Shape, polygon

Description automatically generated

Logo

Description automatically generated

Ministry of Higher Education and Scientific Research

Higher Institute of Engineering in El-Shourouk City

Department of Communications and Computer Engineering

**Face recognition attendance system**

Submitted in partial fulfillment for the requirements for the degree of Bachelor’s   
in Electronics and Communication \ Computer and Control Engineering

**Project Team**

Abdullah Gamil Mohamed Ahmed Omar Gamal Saad Ali

Abdullah Sayed Mohamed Shokry Abdulrahman Mahmoud Ahmed

**Supervisors**

**Associated prof. Ahmed Mostafa El-Mahalawy**

Communication and Computer Engineering Dept., Higher Institute of Engineering, El-shorouk Academy

**Eng. Baraa Ahmed**

Communication and Computer Engineering Dept., Higher Institute of Engineering, El-shorouk Academy

**Sponsors**

ITIDA

Cairo 2022

Shape, polygon

Description automatically generatedLogo

Description automatically generatedوزارة التعليم العالي والبحث العلمي

المعهد العالي للهندسة بمدينة الشروق

قسم هندسة الاتصالات والحاسبات

**نظام الحضور الذكي باستخدام تقنية التعرف على الوجه**

مقدم كجزء من متطلبات الحصول على درجة البكالوريوس في هندسة الاتصالات / الحاسبات و التحكم

**فريق المشروع**

عبدالله جميل محمد أحمد عمر جمال سعد علي

عبدالله سيد محمد شكري عبدالرحمن محمود أحمد طه

**المشرفون**

**أ.م.د. أحمد مصطفى المحلاوي**

قسم هندسة الاتصالات والحاسبات – المعهد العالي للهندسة – أكاديمية الشروق

**م. براء أحمد**

قسم هندسة الاتصالات والحاسبات – المعهد العالي للهندسة – أكاديمية الشروق

**الرعاة**

ITIDA

**القاهرة 2022**

**Chapter 1**

**Introduction**

**1.1 Introduction**

Facial recognition is a way of identifying or confirming an individual’s identity using

their face.

Facial recognition systems can be used to identify people in photos, videos,

or in real-time. Facial recognition is a category of biometric security. Other forms of

biometric software includes voice recognition, fingerprint recognition, and eye retina

or iris recognition. The technology is mostly used for security and law enforcement,

though there is increasing interest in other areas of use.

Many people are familiar with face recognition technology through the FaceID used to

unlock iPhones (however, this is only one application of face recognition). Typically, facial

recognition does not rely on a massive database of photos to determine an individual’s

identity — it simply

identifies and recognizes one person as the sole owner of the device, while limiting

access to others.

Some educational institutions in China use face recognition to ensure students are not

skipping class.

Tablets are used to scan students' faces and match them to photos in a database to

validate their identities.

More broadly, the technology can be used for workers to sign in and out of their workplaces,

so that employers can track attendance.

## **1.2 Project objectives:**

## **- Security:**

* Using security cameras doesn’t allow users enter without registration in the system.
* Using temperature sensor to take body temperature from the users to take a decision if the body temperature of user is normal then the system allow person to enter the building and take his attendance, and if his temperature is not normal then the system does not allow the user to enter and does not take his attendance.

##### **Comfortable:**

* Controlling the attendance system to utilize a virtual assistant or a mobile application will grab the comfortable for the users.
* By the application you can track your attendance and know your salary.
* **Safety:**
* With our system we replace the fingerprint with a camera to take the attendance, so we reduce the risk of infection of COVID-19

## **1.3 Project Development Methodology**

The main characteristics of the Waterfall model are a sequential progression through the different

stages of a project from initiation to the delivery phase. The waterfall model does have its limitations

Diagram

Description automatically generatedand because of this, there have been many spins off models created over the years.

*Figure 1 (1-1 Water fall model)*

The Waterfall model consists of the following phases:

* Requirements
* Analysis
* Design
* Implementation
* Testing
* Maintenance

The waterfall model used with the SDLC progresses through the six phases. You can picture

this process as a waterfall with one phase flowing into the next.

##### **Requirements Phase**

the purpose of the initiation phase is to conduct an initial high-level investigation of the business

need and come up with a recommendation for the solution. Once approved by the

management team, stakeholders, client or project sponsor, it will proceed to the next phase.

##### **Analysis Phase**

the purpose of the requirements analysis phase is to conduct a detailed analysis of the current

business needs and identify what options is available to achieve those business needs.

During the Analysis Phase, the Business Analyst will create the Business Requirements

Document (BRD).

##### **Design Phase**

The purpose of the design phase is to identify and document a solution that will be constructed

including technical and procedural specifications. A design document will be created that should

include but not limited to technical, environmental, data, program, procedural, testing specifications.

##### **Implementation Phase**

The construction or development phase is where a resource will take the design document created

during the design phase and translate it into a functional program or system.

##### **Testing Phase**

The purpose of the testing phase is to test the system and related procedures that it meets the requirements specified by the stakeholders and documented in the BRD, design plan, and testing plan.

##### **Maintenance Phase**

the purpose of the implementation phase is to release a fully tested and operational product to an end user or customer. The product should meet all the requirements that were documented in the BRD and pass the testing phase before it can be released to a production environment.

##### **Advantages of waterfall model:**

* This model is simple and easy to understand and use.
  + It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
* In this model phases are processed and completed one at a time. Phases do not overlap.
  + Waterfall model works well for smaller projects where requirements are very well understood.

##### **Disadvantages of waterfall model:**

* + Once an appl is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to risk of changing.

##### **Why waterfall Model is for our project?**

* Product definition is stable.
* The requirements are very well known, clear and fixed. There are no ambiguous requirements
* Technology is well understood.

## **1.4Project Phases**

In the following, the steps of project will be mentioned as:

Step 1: analysis and requirement gathering

Step 2: General planning and mind storming

Step 3: implement attendance algorithm (Software)

Step 4: test

Step 5: implement mobile application (Software)

Step 6: test in emulator

Step 7: test in real mobile

Step 8: Electronic Parts (raspberry pi and temperate sensor)

Step 9: implement to algorithm in raspberry pi

Step 10: test

Step 11: Connect all parts of the system together by Firebase

Step 12: test whole project

Step 13: Build

1.5 Thesis’s structure

In the next chapter we will show the background of our project and the evolution of the attendance system.

**Chapter 2**

**Background**

# 2.1 About Project

The first attempts to use face recognition began in the 1960’s with a semi-automated system. Marks were made on photographs to locate the major features; it used features such as eyes, ears, noses, and mouths.

Then distances and ratios were computed from these marks to a common reference point and compared to reference data. In the early 1970’s Goldstein, created a system of 21 subjective markers such as hair color and lip thickness. This proved even harder to automate due to the subjective nature of many of the measurements still made completely by hand.

Having an efficient attendance system is important in every organization. Therefore, we create our unique system which we named it (Face Attendance System & Temperature Detection). It is a system with camera to detect the faces and take the attendance and it also provides temperature detection to measure the temperature with alarm to alert when the person’s temperature is higher than the average.

Moreover, we will create a mobile application to follow employee’s attendance with it.

The system compares between the photo of the person in the database and the person who standup in front of the camera. At the same time, it takes the temperature with the temperature detection device. In case the photo and the person match, the system will take his attendance, And from the application you can track your attendance.

In other attendance systems like fingerprint attendance, the system works by touching the fingerprint in the device. This means that all people who work in this organization have to touch it to take their attendance.

Now with the COVID-19 epidemic, using such a system may cause a problem because the virus can be transmitted through touching. It only takes one person have the virus to speared it in all the organization.

With our system we replace the fingerprint with a camera to take the attendance, so we reduce the risk of infection.

Another reason for creating this system is to replace the old version of the temperature measurement device that need a person to use it to take a temperature. In our system we will create a device (Temperature Detection) which take the temperature without any human help to prevent contact or being physically close with more people.

Moreover, this system will also help organization to track the number of suspected COVID-19 cases in the organization (people with temperature more than the normal average), so the organization can take the precautionary measures.

## **2.2 Problem Statement**

Traditional student attendance marking technique is often facing a lot of trouble. The face recognition student attendance system emphasizes its simplicity by eliminating classical student attendance marking technique such as calling student names or checking respective identification cards.

There are not only disturbing the teaching process but also causes distraction for students during exam sessions. Apart from calling names, attendance sheet is passed around the classroom during the lecture sessions. The lecture class especially the class with a large number of students might find it difficult to have the attendance sheet being passed around the class. Thus, face recognition student attendance system is proposed to replace the manual signing of the presence of students which are burdensome and causes students get distracted in order to sign for their attendance.

Furthermore, the face recognition based automated student attendance system able to overcome the problem of fraudulent approach and lecturers does not have to count the number of students several times to ensure the presence of the students.

list the difficulties of facial identification. One of the difficulties of facial identification is the identification between known and unknown images. In addition, found out that the training process for face recognition student attendance system is slow and time-consuming.

In addition, mentioned that different lighting and head poses are often the problems that could degrade the performance of face recognition-based student attendance system. Hence, there is a need to develop a real time operating student attendance system which means the identification process must be done within defined time constraints to prevent omission. The extracted features from facial images which represent the identity of the students must be consistent towards a change in background, illumination, pose and expression. High accuracy and fast computation time will be the evaluation points of the performance.

**2.3 Evolution Of Attendance System:**

**1. Manual handwritten journal:**

surprisingly, some companies still use this method to track employee attendance. Yet most modern companies don’t use it anymore because it’s inconvenient and requires to have a responsible person for filling it in.



*Figure 2 (2-1 annual handwritten journal)*

**2. Paper timesheets:**

it’s already a thing of a past. extremely inefficient now. because employees have to constantly keep track of what they do, and if they don`t – they have to come up with something. So there’s a place for fraud here. Secondly, the procession of paper time sheets takes a lot of time.

****

*Figure 3 (2-2 paper timesheets)*

**3. Traditional punch clock :**

it’s a mechanical or electronic device used to track employee attendance. Traditionally the data was recorded on paper punch cards which were physically stamped or punched by the clock. Paper punch cards are still used at some workplaces even today.



*Figure 2.4:Ttraditional Punch Clock*

**4. Biometric attendance systems**:

biometric attendance systems have been used for security purposes since 1996 and was first installed at the Olympic Village in Atlanta. Biometric systems are still used for security purposes and attendance tracking because it is impossible to copy a finger or handprint.



*Figure 5 (2-4 biometric attendance system)*

5. **Face attendance system:**

Face recognition is an Attendance system recognizing the person by using face biostatistics is aimed to accomplish digitization of the traditional system of   
taking attendance, Present strategies for taking attendance are tedious and time-consuming.   
The traditional process of making attendance and present biometric systems are vulnerable to proxies.

Icon

Description automatically generated

*Figure 6 (2-5 face attendance system)*

2.4 Aims and Objectives

The objective of this project is to develop face recognition based automated attendance system. Expected achievements in order to fulfill the objectives are:

• To detect the face segment from the video frame.

• To extract the useful features from the face detected.

• To classify the features in order to recognize the face detected.

• To record the attendance of the identified student.

In the next chapter we will show the theoretical background of the main 4 technology we use in this project

**Chapter 3**

**Theoretical Basics**

**3.1 History Of Facial Recognition Technology**

Automated facial recognition was pioneered in the 1960s. Woody Bledsoe, Helen Chan Wolf, and Charles Bisson worked on using the computer to recognize human faces. Their early facial recognition project was dubbed "man-machine" because the coordinates of the facial features in a photograph had to be established by a human before they could be used by the computer for recognition.

On a graphics tablet a human had to pinpoint the coordinates of facial features such as the pupil centers, the inside and outside corner of eyes, and the widows peak in the hairline. The coordinates were used to calculate 20 distances, including the width of the mouth and of the eyes. A human could process about 40 pictures an hour in this manner and so build a database of the computed distances. A computer would then automatically compare the distances for each photograph, calculate the difference between the distances and return the closed records as a possible match.

- In 1970, Takeo Kanade publicly demonstrated a face-matching system that located anatomical features such as the chin and calculated the distance ratio between facial features without human intervention. Later tests revealed that the system could not always reliably identify facial features. Nonetheless, interest in the subject grew and in 1977 Kanade published the first detailed book on facial recognition technology.

## - The dawn of Facial Recognition – 1960s

The earliest pioneers of facial recognition were Woody Bledsoe, Helen Chan Wolf and Charles Bisson. In 1964 and 1965, Bledsoe, along with Wolf and Bisson began work using computers to recognize the human face.

Due to the funding of the project originating from an unnamed intelligence agency, much of their work was never published. However, it was later revealed that their initial work involved the manual marking of various “landmarks” on the face such as eye centers, mouth etc.

These were then mathematically rotated by a computer to compensate for pose variation. The distances between landmarks were also automatically computed and compared between images to determine identity.

These earliest steps into Facial Recognition by Bledsoe, Wolf and Bisson were severely hampered by the technology of the era, but it remains an important first step in proving that Facial Recognition was a viable biometric.

## Advancing the accuracy of Facial Recognition – 1970s

Carrying on from the initial work of Bledsoe, the baton was picked up in the 1970s by Goldstein, Harmon and Lesk who extended the work to include 21 specific subjective markers including hair color and lip thickness in order to automate the recognition.

While the accuracy advanced, the measurements and locations still needed to be manually computed which proved to be extremely labour intensive yet still represents an advancement on Bledsoe’s RAND Tablet technology.

## Using linear algebra for Facial Recognition – 1980s/90s

It wasn’t until the late 1980s that we saw further progress with the development of Facial Recognition software as a viable biometric for businesses. In 1988, Sirovich and Kirby began applying linear algebra to the problem of facial recognition.

A system that came to be known as Eigenface showed that feature analysis on a collection of facial images could form a set of basic features. They were also able to show that less than one hundred values were required in order to accurately code a normalized facial image.

In 1991, Turk and Pentland carried on the work of Sirovich and Kirby by discovering how to detect faces within an image which led to the earliest instances of automatic facial recognition. This significant breakthrough was hindered by technological and environmental factors, however, it paved the way for future developments in Facial Recognition technology.

FERET Programme – 1990s/2000s

The Defense Advanced Research Projects Agency (DARPA) and the National Institute of Standards and Technology (NIST) rolled out the Face Recognition Technology (FERET) programme in the early 1990s in order to encourage the commercial facial recognition market. The project involved creating a database of facial images. Included in the test set were 2,413 still facial images representing 856 people. The hope was that a large database of test images for facial recognition would inspire innovation and may result in more powerful facial recognition technology.

## Face Recognition Vendor Tests – 2000s

The National Institute of Standards and Technology (NIST) began Face Recognition Vendor Tests (FRVT) in the early 2000s. Building on FERET, FRVTs were designed to provide independent government evaluations of facial recognition systems that were commercially available, as well as prototype technologies. These evaluations were designed to provide law enforcement agencies and the U.S. government with the information necessary to determine the best ways to deploy facial recognition technology.

## Face Recognition Grand Challenge – 2006

Launched in 2006, the primary goal of the Face Recognition Grand Challenge (FRGC) was to promote and advance face recognition technology designed to support existing face recognition efforts in the U.S. Government.

The FRGC evaluated the latest face recognition algorithms available. High-resolution face images, 3D face scans, and iris images were used in the tests. The results indicated that the new algorithms were 10 times more accurate than the face recognition algorithms of 2002 and 100 times more accurate than those of 1995, showing the advancements in facial recognition technology over the past decade.

## Social Media – 2010-Current

Back in 2010, Facebook began implementing facial recognition functionality that helped identify people whose faces may feature in the photos that Facebook users update daily. The feature was instantly controversial with the news media, sparking a slew of privacy-related articles. However, Facebook users by and large did not seem to mind. Having no apparent negative impact on the website’s usage or popularity, more than 350 million photos are uploaded and tagged using face recognition each day.

## iPhone X – 2017

Facial Recognition technology advanced rapidly from 2010 onwards and September 12, 2017, was another significant breakthrough for the integration of facial recognition into our day to day lives. This was the date that Apple launched the iPhone X – the first iPhone users could unlock with FaceID – Apple’s marketing term for facial recognition.



*Figure 7 (3-1 face rec his.)*

**3.2 Introduction of Flutter**

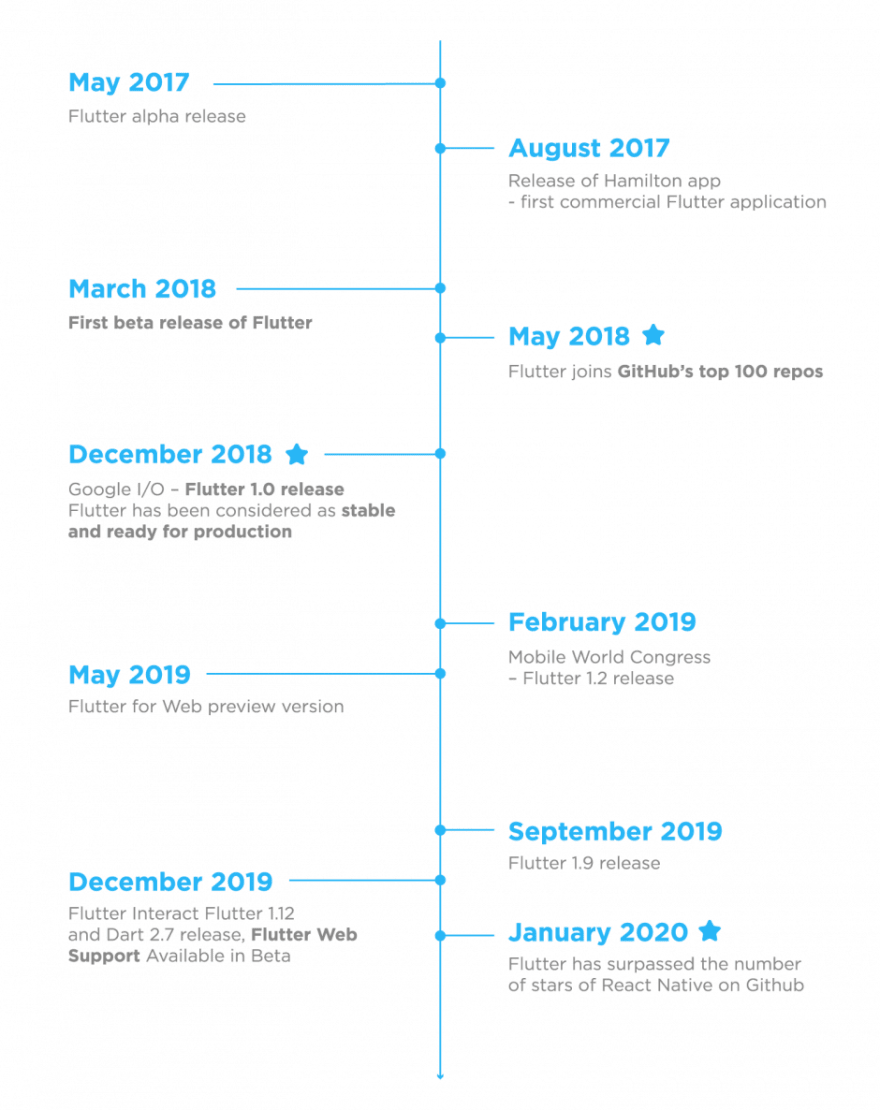
The first version of Flutter was known as "Sky" and ran on the Android operating system. It was unveiled at the 2015 Dart developer summit with the stated intent of being able to render consistently at 120 frames per second. During the keynote of Google Developer Days in Shanghai in September 2018, Google announced Flutter Release Preview 2, the last major release before Flutter 1.0. On December 4th of that year, Flutter 1.0 was released at the Flutter Live event, denoting the first stable version of the framework. On December 11, 2019, Flutter 1.12 was released at the Flutter Interactive event.

On May 6, 2020, the Dart software development kit (SDK) version 2.8 and Flutter 1.17.0 were released, adding support for the Metal API which improves performance on iOS devices by approximately 50%, as well as new Material widgets and network tracking development tools.

On March 3, 2021, Google released Flutter 2 during an online Flutter Engage event. This major update brought official support for web-based applications with a new CanvasKit renderer and web specific widgets, early-access desktop application support for Windows, macOS, and Linux and improved Add-to-App APIs. This release also utilized Dart 2.0 that featured sound null-safety, which caused many breaking changes and issues with many external packages; however, the Flutter team included instructions and tools to mitigate these issues.

On September 8th, 2021, Dart 2.14 and Flutter 2.5 were released by Google. The update brought improvements to the Android full-screen mode and the latest version of Google's Material Design called Material You. Dart received two new updates, standardizing lint conditions and marking support for Apple Silicon as stable.

The current stable channel of Flutter is 2.16.1 and the Dart version is 2.16.1.



*Figure 8 (3-2 flutter his.)*

**3.2.1 What is Flutter?**



*Figure 9 (3-3 flutter)*

Flutter is a free and open-source mobile UI framework created by Google and released in May 2017.

**3.2.2 Why did we choose Flutter?**

In a few words, it allows you to create a native mobile application with only one codebase.

This means that you can use one programming language and one codebase to create two different apps (for iOS and Android).

**3.2.3 Flutter consists of two important parts :**

**1- An SDK (Software Development Kit):**

A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).

**2- A Framework (UI Library based on widgets):**

A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

**3.3 About Dart**



*Figure 10 (3-4 Dart)*

To develop with Flutter, you will use a programming language called Dart. The language was created by Google in October 2011, but it has improved a lot over these past years. Dart focuses on front-end development, and you can use it to create mobile and web applications.

**3.3.1 History of Dart**

Dart was unveiled at the GOTO conference in [Aarhus](https://en.wikipedia.org/wiki/Aarhus), Denmark, October 10–12, 2011. The project was founded by [Lars Bak](https://en.wikipedia.org/wiki/Lars_Bak_(computer_programmer)) and Kasper Lund.[[13]](https://en.wikipedia.org/wiki/Dart_(programming_language)?fbclid=IwAR16TMPBgfi8eMd2x6BkQwEH6P7n3X6brUJfCxgnRtjt-Ce1uND2-jBd1no#cite_note-13) Dart 1.0 was released on November 14, 2013.

Dart initially had a mixed reception and the Dart initiative has been criticized by some for fragmenting the web, due to the original plans to include a Dart VM in Chrome. Those plans were dropped in 2015 with the 1.9 release of Dart to focus instead on compiling Dart to JavaScript.

Dart 2.0 was released in August 2018, with language changes including a sound type system.

Dart 2.6 introduced a new extension, dart2native, which extends native compilation to the Linux, macOS, and Windows desktop platforms. Earlier developers could create new tools using only Android or iOS devices. With this extension it also becomes possible to compose a program into self-contained executables. According to company representatives, it’s no longer necessary to have the Dart SDK installed, as the self-contained executables can now start running in a few seconds. The new extension is also integrated with the [Flutter](https://en.wikipedia.org/wiki/Flutter_(software)) toolkit, making it possible to use the compiler on small services (for example, backend support).

**Standardization**

[Ecma International](https://en.wikipedia.org/wiki/Ecma_International) has formed technical committee TC52 to work on standardizing Dart, and inasmuch as Dart can be compiled to standard JavaScript, it works effectively in any modern browser. EI approved the first version of the Dart language specification in July 2014 at its 107th General Assembly, and a second edition in December 2014.The latest specification is [available here](https://dart.dev/guides/language/spec)

**3.4 History of Firebase**

Firebase evolved from Envolve, a prior startup founded by James Tamplin and Andrew Lee in 2011. Envolve provided developers an API that enables the integration of online chat functionality into their websites. After releasing the chat service, Tamplin and Lee found that it was being used to pass application data that were not chat messages. Developers were using Envolve to sync application data such as game state in real time across their users. Tamplin and Lee decided to separate the chat system and the real-time architecture that powered it. They founded Firebase as a separate company in 2011 and it launched to the public in April 2012.

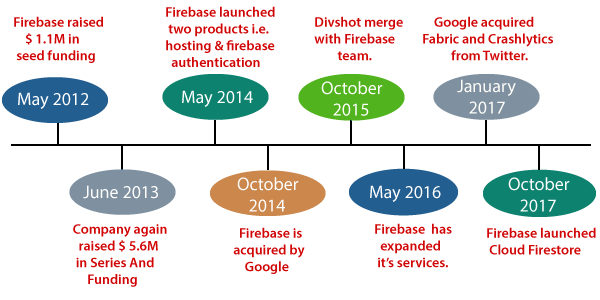
Firebase's first product was the Firebase Realtime Database, an API that synchronizes application data across iOS, Android, and Web devices, and stores it on Firebase's cloud. The product assists software developers in building real-time, collaborative applications.

In May 2012, a month after the beta launch, Firebase raised $1.1 million in [seed funding](https://en.wikipedia.org/wiki/Seed_funding) from [venture capitalists](https://en.wikipedia.org/wiki/Venture_capital) [Flybridge Capital Partners](https://en.wikipedia.org/wiki/Flybridge_Capital_Partners), [Greylock Partners](https://en.wikipedia.org/wiki/Greylock_Partners), [Founder Collective](https://en.wikipedia.org/w/index.php?title=Founder_Collective&action=edit&redlink=1), and [New Enterprise Associates](https://en.wikipedia.org/wiki/New_Enterprise_Associates).[[4]](https://en.wikipedia.org/wiki/Firebase#cite_note-4) In June 2013, the company further raised $5.6 million in [Series A funding](https://en.wikipedia.org/wiki/Series_A_funding) from [Union Square Ventures](https://en.wikipedia.org/wiki/Union_Square_Ventures) and [Flybridge Capital Partners](https://en.wikipedia.org/wiki/Flybridge_Capital_Partners).

In 2014, Firebase launched two products. Firebase Hosting and Firebase Authentication. This positioned the company as a [mobile backend as a service](https://en.wikipedia.org/wiki/Mobile_backend_as_a_service), Firebase was [acquired by Google](https://en.wikipedia.org/wiki/List_of_mergers_and_acquisitions_by_Alphabet). A year later, in October 2015, Google acquired Divshot, an [HTML5](https://en.wikipedia.org/wiki/HTML5) web-hosting platform, to merge it with the Firebase team.

In July 2016, Google announced that it was acquiring the mobile developer platform LaunchKit, which specialized in app developer marketing, and would be folding it into the Firebase Growth Tools team. In January 2017, Google acquired Fabric and [Crashlytics](https://en.wikipedia.org/wiki/Crashlytics) from [Twitter](https://en.wikipedia.org/wiki/Twitter,_Inc.) to add those services to Firebase.

In October 2017, Firebase launched Cloud Firestore, a real-time [document database](https://en.wikipedia.org/wiki/Document-oriented_database) as the successor product to the original Firebase Realtime Database.



*Figure 11 (3-5 firebase his.)*

**3.5 Class of Raspberry Pi**

**The first generation (Raspberry Pi Model B):**

was released in February 2012, followed by the simpler and cheaper Model A.

In 2014, the Foundation released a board with an improved design, Raspberry Pi Model B+. These first generation boards feature ARM11 processors, are approximately credit-card sized and represent the standard mainline form-factor. Improved A+ and B+ models were released a year later.[clarification needed] A "Compute Module" was released in April 2014 for embedded applications.

The Raspberry Pi 2 was released in February 2015 and initially featured a 900 MHz 32-bit quad-core ARM Cortex-A7 processor with 1 GB RAM. Revision 1.2 featured a 900 MHz 64-bit quad-core ARM Cortex-A53 processor (the same as that in the Raspberry Pi 3 Model B, but underclocked to 900 MHz).

**Raspberry Pi 3 Model B:**

was released in February 2016 with a 1.2 GHz 64-bit quad core ARM Cortex-A53 processor, on-board 802.11n Wi-Fi, Bluetooth and USB boot capabilities.

On Pi Day 2018, the Raspberry Pi 3 Model B+ was launched with a faster 1.4 GHz processor, a three-times faster gigabit Ethernet (throughput limited to ca. 300 Mbit/s by the internal USB 2.0 connection), and 2.4 / 5 GHz dual-band 802.11ac Wi-Fi (100 Mbit/s). Other features are Power over Ethernet (PoE) (with the add-on PoE HAT), USB boot and network boot (an SD card is no longer required).

**Raspberry Pi 4 Model B:**

A picture containing electronics, circuit

Description automatically generated

*Figure 12 (3-6 Raspberry pi 4.)*

was released in June 2019 with a 1.5 GHz 64-bit quad core ARM Cortex-A72 processor, on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet (throughput not limited), two USB 2.0 ports, two USB 3.0 ports, 1–8 GB of RAM, and dual-monitor support via a pair of micro HDMI (HDMI Type D) ports for up to 4K resolution. The version with 1 GB RAM has been abandoned and the prices of the 2 GB version have been reduced. The 8 GB version has a revised circuit board.

The Pi 4 is also powered via a USB-C port, enabling additional power to be provided to downstream peripherals, when used with an appropriate PSU. But the Pi can only be operated with 5 volts and not 9 or 12 volts like other minicomputers of this class. The initial Raspberry Pi 4 board has a design flaw where third-party e-marked USB cables, such as those used on Apple MacBooks, incorrectly identify it and refuse to provide power. Tom's Hardware tested 14 different cables and found that 11 of them turned on and powered the Pi without issue. The design flaw was fixed in revision 1.2 of the board, released in late 2019.

In mid-2021, Pi 4 B models appeared with the improved Broadcom BCM2711C0. The manufacturer is now using this chip for the Pi 4 B and Pi 400.

**3.6 Infrared Temperature Sensors – Thermopiles**

Infrared (IR) temperature sensors enable accurate non-contact temperature measurement in medical applications. The most common applications for this type of temperature sensor is measuring ear temperature, forehead temperature, or skin temperature. The sensing element is composed of multiple thermocouples on a silicon chip to measure an object's infrared energy. TE packages and customizes thermopiles in various package sizes and with different wire lengths to accommodate customer needs.

**This project uses mlx90614 infrared temperature sensor, which specification as follows:**

The MLX90614 is an infrared thermometer for non-contact temperature measurements. Both the IR sensitive thermopile detector chip and the signal conditioning ASIC are integrated in the same TO-39 can. Integrated into the MLX90614 are a low noise amplifier, 17-bit ADC and powerful DSP unit thus achieving high accuracy and resolution of the thermometer.

In the next chapter we will show what our system consist of and the main features we use in our main parts of project

**Chapter 4**

**System’s Structure**

**4.1 Project’s System:**

Our system is contains 2 main parts, first part hardware and the second part is software.

Diagram

Description automatically generated

*Figure 13 (4-1 Project structure.)*

* Hardware consists of 3 items :

3-**Camera module for raspberry pi:** it will

take a photo and work as a live camera to take

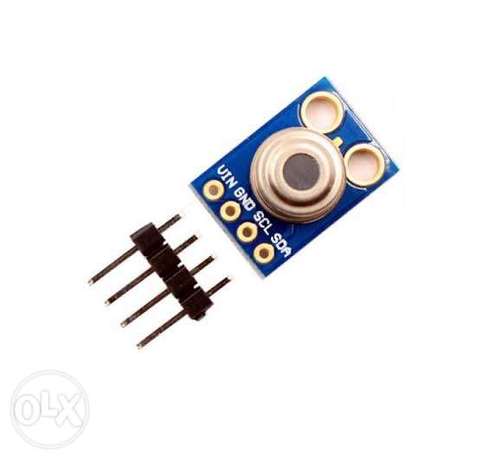
the attendance for everyone

2- **MLX90614:** it is temperature sensor that will detect

if the member temperature is normal or not

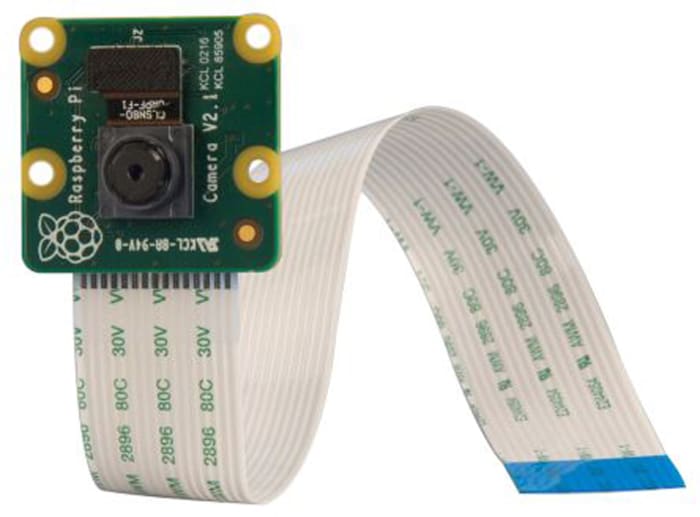
1- **Raspberry pi 4** : the main bored in this project that

will control and connect the other component



*Figure 14 (4-2 Raspberry pi 4.)*

*Figure 15 (4-3 MLX90614.)*



*Figure 16 (4-4 camera module.)*

* **Raspberry pi:**

Diagram

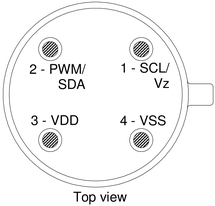
Description automatically generated

*Figure 17 (4-5 Raspberry pi 4 tech specs.)*

The main element in the hardware components is raspberry pi but why? Because we make 3 implementations on it:

1. temperature sensor
2. camera module
3. the face recognition algorism

**MLX9614**:



*Figure 18 (4-6 MlX9614 datasheet)*

First the sensor takes the temperature of the member in this state we have 2 conditions:

1. normal temperature then it allows member to enter and take his/her attendance
2. up normal temperature then it sets alarm with the buzzer

Flow Chart for MLX9614 in our system:

Diagram

Description automatically generated

*Figure 17 (4-7 flowchart of the sensor in our system)*

**Camera module:**

High definition camera module compatible with the Raspberry Pi model A and model B. Provides high sensitivity, low crosstalk and low noise image capture in an ultra small and lightweight design. The camera module connects to the Raspberry Pi board via the CSI connector designed specifically for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the BCM2835 processor.

**Image Sensor:**

Omni Vision 5647 CMOS image sensor in a fixed-focus module with integral IR filter

**Resolution:**

5-megapixel Still picture resolution 2592 x 1944 Max image transfer rate 1080p: 30fps (encode and decode) 720p: 60fps

**Connection to Raspberry Pi:**

15 Pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2)

**4.3 Analysis Of Face Recognition:**

In Chapter 1, we introduced the facial recognition, discussed the use case and bright future of this technology. A tremendous amount of research and effort from many major company and universities and been dedicated to this field. In the first part of this chapter, we will review the most significant work in the facial recognition field. A screenshot of a computer

Description automatically generated with low confidence

*Figure 18 (4-8 face recognition algorithm lifecycle)*

**4.2.1 Face detection**

The first step in our pipeline is face detection*.* Obviously, we need to locate the faces in a photograph before we can try to tell them apart!

Face detection is a great feature for cameras. When the camera can automatically pick out faces, it can make sure that all the faces are in focus before it takes the picture. But we’ll use it for a different purpose — finding the areas of the image we want to pass on to the next step in our pipeline.

Face detection went mainstream in the early 2000's when Paul Viola and Michael Jones invented a way to detect faces that was fast enough to run on cheap cameras. However, much more reliable solutions exist now. We’re going to use a method invented in 2005 called Histogram of Oriented Gradients — or just **HOG** for short.

To find faces in an image, we’ll start by making our image black and white because we don’t need color data to find faces:

A person with a beard

Description automatically generated with low confidence

*Figure 19 (4-9 experiment black and white image for Mohamed salah)*

Then we’ll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it.

Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker.

If you repeat that process for **every single pixel** in the image, you end up with every pixel being replaced by an arrow. These arrows are called gradients and they show the flow from light to dark across the entire image.

This might seem like a random thing to do, but there’s a really good reason for replacing the pixels with gradients. If we analyze pixels directly, really dark images and really light images of the same person will have totally different pixel values. But by only considering the direction that brightness changes, both really dark images and really bright images will end up with the same exact representation. That makes the problem a lot easier to solve!

But saving the gradient for every single pixel gives us way too much detail.

We end up missing the forest for the trees. It would be better if we could just see the basic flow of lightness/darkness at a higher level so we could see the basic pattern of the image.

To do this, we’ll break up the image into small squares of 16x16 pixels each. In each square, we’ll count how many gradients point in each major direction (how many points up, point up-right, point right, etc.…). Then we’ll replace that square in the image with the arrow directions that were the strongest.

The end result is we turn the original image into a very simple representation that captures the basic structure of a face in a simple way.

To find faces in this HOG image, all we have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces.

Using this technique, we can now easily find faces in any image:

A picture containing hairpiece, person, smiling, jacket

Description automatically generated

*Figure 20 (4-10 face detection for Mohamed salah)*

**4.2.2 Features Extraction**

we isolated the faces in our image. But now we have to deal with the problem that faces turned different directions look totally different to a computer:

A person smiling for the camera

Description automatically generated with medium confidence 

*Figure 21 (4-11 different face sides of images for Mohamed salah)*

To account for this, we will try to warp each picture so that the eyes and lips are always in the sample place in the image. This will make it a lot easier for us to compare faces in the next steps.

To do this, we are going to use an algorithm called face landmark estimation. There are lots of ways to do this, but we are going to use the approach .Invented in 2014 by Vahid Kazemi and Josephine Sullivan.

The basic idea is we will come up with 68 specific points (called *landmarks*) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face:

Chart, scatter chart

Description automatically generated

*Figurer 22 (4-12 different specific points of any face)*

The 68 landmarks we will locate on every face. This image was created by Brandon Amos of CMU who works Open Face.

Now that we know where the eyes and mouth are, we’ll simply rotate, scale, and shear the image so that the eyes and mouth are centered as best as possible. We won’t do any fancy 3d warps because that would introduce distortions into the image. We are only going to use basic image transformations like rotation and scale that preserve parallel lines (called affine transformation):

A picture containing diagram

Description automatically generated

*Figure 23 (4-13 experiment black and white image for Mohamed salah)*

Now no matter how the face is turned, we are able to center the eyes and mouth are in roughly the same position in the image. This will make our next step a lot more accurate.

**4.2.3 Encoding Faces**

Now we are to the meat of the problem — actually telling faces apart. This is where things get really interesting!

The simplest approach to face recognition is to directly compare the unknown face we found in Step 2 with all the pictures we have of people that have already been tagged. When we find a previously tagged face that looks very similar to our unknown face, it must be the same person. Seems like a pretty good idea, right?

There’s actually a huge problem with that approach. A site like Facebook with billions of users and a trillion photos can’t possibly loop through every previous-tagged face to compare it to every newly uploaded picture. That would take way too long. They need to be able to recognize faces in milliseconds, not hours.

What we need is a way to extract a few basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements. For example, we might measure the size of each ear, the spacing between the eyes, the length of the nose, etc. The most reliable way to measure a face.

Ok, so which measurements should we collect from each face to build our known face database? Ear size? Nose length? Eye color? Something else?

It turns out that the measurements that seem obvious to us humans (like eye color) don’t really make sense to a computer looking at individual pixels in an image. Researchers have discovered that the most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure.

The solution is to train a Deep Convolutional Neural Net. But instead of training the network to recognize pictures objects like we did last time, we are going to train it to generate 128 measurements for each face.

The training process works by looking at 3 face images at a time:

1. Load a training face image of a known person
2. Load another picture of the same known person
3. Load a picture of a totally different person

Then the algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for 1 and 2 are slightly closer while making sure the measurements for 2 and 3 are slightly further apart:

A single ‘triplet’ training step:

Diagram

Description automatically generated

*Figure 24 (4-14 how face recognition work)*

After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person. Any ten different pictures of the same person should give roughly the same measurements.

Machine learning people call the 128 measurements of each face an embedding. The idea of reducing complicated raw data like a picture into a list of computer-generated numbers comes up a lot in machine learning (especially in language translation).

The exact approach for faces we are using we invented in 2015 by researchers at Google but many similar approaches exist.

This process of training a convolutional neural network to output face embeddings requires a lot of data and computer power. Even with an expensive NVidia Telsa video card, it takes about 24 hours of continuous training to get good accuracy.

But once the network has been trained, it can generate measurements for any face, even ones it has never seen before! So, this step only needs to be done once. Lucky for us, the fine folks at OpenFace already did this and they published several trained networks which we can directly use. Thanks Brandon Amos and team!

Table

Description automatically generated So, all we need to do ourselves is run our face images through their pre-trained network to get the 128 measurements for each face. Here’s the measurements for our test image:

A person smiling for the camera

Description automatically generated with medium confidence So, what parts of the face are these 128 numbers measuring exactly? It turns out that we have no idea. It doesn’t really matter to us. All that we care is that the network generates

*Figure 25 (4-15 128 measurements generated from image for Mohamed salah)*

nearly the same numbers when looking at two different pictures of the same person.

**4.2.4 Face Recognition:**

This last step is actually the easiest step in the whole process. All we have to do is find the person in our database of known people who has the closest measurements to our test image.

You can do that by using any basic machine learning classification algorithm. No fancy deep learning tricks are needed. We’ll use a simple linear SVM classifier, but lots of classification algorithms could work.

Diagram

Description automatically generated

*Figure 25 (4-15 flowchart for SVM)*

All we need to do is train a classifier that can take in the measurements from a new test image and tells which known person is the closest match. Running this classifier takes milliseconds. The result of the classifier is the name of the person!

**4.2.5 Attendance**

After the previous step, Now this person is recognize for the system then the system send his name and his arrival time to the firebase.

**4.3 Mobile application:**

We make a mobile application by flutter to make it easy for the employees to track them attendance by it and like we mention in Chapter 2 we choose flutter to make the application work in more than 1 platform.

**4.3.1 Flowchart of the project**

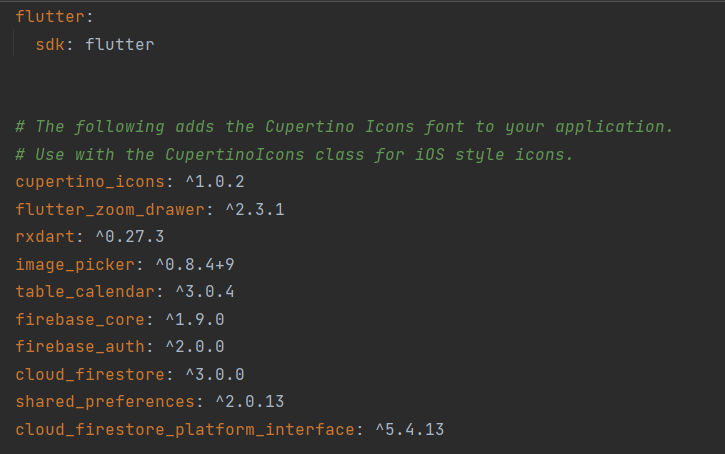
**In figure 4.5, the flowchart of the project will be illustrated.**

**Diagram

Description automatically generated**

*Figure 26 (4-16 flowchart for Mobile APP.)*

**4.3.2 Flutter Plugins**

**It will be shown the plugins have been used in the project as in figure 4.**

*Figure 26 (4-16 plugins we used in flutter code)*

1. **cupertino\_icons: ^1.0.2**

This is an asset repo containing the default set of icon assets used by Flutter's Cupertino widgets.

1. **flutter\_zoom\_drawer: ^2.3.1**

A Flutter package with custom implementation of the Side Menu (Drawer)

Features

* Simple sliding drawer
* Sliding drawer with shadows
* Sliding drawer with rotation
* drawer with rotation and shadows Support for both LTR & RTL

1. **rxdart ^0.27.3:**

RxDart extends the capabilities of Dart Streams and Stream Controllers. Dart comes with a very decent Streams API out-of-the-box; rather than attempting to provide an alternative to this API, RxDart adds functionality from the reactive extensions specification on top of it. RxDart does not provide its Observable class as a replacement for Dart Streams. Instead, it offers several additional Stream classes, operators (extension methods on the Stream class), and Subjects.

1. **image\_picker: ^0.8.5+3**

A Flutter plugin for iOS and Android for picking images from the image library and taking new pictures with the camera.

1. **table\_calendar: ^3.0.5**

Highly customizable, feature-packed calendar widget for Flutter.

1. **shared\_preferences: ^2.0.15**

Wraps platform-specific persistent storage for simple data (NSUserDefaults on iOS and macOS, SharedPreferences on Android, etc.). Data may be persisted to disk asynchronously, and there is no guarantee that writes will be persisted to disk after returning, so this plugin must not be used for storing critical data. Supported data types are int, double, bool, String and List<String>.

1. **flutter\_bloc: ^8.0.1**

Widgets that make it easy to integrate blocs and cubits into Flutter

**Features**:

* Extensive, yet easy to use API
* Preconfigured UI with customizable styling
* Custom selective builders for unlimited UI design
* Locale support
* Range selection support
* Multiple selection support
* Dynamic events and holidays
* Vertical auto sizing - fit the content, or fill the viewport
* Multiple calendar formats (month, two weeks, week)
* Horizontal swipe boundaries (first day, last day)

**4.4 Firebase**



*Figure 27 (4-17 firebase logo)*

Firebase is the live(Real-time) database that will connect the application with the system

,it is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google’s infrastructure. Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

In Firebase, a document is a set of key-value pairs defined by a schema. A group of documents makes up a collection Key

**Features of firebase**

* Authentication:

It supports authentication using passwords, phone numbers, Google, Facebook, Twitter, and more. The Firebase Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app.

* Realtime database:

Data is synced across all clients in realtime and remains available even when an app goes offline.

* Hosting:

Firebase Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

* Test lab:

The application is tested on virtual and physical devices located in Google’s data centers.

* Notifications:

Notifications can be sent with firebase with no additional coding. Users can get started with firebase for free; more details can be found on the official website.

To connect firebase with flutter you need to implement some plugins in your flutter code

* firebase\_core: ^1.9.0:

A Flutter plugin to use the Firebase Core API, which enables connecting to multiple Firebase apps.

* firebase\_auth: ^3.3.18:

A Flutter plugin to use the Firebase Authentication API.

* cloud\_firestore: ^3.0.0:

A Flutter plugin to use the Cloud Firestore API.

To connect firebase with python you need to implement some libraries in your python code:

from firebase import firebase

firebase = firebase.FirebaseApplication()

result = firebase.get()

The library provides all the corresponding methods for those actions in both synchronous and asynchronous manner. You can just start an asynchronous GET request with your callback function, and the method

In the next chapter will talk about the results of our project and the scenario about how it works after the project it finished

**Chapter 5**

**Project’s Experiment**

**5.1 Experiment 1**

**This is an Experiment of how Face recognition system works in without live camera:**

**A person smiling for the camera

Description automatically generated with medium confidenceStep 1:** we take a training photo of Mohammed Salah as a reference to be uploaded and stored on the system.

*Figure 28 (5-1 training photo of Mohammed Salah)*

A person wearing a helmet

Description automatically generated with medium confidence**Step 2:** Running this classifier takes milliseconds to find result, differentiate and find the matches between the training photo and test photo.

*Figure 29 (5-2 test photo of Mohammed Salah)*

**Step 3:** Blurry Image Here we used another test photo of Mohamed Salah but blurry or pixeled and the result came as true.

A picture containing text, person, person, male

Description automatically generated

*Figure 30 (5-3 test Blurry Image of Mohammed Salah)*

**Step 4:** Different faces in one Image We put another photo with another player and the system detected Mo Salah’s face and can recognized him successfully.

**A picture containing text, person, person, sport

Description automatically generated**

*Figure 31 (5-4 test Different faces* *of Mohammed Salah)*

**Step 5:** Close features last, we added a photo of Nasr Salah who is Salah’s brother as the have very close face features and the result came as false.

A picture containing text, grass, outdoor, person

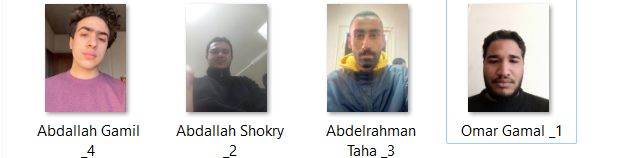
Description automatically generated

*Figure 32 (5-5 test photo of* *Nasr Salah)*

**5.2 Experiment 2**

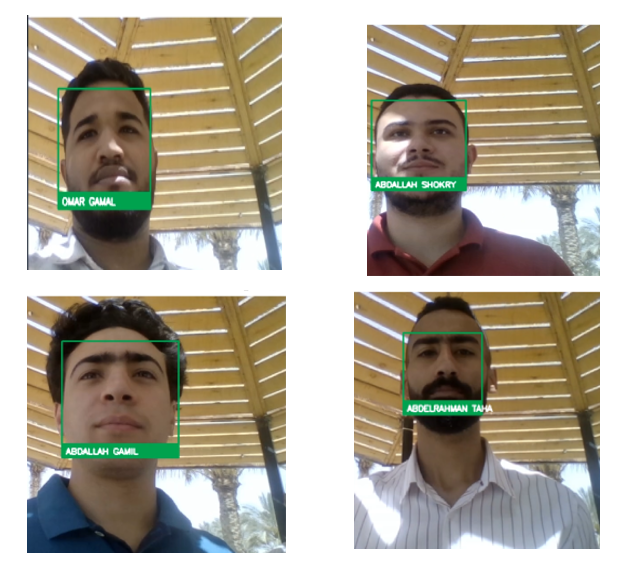
**This is an Experiment of how Face recognition system works in with live camera:**

**Step 1:** we put our dataset of our photos in local database.



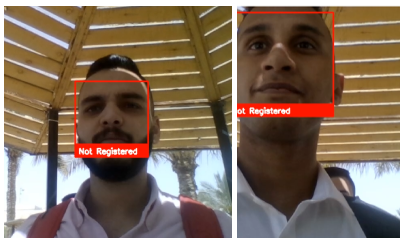
*Figure 33 (5-6 Training photos)*

**Step 2:** now we start our system to test the ability of it to recognize our team



*Figure 34 (5-7 Result of testing)*

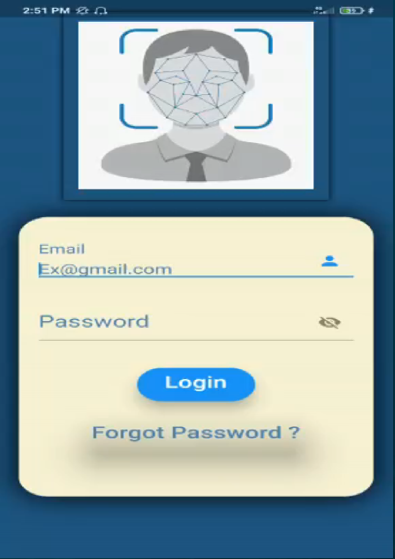
**Step 3:** now test of users not in the system



*Figure 35 (5-8 Result of testing)*

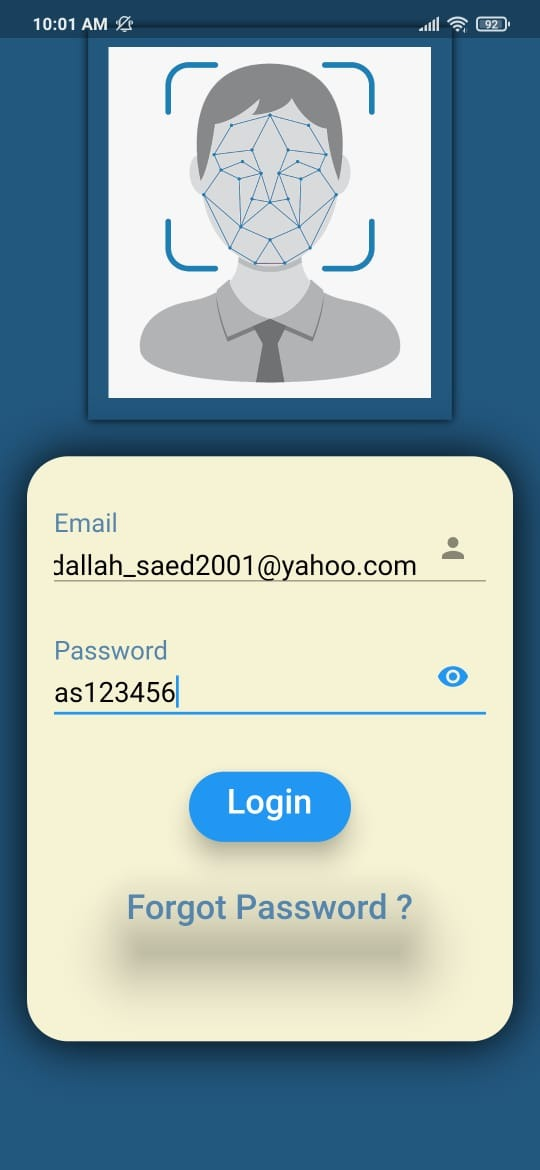
**5.3 Experiment for mobile**

1. Main page (login page): this is our home page you as a member can login by it and there is an option to choose to login as admin. And it also contains an option for who forget his password

****

*Figure 36 (5-9 main screen)*

1.1 **Member Page**: it is contained what member can do in this app which is 4 things



*Figure 37 (5–10 Member login screen)*

- track his attendance in this month

Background pattern

Description automatically generated with medium confidence

*Figure 38 (5–11 Member track attendance)*

- Track his attendance over the year to know his holidays Left in his year

Graphical user interface, application

Description automatically generated

*Figure 38 (5–12 request holiday)*

- show if he has salary deductions because He came late for example

Text, letter

Description automatically generated

*Figure 39 (5–12 request holiday)*

* Graphical user interface, application

  Description automatically generatedChange password

*Figure 40 (5–13 Change Password)*

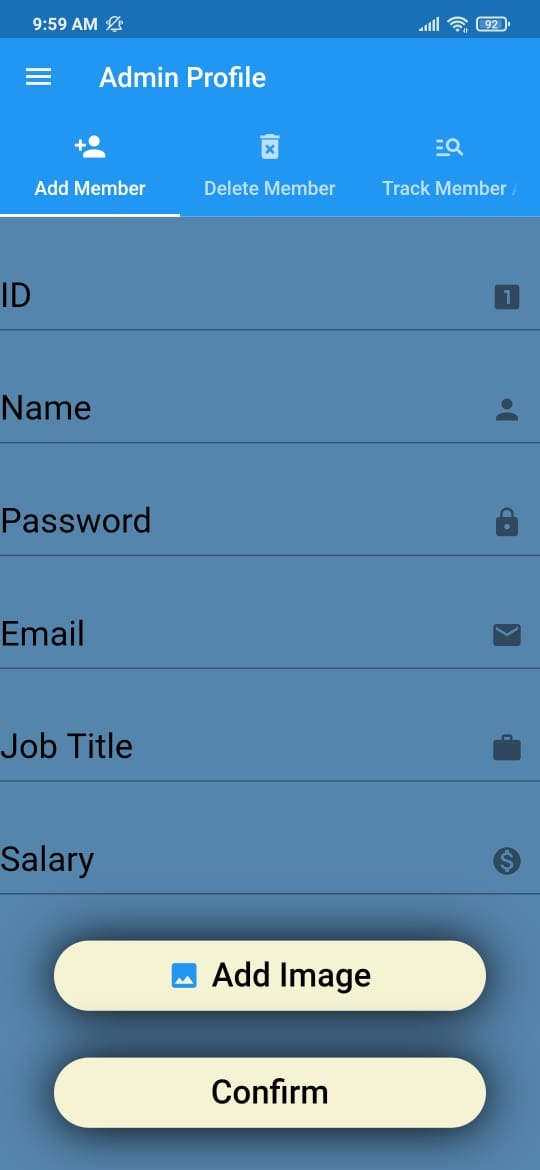
1.2 **Admin page:**

Graphical user interface, application

Description automatically generated

*Figure 41 (5–14 Admin login screen)*

-Add member



*Figure 41 (5–14 Admin add member)*

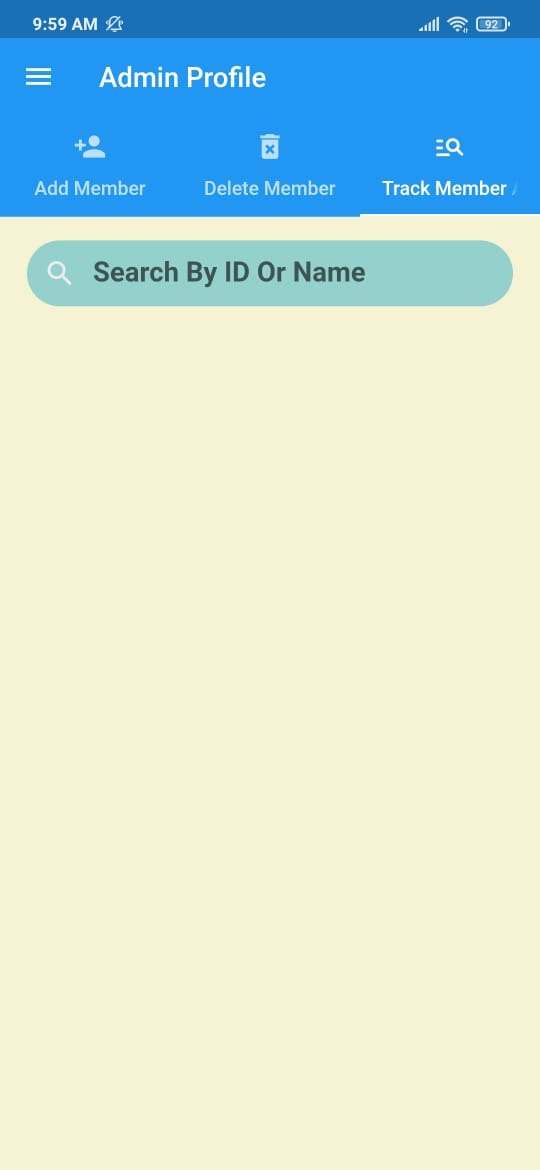
* Delete member

Graphical user interface, text, application, chat or text message

Description automatically generated

*Figure 42 (5–15 Admin delete member)*

* Track all member attendance.



*Figure 43 (5–16 Admin search member)*

2.1 Reset Password

Graphical user interface, text, application, chat or text message

Description automatically generated

*Figure 44 (5–17 Reset password)*

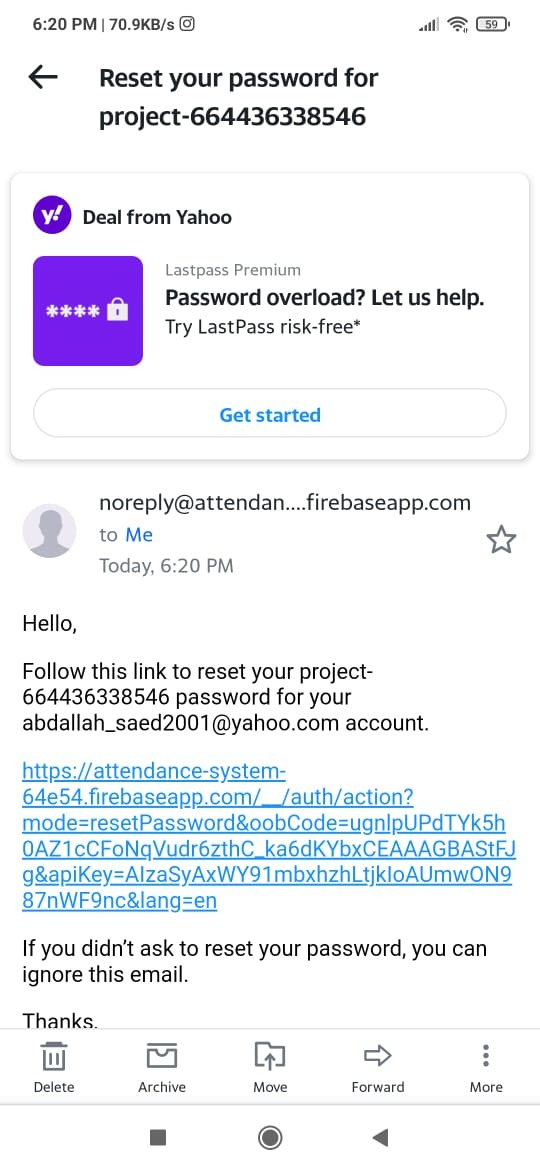
* Send reset password to email

Graphical user interface, text, application, chat or text message

Description automatically generated

*Figure 45 (5–18 Reset password with email)*

* Firebase send email to change password



*Figure 45 (5–18 Email from firebase)*

* After clicking the email

Graphical user interface, application, Teams

Description automatically generated

*Figure 46 (5–19 Reset password from email)*

* Confirmation email

A picture containing background pattern

Description automatically generated

*Figure 46 (5–19 Confirmation email)*

**Chapter 6**

**Conclusion**

**and**

**Future Work**

6.1 **project summary**

our project is contain from 3 main axis: first the hardware axis in this part we make a temperature detect device to check the temperature for all people want to enter the organize if the temperature of them in normal case then they will move to the next step of our system, if the temperature is up normal then them attendance will not take to the system.

Another use for the hardware component is to implement the face recognition algorithm for make us replace the pc.

The second axis is mobile application it make the adding new member and delate member easier for the administrator with access to track all member attendance, and another usage for the mobile application is to make the workers track them attendance and request a day-off and the last usage for mobile app is make a workers show the details of them salary

The third axis is the database we make it by firebase to connect the hardware axis with mobile application to connect the other axis with themselves

* 1. **conclusion**

This system aims to build an effective class attendance system using face recognition techniques. The proposed system will be able to mark the attendance via face Id. It will detect faces via webcam and then recognize the faces. After recognition, it will mark the attendance of the recognized student and update the attendance record.

**6.3 future work**

**References:**

* Nilsson, Nils J. (2009). The Quest for Artificial Intelligence. Cambridge University Press.
* de Leeuw, Karl; Bergstra, Jan (2007). *The History of Information Security: A Comprehensive Handbook*. Elsevier.
* ["Flutter: the first UI platform designed for ambient computing"](https://developers.googleblog.com/2019/12/flutter-ui-ambient-computing.html). Flutter blog.
* [Flutter documentation | Flutter](https://docs.flutter.dev/)
* [*"Version 2 of Google's Flutter toolkit adds support for desktop and web apps"*](https://social.techcrunch.com/2021/03/03/version-2-of-googles-flutter-toolkit-adds-support-for-desktop-and-web-apps/). TechCrunch*.*
* [*"Dart 1.0: A stable SDK for structured web apps"*](https://news.dartlang.org/2013/11/dart-10-stable-sdk-for-structured-web.html). news.dartlang.org*.*
* [Dart documentation | Dart](https://dart.dev/guides)