.clear()

    cam = cv2.VideoCapture(0)
    cam.set(3, 640) # set video widht
    cam.set(4, 480) # set video height

    minW = 0.1*cam.get(3)
    minH = 0.1*cam.get(4)
    while True:
        ret, img =cam.read()
```

```
img = cv2.flip(img, 1) # Flip vertically
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor = 1.3, # For more reliable results, scaleFactor = 1.05
    minNeighbors = 5,  # and minNeighbors = 3
    minSize = (int(minW), int(minH)),
   )
for(x,y,w,h) in faces:
    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
    id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

    profile=getProfile(id)
    print(profile[2])
    temp_name = profile[2]
    temp_id = profile[1]
    temp_age = profile[3]
    temp_gender = ""


    if (confidence >= 40):
        temp_name = "Unknown"
        temp_id = 0
        temp_age = 0
        temp_gender = 0

    if(profile!=None):
        if(confidence<=40):
            markAttendance(str(profile[1]), str(profile[2]), lesson, lecture_date)
        cv2.putText(
                img,
                str(temp_id),
                (x+5,y-5),
                font,
                1,
                (255,255,255),
                2
               )
        cv2.putText(
                img,
                str(100-confidence),
                (x+5,y+h-5),
                font,
                1,
                (255,255,0),
                1
               )
        cv2.putText(
                img,
                str(temp_id),
                (x,y+h+30),
                font,
                1,
                (255,255,0),
                1
               )
```

```
                cv2.putText(
                        img,
                        str(temp_name),
                        (x,y+h+60),
                        font,
                        1,
                        (255,255,0),
                        1
                       )
                cv2.putText(
                        img,
                        str(temp_age),
                        (x,y+h+90),
                        font,
                        1,
                        (255,255,0),
                        1
                       )
                cv2.putText(
                        img,
                        str(temp_gender),
                        (x,y+h+120),
                        font,
                        1,
                        (255,255,0),
                        1
                       )

        cv2.imshow('camera',img)
        k = cv2.waitKey(10) & 0xff # Press 'ESC' for exiting video
        if k == 27:
            break

    print("\n [INFO] Exiting Program and cleanup stuff")
    cam.release()
    cv2.destroyAllWindows()

def markAttendance(st_id, st_name, lesson, lecture_date):
    if st_id not in nameList:
        print("eseseses")
        nameList.append(st_id)

        cmd="INSERT INTO ATTENDANCE (STUDENT_ID, STUDENT_NAME, LESSON, ATTENDANCE_DATE)
        VALUES (" +st_id+", '"+st_name+"', '"+lesson+"', '" +
        str(lecture_date) + str(time.strftime (" %H:%M:%S")) + "');"
        cursor=conn.execute(cmd)
        conn.commit()

def getProfile(id):
    cmd="SELECT * FROM STUDENT WHERE STUDENT_ID ="+str(id)
    cursor=conn.execute(cmd)
    profile=None
    for row in cursor:
        profile=row
    return profile
```

\\app-gui.py

```
import tkinter as tk
from tkinter import ttk
from tkinter import font as tkfont
from tkinter import messagebox,PhotoImage
from tkcalendar import Calendar, DateEntry
from create_dataset import startCreate
from create_classifier import startTrainer
from Detector import startRecognize
import sqlite3
from tkinter import BOTH, END, LEFT

#from PIL import ImageTk, Image
#from gender_prediction import emotion,ageAndgender


def my_details(controller, lesson, course_date, rec_date, offset):
    controller.show_frame("PageFour")
    controller.update_frame("PageFour", lesson, course_date, rec_date, offset)


def on_createUser(id_var, name_var, age_var, gender_var):
    startCreate(str(id_var.get()),str(name_var.get()),str(age_var.get()) ,str(gender_var.get()))

def on_startTrainer():
    if messagebox.askokcancel("Trainer", "Dataset will be trained."):
        startTrainer()

def on_detector(lesson, lecture_date):
    startRecognize(lesson, lecture_date)

class MainUI(tk.Tk):
    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        self.title_font = tkfont.Font(family='Helvetica', size=16, weight="bold")
        self.title("Student Attendance With Face Recognition")
        self.resizable(False, False)
        self.protocol("WM_DELETE_WINDOW", self.on_closing)
        self.active_name = None
        self.geometry("960x540")
        container = tk.Frame(self)
        container.pack(side = "top", fill = "both", expand = True)

        container.grid_rowconfigure(0, weight = 1)
        container.grid_columnconfigure(0, weight = 1)
        self.frames = {}
        for F in (StartPage, PageOne, PageTwo, PageThree, PageFour):
            page_name = F.__name__
            frame = F(parent=container, controller=self)
            self.frames[page_name] = frame
            frame.grid(row=0, column=0, sticky="nsew")
            self.show_frame("StartPage")

    def show_frame(self, page_name):
            frame = self.frames[page_name]
            frame.tkraise()
```

```
    def update_frame(self, page_name, lecture, course_date, rec_date, offset):
        frame = self.frames[page_name]
        frame.update_list(lecture, course_date, rec_date, offset)

    def on_closing(self):

        if messagebox.askokcancel("Quit", "Are you sure?"):
            self.destroy()



class StartPage(tk.Frame):

        def __init__(self, parent, controller):
            tk.Frame.__init__(self, parent)
            self.controller = controller
            render = PhotoImage(file='logoalp.png')
            img = tk.Label(self, image=render, borderwidth=0)
            img.image = render
            img.grid(row=0, column=8, rowspan=14, sticky="nsew")
            label = tk.Label(self, text="ATTENDANCE MANAGER", font='Helvetica 22 bold', fg="black", bg="#EDEDED")
            label2 = tk.Label(self, text="                      ", font='Helvetica 22 bold',
            fg="#122B3F", bg="#EDEDED")
            label.grid(row=12, column=8,padx=10, pady=10)
            label2.grid(row=12, column=1,padx=20, pady=10)

            button1 = tk.Button(self, text="    Add a Student    ", font='Helvetica 12 bold', fg="#000",
            bg="#FFC300", height=1, width=35, command=lambda: self.controller.show_frame("PageOne"))
            button2 = tk.Button(self, text="    Start Lecture    ", font='Helvetica 12 bold', fg="#000",
            bg="#FFC300", height=1, width=35, command=lambda: self.controller.show_frame("PageTwo"))
            button4 = tk.Button(self, text="   Attendance List   ", font='Helvetica 12 bold', fg="#000",
            bg="#FFC300", height=1, width=35, command=lambda: self.controller.show_frame("PageThree"))

            button1.grid(row=13, column=8,padx=10, pady=10, ipady=5)
            button2.grid(row=14, column=8,padx=10, pady=10, ipady=5)
            button4.grid(row=15, column=8,padx=10, pady=10, ipady=5)

            self.configure(bg= "#EDEDED", pady=1)


        def on_closing(self):
            if messagebox.askokcancel("Quit", "Are you sure?"):
                self.controller.destroy()




class PageOne(tk.Frame):
    def __init__(self, parent, controller):
        id_var=tk.StringVar()
        name_var=tk.StringVar()
        age_var=tk.StringVar()
        gender_var=tk.StringVar()

        tk.Frame.__init__(self, parent)
        self.controller = controller
```

```
        label2 = tk.Label(self, text="                      ", font='Helvetica 22 bold', fg="#122B3F", bg="#EDEDED")
        label2.grid(row=3, column=1,padx=20, pady=10)

        label = tk.Label(self, text="Personal Information",
        font='Helvetica 22 ', fg="black", bg="#EDEDED")
        label.grid(row=1,  column=3,padx=0)


        tk.Label(self, text="Student Name", bg="#EDEDED", fg="#000", font='Helvetica 12 bold').grid(row=8,
        column=2, pady=10, padx=1,)
        self.user_name = tk.Entry(self, textvariable = name_var, borderwidth=3, bg="lightgrey",
        font='Helvetica 11 bold')
        self.user_name.grid(row=8,  column=3,padx=0)

        tk.Label(self, text="Student Id", bg="#EDEDED", fg="#000", font='Helvetica 12 bold').grid(row=9,
        column=2, pady=10, padx=0,)
        self.user_id = tk.Entry(self, textvariable = id_var, borderwidth=3, bg="lightgrey",
        font='Helvetica 11 bold')
        self.user_id.grid(row=9, column=3,padx=0)

        tk.Label(self, text="Student Age", bg="#EDEDED", fg="#000", font='Helvetica 12 bold').grid(row=10,
        column=2, pady=10, padx=0,)
        self.user_age = tk.Entry(self, textvariable = age_var, borderwidth=3, bg="lightgrey",
        font='Helvetica 11 bold')
        self.user_age.grid(row=10, column=3,padx=0)

        tk.Label(self, text="Student Gender", bg="#EDEDED", fg="#000", font='Helvetica 12 bold').grid(row=11,
        column=2, pady=10, padx=0,)
        self.user_gender = tk.Entry(self, textvariable = gender_var, borderwidth=3, bg="lightgrey",
        font='Helvetica 11 bold')
        self.user_gender.grid(row=11,  column=3,padx=0)

        self.buttoncanc = tk.Button(self, text="Back", bg="#FFC300", fg="#000", width=20,
        command=lambda: controller.show_frame("StartPage"))
        self.buttoncanc.grid(row=12,  column=10,padx=25, ipadx=10, ipady=4, pady=10)

        self.buttonadd = tk.Button(self, text ="Add User", bg="#FFC300", fg="#000", width=20,
        command=lambda: on_createUser(id_var, name_var, age_var, gender_var))
        self.buttonadd.grid(row=12, column=1, pady=10, padx=5, ipadx=5, ipady=4)

        button3 = tk.Button(self, text="Train Dataset", font='Helvetica 12', fg="#000", bg="#CCFF00",
        height=1, width=25, command=lambda: on_startTrainer()) button3.grid(row=12, column=3,padx=75, pady=10)


        self.configure(bg="#EDEDED")


class PageTwo(tk.Frame):

    def __init__(self, parent, controller):
        coursename_var = tk.StringVar()

        tk.Frame.__init__(self, parent)
        self.controller = controller

        label = tk.Label(self, text=" ", font='Helvetica 22 ', fg="black", bg="#EDEDED")
        label.grid(row=2,  column=2,ipadx=5, pady=40)
```

```
        tk.Label(self, text="Course Name", bg="#EDEDED", fg="#000", font='Helvetica 12 bold').grid(row=8,
        column=2, pady=40, padx=16)
        tk.Label(self, text="Course Date", bg="#EDEDED", fg="black", font='Helvetica 12 bold').
        grid(row=6, column=2, pady=10, padx=25)


        self.buttoncanc = tk.Button(self, text="Back", command=lambda: controller.show_frame("StartPage"),
        bg="#FFC300", fg="#000", width=20)
        self.menuvar = tk.StringVar(self)
        self.buttonext = tk.Button(self, text="Next", command=lambda: on_detector(self.cb.get(),
        self.lesson_date.get_date()), fg="#000", bg="#FFC300", width=20)
        self.buttoncanc.grid(row=10,  column=5,padx=30, pady=10, ipadx=10, ipady=7)
        self.buttonext.grid(row=10, column=1, pady=10, padx=70,ipadx=10, ipady=7)

        self.cb = ttk.Combobox(self, state="readonly")
        self.cb['values'] = self.combo_input()
        self.cb.grid(row=8,ipadx=5, ipady=4, column=3, pady=10, padx=5)

        #date
        self.lesson_date = DateEntry(self, state="readonly", locale='tr_TR', date_pattern='yyyy-mm-dd')
        self.lesson_date.grid(row=6, ipadx=5, ipady=4, column=3, pady=10, padx=5)

        self.configure(bg="#EDEDED")

    def combo_input(self):
        db = sqlite3.connect('FaceBase.db')
        cursor = db.cursor()
        cursor.execute('SELECT NAME FROM LESSON')
        data = []

        for row in cursor.fetchall():
            data.append(row[0])

        db.close()
        return data




class PageThree(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        course_var=tk.StringVar()
        section_var=tk.StringVar()

        self.controller = controller

        #header label
        label = tk.Label(self, text="        Attendance List", font='Helvetica 22 ', fg="black", bg="#EDEDED")
        label.grid(row=2,  column=2,ipadx=5, pady=40)

        #name label
        tk.Label(self, text="Course Name", bg="#EDEDED", fg="black", font='Helvetica 12 bold').grid(row=5,
        column=2, pady=10, padx=25)
```

```
        #date label
        tk.Label(self, text="Course Date", bg="#EDEDED", fg="black", font='Helvetica 12 bold').grid(row=6,
        column=2, pady=10, padx=25)

        #back button
        self.buttoncanc = tk.Button(self, text="Back", command=lambda: controller.show_frame("StartPage"),
        bg="#FFC300", fg="#000", width=20)
        self.buttoncanc.grid(row=7, column=4, ipadx=5, ipady=4,pady=10, padx=5)

        #show attendance button
        self.attendance_list = tk.Button(self, text="Attendance List", command=lambda:
        [my_details(controller, self.cb.get(), self.lesson_date.get_date(), 0, 0)], bg="#FFC300", fg="#000",
        width=20)
        self.attendance_list.grid(row=7, column=1, ipadx=5, ipady=4, pady=10, padx=65)

        #lesson
        self.cb = ttk.Combobox(self, state="readonly")
        self.cb['values'] = self.combo_input()
        self.cb.grid(row=5, ipadx=5, ipady=4, column=3, pady=10, padx=25)

        #date
        self.lesson_date = DateEntry(self, state="readonly", locale='tr_TR', date_pattern='yyyy-mm-dd')
        self.lesson_date.grid(row=6, ipadx=5, ipady=4, column=3, pady=10, padx=5)

        self.configure(bg="#EDEDED")

    def combo_input(self):
        db = sqlite3.connect('FaceBase.db')
        cursor = db.cursor()
        cursor.execute('SELECT NAME FROM LESSON')
        data = []

        for row in cursor.fetchall():
            data.append(row[0])

        db.close()
        return data



class PageFour(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

    def update_list(self, lesson, course_date, rec_date, offset):
        my_c = sqlite3.connect("FaceBase.db")
        my_c = my_c.cursor()
        r_set=my_c.execute("SELECT count(*) FROM ATTENDANCE WHERE LESSON='"+lesson+"' AND ATTENDANCE_
        DATE>='"+str(course_date)+" 00:00:00' AND ATTENDANCE_DATE<'"+str(course_date)+" 23:59:59'")
        data_row=r_set.fetchone()
        nu_rec=data_row[0] # Total number of rows in table
        limit = 8; # No of records to be shown per page.
        print(str(nu_rec) + "-" + str(course_date))
```

```
self.buttoncanc = tk.Button(self, text="Back", command=lambda: [self.controller.show_frame("PageThree")],
bg="#555956", fg="#fff", width=20)
self.buttoncanc.grid(row=13, column=1, ipadx=5, ipady=4,pady=10, padx=5)

if(nu_rec > 0):
    try:
        try:
            q = "SELECT STUDENT_ID, STUDENT_NAME, STRFTIME('%d/%m/%Y - %H:%M', ATTENDANCE_DATE) AS
            ATTENDANCE_DATE FROM ATTENDANCE  WHERE LESSON='"+lesson+"' AND ATTENDANCE_DATE>=
            '"+str(course_date)+" 00:00:00' AND ATTENDANCE_DATE<'"+str(course_date)+" 23:59:59' LIMIT "
            + str(offset) + "," +str(limit)
            r_set=my_c.execute(q)
            i=1 # row value inside the loop
            label = tk.Label(self, text=lesson, font='Helvetica 22 ', fg="black", bg='white')
            label.grid(row=0, column=0,ipadx=5, pady=5)


            for student in r_set:
                for j in range(len(student)):
                    self.e = tk.Entry(self, width=20, fg='black', bg='white', font = "Helvetica 18 bold")
                    self.e.grid(row=i, column=j, ipadx=28, ipady=10)
                    self.e.insert(END, student[j])
                i=i+1
            while (i<=limit):
                for j in range(len(student)):
                    self.e = tk.Entry(self, width=20, fg='black', bg='white', font = "Helvetica 18 bold")
                    self.e.grid(row=i, column=j, ipadx=28, ipady=10)
                    self.e.insert(END, "")
                i=i+1

            back = offset - limit
            next = offset + limit

            b1 = tk.Button(self, text='Next >', command=lambda: self.update_list(lesson, course_date,
            rec_date, next))
            b1.grid(row=13,column=2, ipadx= 10, ipady=5, pady=5)
            b2 = tk.Button(self, text='< Prev', command=lambda: self.update_list(lesson, course_date,
            rec_date, back))
            b2.grid(row=13,column=0, ipadx= 10, ipady=5, pady=5)

            if(nu_rec <= next):
                b1["state"]="disabled"
            else:
                b1["state"]="active"

            if(back >= 0):
                b2["state"]="active"
            else:
                b2["state"]="disabled"
        except:
            print("db error")
    except:
        print("db error")
else:
    self.controller.show_frame("PageThree")
my_c.close()
self.configure(bg='white')
```

```
app = MainUI()
app.iconphoto(False, tk.PhotoImage(file='ikon.png'))
app.mainloop()
```