

Smart safety helmet



Faculty of computers and information

Mohamed Ahmed Abd Elghany

Mohamed Ahmed Asker

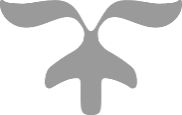
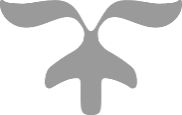
Rahma Mohamed Ebrahim Salem

Habiba Essam Mohamed Kamal Abd ElDayem

Alaa Mohamed Abdelsatar Elattar

Fatma Hosafi Saad Ibrahim Ageba

Amr Sabry Abd Elmoneim Amr



2023 / 2024



**Graduation Project - English Abstract**

|  |  |
| --- | --- |
| **Project Code:** |  |
| **Project Title (in English):** | Smart Safety Helmet |
| **Project Title (in Arabic):** | خوذه السلامة الذكية |
| **Scientific Department:** | IS / IT |
| **Supervisor(s):** | DR / walaa mohamed |
| **Supervisor Assistant** | DR / shimaa sami |
| **Project Team:** | 1- Mohamed Ahmed Abd-elghany  2- Mohamed Ahmed Asker  3- Rahma Mohamed Ebrahim Salem  4- Habiba Essam Mohamed Kamal Abd ElDayem  5- Alaa Mohamed Abdelsatar Elattar  6- Fatma Hosafi Saad Ibrahim Ageba  7- Amr Sabry Abd Elmoneim Amr |

**Project Abstract**

This project summary is to manufacture a smart helmet for workers on hazard construction sites with several features, including tracking the worker location , dropping detection, measuring temperature and humidity in the air, objects detection, and checking gases that can cause ignition in the atmosphere. The data of Helmets will be collected and fetched on our system (Website / mobile application) which managers and supervisors can check that statistics and handle any danger could happen . The project must be implemented to maintain the safety of workers and reduce the incidence of accidents and injuries. Providing the advantage of monitoring workers remotely for supervisors or site mentors. The project does not require external parties or official communications.





الملخص العربي لمشروع التخرج

|  |  |
| --- | --- |
| **كود المشروع:** |  |
| **عنوان المشروع باللغة العربية:** | خوذة السلامة الذكية |
| **عنوان المشروع باللغة الإنجليزية:** | **Smart Safety Helmet** |
| **القسم العلمي:** | **IS / IT** |
| **اسم المشرف على المشروع:** | **د / ولاء محمد** |
| **اسم مساعد المشرف** | **د / شيماء سامى** |
| **أسماء فريق العمل من الطلاب:** | **-1 محمد احمد محمد عبد الغنى**  **-2محمد احمد عسكر**  **-3رحمة محمد ابراهيم سالم**  **-4حبيبة عصام محمد كمال عبدالدايم**  **-5الاء محمد عبدالستار العطار**  **-6فاطمه حصافى سعد ابراهيم عجيبه**  **-7عمرو صبري عبدالمنعم عمرو** |

**ملخص المشروع**

ملخص المشروع هذا هو تصنيع خوذة ذكية للعاملين في مواقع البناء الخطرة مع عدة ميزات، بما في ذلك تتبع موقع العامل، واكتشاف السقوط، وقياس درجة الحرارة والرطوبة في الهواء، واكتشاف الأشياء، وفحص الغازات التي يمكن أن تسبب الاشتعال في الغلاف الجوي. ويجب تنفيذ المشروع للحفاظ على سلامة العاملين وتقليل وقوع الحوادث والإصابات. توفير ميزة مراقبة العاملين عن بعد للمشرفين أو مرشدي الموقع



نموذج وضع درجات أعمال السنة لمشروع التخرج (من قبل المشرف)

|  |  |
| --- | --- |
| **اسم المشروع باللغة الانجليزية** | **Smart Safety Helmet** |
| **القسم العلمي** | **○** Computer Science **○** InformationSystems **○** Information technology **○** Multimedia |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **م** | **اسم الطالب** | **مدى التزام الطالب بحضور اللقاءات والمتابعة المستمرة مـع المـشرف** | **المشاركة في النقاشات والاجابة على الأسئلة المطروحة اثناء اللقاءات الدورية** | **استيعاب وتمكن الطالب من الأدوات التي استخدمت في تطوير المشروع** | **قدرة الطالب على تنفيذ المهام المكلف بها** | **قدرة الطالب على العمل ضمن فريق عمل** | **اجمالي درجات أعمال السنة** |
| **10 درجة** | **10 درجة** | **10درجة** | **20 درجة** | **10 درجة** | **60 درجة** |
| 1 | محمد احمد محمد عبد الغنى |  |  |  |  |  |  |
| 2 | محمد احمد عسكر |  |  |  |  |  |  |
| 3 | رحمة محمد ابراهيم سالم |  |  |  |  |  |  |
| 4 | حبيبة عصام محمد كمال عبدالدايم |  |  |  |  |  |  |
| 5 | الاء محمد عبدالستار العطار |  |  |  |  |  |  |
| 6 | فاطمه حصافى سعد ابراهيم عجيبه |  |  |  |  |  |  |
| 7 | عمرو صبري عبدالمنعم عمرو |  |  |  |  |  |  |

**المشرف على المشروع وكيل الكلية لشئون التعليم والطلاب عميد الكلية**

نموذج تقييم ومناقشة مشروع التخرج

|  |  |
| --- | --- |
| **اسم المشروع باللغة الانجليزية** | **Smart Safety Helmet** |
| **القسم العلمي** | **○** Computer Science **○** InformationSystems **○** Information technology **○** Multimedia |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **م** | **اسم الطالب** | **هل المشروع يحقق الهدف المطلوب؟** | **Documentation**  **(التنظيم العام، المحتوى، مستوى الكتابة، الأسلوب، مراعاة الاخلاقيات)** | **Presentation**  **(درجة الإلمام،**  **الوضوح، الإتقان).** | **استيعاب وتمكن الطالب من المشروع والرد على استفسارات وأسئلة اللجنة** | **اجمالي الدرجات** |
| **5 درجات** | **5 درجات** | **5 درجات** | **25 درجة** | **40 درجة** |
| 1 | محمد احمد محمد عبد الغنى |  |  |  |  |  |
| 2 | محمد احمد عسكر |  |  |
| 3 | رحمة محمد ابراهيم سالم |  |  |
| 4 | حبيبة عصام محمد كمال عبدالدايم |  |  |
| 5 | الاء محمد عبدالستار العطار |  |  |
| 6 | فاطمه حصافى سعد ابراهيم عجيبه |  |  |
| 7 | عمرو صبري عبدالمنعم عمرو |  |  |  |  |  |

**الممتحنين وكيل الكلية لشئون التعليم والطلاب عميد الكلية**

**Acknowledgment**

I'd be glad to acknowledge the contributions of those who have made my development possible:

**My Supervisors:** Their guidance, support, and expertise have been invaluable throughout my creation and ongoing learning process.

**Faculty Members:** I am grateful to the faculty members who share their knowledge and mentorship, contributing to the foundation upon which I am built.

**My Team:** The engineers and researchers who work tirelessly to improve me deserve a tremendous amount of credit.

**Declaration**

We hereby certify that this material, which we now submit for assessment on the program of study leading to the award of Bachelor of Computers and Information in ( information technology and information system ) is entirely our own work, that we have exercised reasonable care to ensure that the work is original, and does not to the best of our knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of our work.

**Project Abstract**

**Table of contents**

|  |  |
| --- | --- |
|  | **Chapter 1: introduction.** |
|  | **1.1 project definition** |
|  | **1.2 project objectives** |
|  | **1.3 project scope** |
|  | **1.4 project timeline** |
|  | **1.5 project organization** |
|  | **Chapter 2: literature review.** |
|  | **2.1 Background** |
|  | **2.2 purpose** |
|  | 2.2.1 Goal |
|  | 2.2.2 Target Audience |
|  | **2.3 Overall Description** |
|  | 2.3.1 Project perspective |
|  | 2.3.2 Product function |
|  | 2.3.3 User Classes and Characteristics |
|  | 2.3.4 User Documentation |
|  | **2.4 Programming languages and IDE** |
|  | 2.4.1 Mobile app |
|  | 2.4.2 Flutter |
|  | 2.4.3 Firebase |
|  | 2.4.4 Website |
|  | 2.4.5 Front-end |
|  | 2.4.6 Back-end |
|  | **Chapter 3: System analysis** |
|  | **3.1 Requirements** |
|  | 3.1.1 User Requirements |
|  | 3.1.2 System Requirements |
|  | **3.2 Review of relevant work** |
|  | 3.2.1 KCN901 |
|  | 3.2.2 CrossHelmet X1 |
|  | **3.3 The difference between relevant work and our project** |
|  | **Chapter 4: System Design** |
|  | **4.1 Diagrams** |
|  | 4.1.1 Usecase diagram |
|  | 4.1.2 DFD |
|  | **4.2 External interfaces** |
|  | 4.2.1 User interfaces |
|  | **4.3 Hardware Design** |
|  | **Chapter 5: System implementation** |
|  | **5.1 Mobile app** |
|  | 5.1.1 Introduction |
|  | 5.1.2 Connection with database |
|  | 5.1.3 Dependencies |
|  | 5.1.4 Login |
|  | 5.1.5 Stream builder function |
|  | 5.1.6 Chat functions |
|  | 5.1.7 Notifications functions |
|  | **5.2 Website** |
|  | **5.3 Web Deployment** |
|  | **Chapter 6: Helmet implementation** |
|  | **6.1 temperature & humidity**  6.1.1 Overview of the Sensor  6.1.2 Wiring diagram and working method  6.1.2.1 Wiring diagram  6.1.2.2 Working method  6.1.3 Code and Test result  6.1.3.1 Code  6.1.3.2 Test result |
|  | **6.2 Gas monitoring**  6.2.1 Overview of the Sensor  6.2.2 Wiring diagram and working method  6.2.2.1 Wiring diagram  6.2.2.2 Working method  6.2.3 Code and Test result  6.2.3.1 Code  6.2.3.2 Test result |
|  | **6.3 Motion objects detection** |
|  | **6.4 GPS** |
|  | **6.5 Fall detection** |
|  | **6.6 Sensor-Server connection** |
|  | **6.7 Compare between low-cost sensor and high-cost sensors** |
|  | **Chapter 7: Conclusion and future work** |
|  | 7.1 Future work |
|  | 7.2 Project budget and Conclusion |
|  | 7.3 Resources |

**Table of Figures**

|  |  |
| --- | --- |
| Figure 1: Gantt chart |  |
| Figure 2: manager flow chart |  |
| Figure 3: supervisor flow chart |  |
| Figure 4: Structure of firebase database |  |
| Figure 5: KCN901 |  |
| Figure 6: crossHelmet x1 |  |
| Figure 7: Usecase diagram |  |
| Figure 8: DFD |  |
| Figure 9: manager login (web) |  |
| Figure 10: manager login (mobile) |  |
| Figure 11: manager home (web) |  |
| Figure 12: manager home (mobile) |  |
| Figure 13: add supervisor (web) |  |
| Figure 14: add supervisor (mobile) |  |
| Figure 15: add location (web) |  |
| Figure 16: add location (mobile) |  |
| Figure 17: helmets page (web) |  |
| Figure 18: helmets page (mobile) |  |
| Figure 19: add worker (web) |  |
| Figure 20: add worker (mobile) |  |
| Figure 21: account page (mobile) |  |
| Figure 22: info page (web) |  |
| Figure 23: info page (mobile) |  |
| Figure 24: supervisor login (web) |  |
| Figure 25: supervisor login (mobile) |  |
| Figure 26: supervisor home (web) |  |
| Figure 27: supervisor home (mobile) |  |
| Figure 28: assign page (web) |  |
| Figure 29: assign page (mobile) |  |
| Figure 30: workers page (web) |  |
| Figure 31: workers page (mobile) |  |
| Figure 32: worker page (web) |  |
| Figure 33: worker page (mobile) |  |
| Figure 34: support page (web) |  |
| Figure 35: support page (mobile) |  |
| Figure 36: chat page (mobile) |  |
| Figure 37: chats page (mobile) |  |
| Figure 38: DTH22 sensor pinout |  |
| Figure 39: DHT22 & ESP 32 |  |
| Figure 40: Temperature and Humidity flowchart |  |
| Figure 41: Test temperature and Humidity |  |
| Figure 42: MQ5 sensor pinout |  |
| Figure 43: MQ5 & ESP 32 |  |
| Figure 44: CO and LPG flowchart |  |
| Figure 45: Test CO and LPG |  |
| Figure 46: SQLite database schema |  |
| Figure 47: |  |
| Figure 48: |  |
| Figure 49: |  |
| Figure 50: |  |
| Figure 51: |  |
| Figure 52: |  |
| Figure 53: |  |
| Figure 54: |  |
| Figure 55: |  |
| Figure 56: |  |
| Figure 57: |  |
| Figure 58: |  |
| Figure 59: |  |
| Figure 60: |  |
| Figure 61: |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |



Smart safety helmet

Introduction



smart safety Helmet team

Faculty of computers and information

**Introduction**

In today's fast-paced world, safety remains a paramount concern, especially in high-risk environments like construction sites, industrial settings, and for cyclists and motorcyclists. Traditional helmets, while crucial for protection, lack the ability to actively prevent accidents or provide real-time assistance. This is where the **Smart Safety Helmet** emerges as a revolutionary innovation.

Imagine a future where construction sites, factories, and other potentially hazardous workplaces are significantly safer for everyone. This vision is becoming a reality with the introduction of smart safety helmets. This innovative project combines cutting-edge technology with robust software to create a comprehensive safety solution for both workers and management.

**For Workers:**

Enhanced Protection: Smart helmets go beyond the traditional hard hat, offering features like impact detection, fall alerts, and environmental monitoring.

Real-Time Awareness: Built-in sensors and communication systems can alert workers to potential hazards, such as gas leaks or approaching vehicles, keeping them informed and proactive.

Improved Comfort: Advanced designs consider ergonomics and ventilation to ensure worker comfort during long shifts.

**For Supervisors and Managers:**

Centralized Monitoring: A dedicated software system accessible through a website and mobile app provides supervisors and managers with a real-time overview of worker safety.

Data-Driven Insights: The system collects and analyzes data from the helmets, allowing for proactive identification and mitigation of potential risks.

Improved Communication: Integrated communication features enable clear and efficient communication between supervisors and workers, facilitating incident response and safety briefings.

This smart safety helmet project represents a significant stride towards a future where workplaces prioritize worker well-being. By empowering both workers and management with advanced technology, we can create safer environments for everyone.

**1.1 Problem Definition: Workplace Safety Challenges**

**Current State:**

* Traditional safety helmets offer limited protection beyond basic impact absorption.
* Lack of real-time awareness of potential hazards for workers.
* Limited visibility into worker safety for supervisors and managers.
* Reactive safety measures often implemented after accidents occur.
* Inefficient communication channels between workers and management can hinder incident response.

**Consequences:**

* Workers face a higher risk of injury or fatality from falls, impacts, and exposure to hazardous environments.
* Delayed response times to accidents can exacerbate injuries.
* Difficulty in proactively identifying and mitigating safety risks due to limited data and insights.
* Increased healthcare costs and worker downtime due to preventable accidents.
* Negative impact on company morale and productivity.

**Problem Statement:**

Develop a comprehensive safety solution for workers in hazardous environments that addresses the limitations of traditional safety equipment and empowers supervisors and managers with real-time data and communication tools.

**This project aims to:**

* Enhance worker safety through advanced helmet technology.
* Increase worker awareness of potential hazards.
* Provide real-time worker safety data and insights to supervisors and managers.
* Facilitate proactive risk management and accident prevention.
* Improve communication and response times in case of incidents.
* Ultimately, create a safer and healthier work environment for everyone.

**1.2 Project Objectives: Advancing Workplace Safety**

**The smart safety helmet project seeks to achieve the following key objectives:**

**1. Enhance Worker Protection:**

* Develop smart helmets equipped with features that significantly reduce the risk of head injuries from falls and impacts.
* Integrate sensors that detect environmental hazards like gas leaks or excessive heat, alerting workers to potential dangers.
* Improve helmet comfort through ergonomic design and ventilation, encouraging workers to wear them consistently.

**2. Increase Worker Situational Awareness:**

* Implement real-time communication capabilities within the helmets, enabling workers to receive crucial safety alerts from supervisors or the system itself.
* Design a visual or audio system within the helmets that provides warnings when approaching hazards or violating safety protocols.
* Enhance worker visibility through helmet-integrated lighting systems that can be activated in low-light conditions.

**3. Empower Supervisors and Managers:**

* Develop a comprehensive software system accessible through a website and mobile app to provide real-time worker location and safety data.
* Allow supervisors to monitor worker helmet status, including battery levels and potential alerts triggered by sensors.
* Enable supervisors to communicate directly with workers through the helmets, facilitating targeted safety briefings and incident response.

**4. Foster Proactive Safety Management:**

* Utilize the data collected from the smart helmets to identify recurring risks and trends in hazardous activities.
* Develop data-driven safety protocols and best practices based on real-world worker behavior and environmental conditions.
* Integrate the software system with existing safety management tools for centralized data analysis and reporting.

**5. Improve Communication and Collaboration:**

* Enable seamless communication between workers and supervisors through an integrated messaging system within the helmets and the software platform.
* Facilitate efficient reporting of safety incidents and near misses by workers directly through the helmets or app.
* Encourage a culture of open communication and shared responsibility for safety within the workplace.

By achieving these objectives, the smart safety helmet project aims to create a safer work environment, reduce worker injuries, and improve overall safety culture within organizations.

**1.3 Scope**

Project scope is a way to set boundaries on your project and define exactly what goals, deadlines, and project deliverables you'll be working towards.

**1.3.1 Statement of Work**

The document part is a very important part, as it provides a detailed explanation of the hardware and software, the interaction between them, the most important functions in the system, and how they work.

The unified modeling language (UML) is a general-purpose visual modeling language that is intended to provide a standard way to visualize the design of a system , it provides a standard notation for many types of diagrams which can be roughly divided into three main groups: behavior diagrams, interaction diagrams, and structure diagrams.

The word UI is an abbreviation for User Interface, which means user interface, which is the interface that the user sees when entering your website or application. UX is an abbreviation for User Experience, which means user experience, and it is everything related to the experience that the user performs within your website or application, and it depends on facilitating Its own experience by creating good colors that are comfortable for the eye, making the elements easily accessible, and making the experience simple and smooth on your site, and access to everything is simple and there is nothing complicated on your site.

The application or website part is responsible for clarifying what the manager and supervisor can do to interact with the smart helmet.

The hardware part , in which the sensors will be installed on the helmet and will be programmed and connected to the application or website.

**1.3.2 Deliverables**

Application And Website UX/UI (6/11/2023).

Documentation, Application, Website and Diagrams (11/2/2024).

**1.3.3 Acceptance criteria**

This system will be tested with a simple experiment of the risks to which the worker may be exposed. For example, we can light a fire, which causes a rise in temperature, and from that we can test the temperature and humidity sensor. We can also drop the helmet from a high place to test the fall sensor. These are simple examples with which we can test the accuracy of the smart helmet.

**1.4 Project timeline**

**GANTT chart:**

Figure 1 / gantt chart

**1.5 Document Organization**

Our project documentation consists of six parts in addition to three appendices. These parts give a scientific definition of our project and an understanding of stated problems, how they can be solved, and the objective of the project.

A brief description of the contents of each part:

**Chapter 1:** Introduction, introduces the project objectives, the motivation of the project, the approach used in this project, and the scope of the project.

**Chapter 2:** This section covers project techniques, technologies, related works, and new features. It highlights pros and cons of the related works, and theinspiration for the application to combine features and address defects.

**Chapter 3:** This part, will focus on explaining the requirements, and some relevant work with a compare between them and out work.

**Chapter 4:** This part, will focus on explaining the ERD, class diagram, and some detailed screens of the application, to show the importance of each of them

**Chapter 5:** This part shows the implementation of the system, shows the process of mapping design into implementation, and we also discuss test/achieved results.

**Chapter 6:** This part shows the implementation of the helmet, shows the process of mapping design into implementation, and we also discuss test/achieved results.

**Chapter 7:** Conclusion and future work, summarizes the entire research, and addresses the suggested improvement.



Chapter 2

Literature review



smart safety Helmet team

Faculty of computers and information

**2.1 Background**

Security and safety rules at construction and excavation sites are a major priority for any construction project, as maintaining the safety of workers, employees, contractors and visitors is an essential part of achieving project success.

Work-related diseases and injuries are responsible for the deaths of 1.9 million people in 2016, according to the first joint estimates from the World Health Organization and the International Labor Organization. Construction and excavation work is a dangerous job site. This work involves a number of hazardous tasks and conditions such as working at height, excavation, noise, dust, power tools, and equipment. The most common deaths in these jobs occur as a result of falling or being hit by a machine, tool, or thing, toxic gases, getting stuck between two things, and others. It is one of the most important preventive measures that must be taken to achieve security and safety on construction sites is to ensure that the necessary personal protective equipment, such as safety helmets, glasses, masks, protective shoes, and seat belts are provided, and that they are used correctly.

A helmet in work sites is the most important preventive safety equipment. In all laws of international safety organizations and occupational health and safety laws in governments and companies, everyone is required to adhere to wearing it in work sites. It is one of the proven tools to protect lives at work sites, according to previous studies and data. The helmet is important in preventing sudden flying objects from hitting the head, preventing electrical shock to the upper head, preventing head injuries when chemical liquids and high temperature are poured from the upper head, etc.

A smart helmet for workers on construction sites represents a technological advancement aimed at enhancing safety, efficiency, and overall well-being in the construction industry. This innovative piece of personal protective equipment integrates cutting-edge technologies to provide a comprehensive solution for addressing various challenges faced by construction workers.

The smart helmet is designed to offer robust protection against head injuries while incorporating a range of smart features to improve communication, monitoring, and data collection. It typically includes a durable and impact-resistant outer shell, ensuring the highest standards of safety compliance. The integration of intelligent technologies sets it apart from traditional helmets, transforming it into a multifunctional device that contributes to a safer and more productive work environment.

**Key features of a smart helmet for construction workers may include:**

1. **Communication Systems:** Built-in communication systems enable seamless interaction between workers, supervisors, and other stakeholders. This feature promotes instant communication, enhancing coordination and response times in emergency situations.
2. **Temperature and humidity measurement system:** By integrating a temperature and humidity sensor into the smart helmet, supervisors can be aware of workers' environmental conditions, especially in situations where extreme temperatures or high humidity levels may impact comfort and safety. If temperatures rise or fall abnormally , and of course humidity, the system can alert supervisors and initiate emergency protocols.
3. **Gas ratio sensor:** To measure the percentage of carbon dioxide (CO2) and petroleum gas derivatives (IPG) in the air, a gas sensor with specific capabilities is required. In the event of abnormal readings, the system can alert supervisors and initiate emergency protocols.
4. **Fall Detection System:** Accelerometers and gyroscopes can be employed to detect sudden indicative movements of a fall. In such cases, the helmet can automatically send alerts to the appropriate personnel and trigger emergency response protocols.
5. **GPS and Location Tracking:** Integrated GPS technology allows for real-time tracking of workers' locations, enhancing overall site management, and ensuring that workers are in designated safe zones.
6. **Object Detection System:** By integrating an object detection system into a smart helmet, the wearer gains an additional layer of awareness and safety, especially in environments where visibility is crucial, such as construction sites. The combination of object detection, augmented reality, and real-time alerts makes the smart helmet a valuable tool for enhancing situational awareness and preventing accidents.

The deployment of smart helmets in construction not only prioritizes the safety of workers but also contributes to increased efficiency, streamlined communication, and data-driven decision-making. As technology continues to evolve, smart helmets are likely to become integral tools in the modern construction industry, fostering a safer and more connected workplace.

**2.2 Purpose**

The purpose of this documentation is to provide a detailed description of the application or website, which is the main engine of the smart helmet, and explain the functions contained in it. It will explain the purpose and features of the system, the interfaces of the system, what the system will do. It will also explain the most important functions of the smart helmet, its shape, the types of sensors used in it, how it works and how it was programmed.

**2.2.1 Goal**

The software product is an application or website through which the manager can see the supervisors, work sites, workers and smart helmets worn by the workers and manage them in terms of additions and deletions. Through it, each supervisor can see the number of workers , see the status of the smart helmet if it is connected or not, and can send a message or call emergency in case of danger. The hardware product is a smart helmet equipped with many sensors. Through them, it can sense any change that occurs, such as a change in temperature and humidity, or sense toxic gases Such as carbon dioxide and petroleum derivatives. It can determine the worker’s location and also determine the presence of a foreign body near the worker. Finally, it can sense the worker falling. If any of these dangers are sensed, the smart helmet sends an alert to the supervisor through the application, and the supervisor takes the necessary steps.

**2.2.2 Target Audience**

The people who will benefit from this system and the smart helmet are those who may be exposed to a lot of financial losses or legal liability as a result of repeated accidents of workers at work sites that can cause serious injuries or, worse, the death of workers. This helmet can save a lot of money and effort, and of course the basic is to protect workers and rescue them quickly, and of course the worker is the person who benefits most from this smart helmet because he will work in comfort and without fear of serious injuries.

**2.3 Overall Description**

**2.3.1 Product Perspective**

The main goal is to design an effective system that provides rapid protection and rescue for the worker in the workplace. Therefore, the system was implemented in three stages. The first stage, which is the platform design stage, is suitable for displaying the smart helmet’s statistics and status. The second stage is designing and implementing the smart helmet and placing the sensors in the appropriate place to send the data. The third and final stage is to connect the smart helmet to the server so that the data and statistics appear on the platform. Thus, the supervisor is aware of the statistics and does what is required in the event of any danger.

A simple diagram explaining the key components in a manager's case.

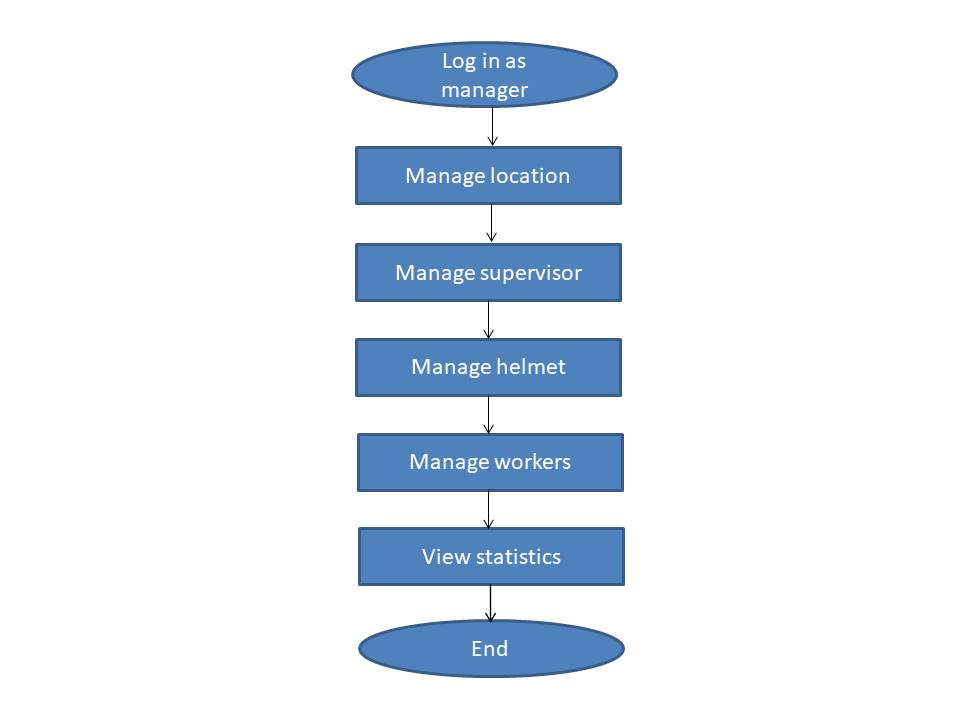


Figure 2 / manager flow chart

A simple diagram explaining the main components of a supervisor's case.

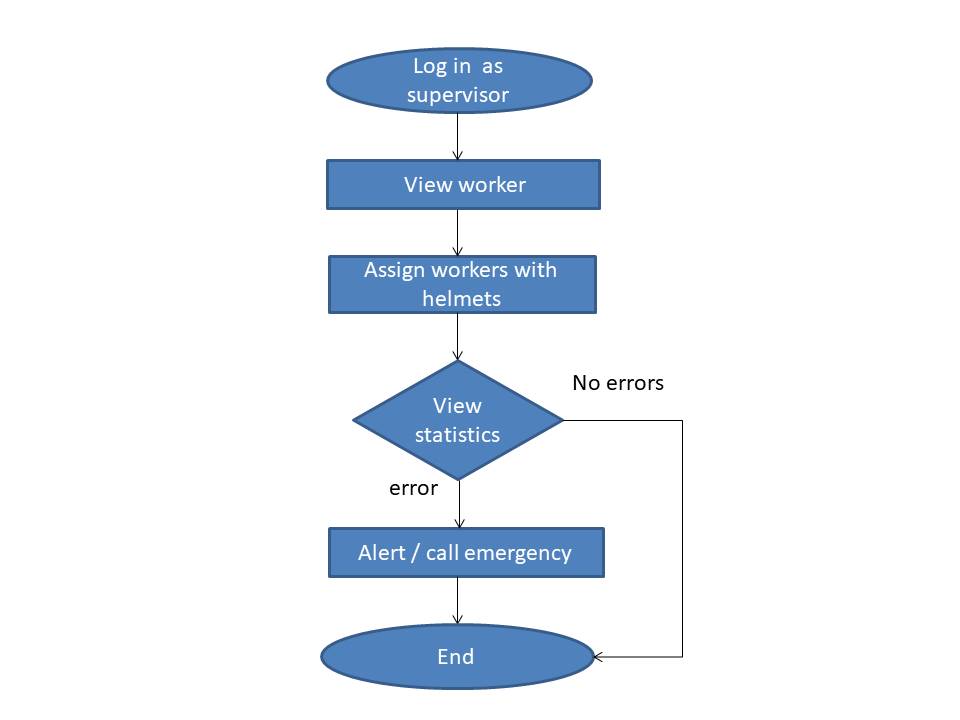


Figure 3 / supervisor flow chart

**2.3.2 Product Functions**

* A quick and accurate calculation of the helmet's condition, whether it is connected or not.
* Embedded System is used.
* Web site runs on both windows and Linux.
* Application runs on both android and ios.

**2.3.3 User Classes and Characteristics**

In our system, there are the admin and two main users, the first is the manager and the second is the supervisor.

* **Admin:** The person who has authority to manage the system, add the manager’s information, and confirm his account. He is also responsible for receiving reports.
* **Manager:** The manager is considered the first user in our system as he can control all of the supervisors, work sites, workers, and smart helmets, and manage them in terms of addition, deletion, and modification.
* **Supervisor:** The supervisor is the second user who plays a very important role in the system, as he is responsible for supervising the workers and the smart helmet, constantly reviewing the statistics, and contacting the emergency services in the event of a danger. He is also responsible for seeing the status of the helmet if it is connected or not.

**2.3.4 User Documentation**

The user (manager) must follow the following steps in order to get the desired result:

1. The manager will log in using his data previously entered by the admin.
2. He can control work sites and add them.
3. Then he manages the helmets and adds them to the work sites.
4. He employs workers and controls them in terms of additions and deletions
5. He also manages supervisors and adds their information.
6. Supervisors enter their data and can then do their job.

The user (supervisor) must follow the following steps in order to get the desired result:

1. The supervisor will log in using his data previously entered by the manager.
2. He will assign workers with helmets.
3. And he will view the workers page and reviews the status of the helmet whether it is connected or not.
4. Each sensor connected to the helmet will carry out its task.
5. The helmet will send data and statistics to the application.
6. The supervisor will constantly view and review the sent data and statistics.
7. If the statistics and data are normal and the helmet’s status is connected, the supervisor will complete his work and review the data.
8. If the statistics sent by the sensor related to the data are abnormal, the application sends an alert to the supervisor through the application.
9. The supervisor then calls emergency services to send help to the worker as soon as possible.
10. In the end, he can exit the application.

**2.4 Programming languages and IDE**

**2.4.1 Mobile app**

**What is a mobile app (mobile application)?**

A mobile app (or mobile application) is a software application developed specifically for use on small, wireless computing devices, such as smartphones and tablets , rather than desktop or laptop computers.  
Mobile apps are sometimes categorized according to whether they are web-based or native app , which are created specifically for a given platform. A third category, hybrid app , combines elements of both native and web apps. In today's digital age, mobile apps are an essential part of most people's daily lives. From social networking  and entertainment to productivity and business, mobile apps play a vital role in how we interact with technology.

## How are mobile apps built?

Mobile apps are built using a variety of programming languages and frameworks, and they can be downloaded and installed from app stores such as the Apple App store or Google play .Mobile apps are designed to provide a wide range of functions and services and with consideration for the demands, constraints and capabilities of the devices they're built for. For example, a gaming app might take advantage of the iPhone's accelerometer. Other examples include games, social media platforms , email clients and banking apps. They can also be used to access information, such as news and weather updates, and to perform tasks, such as online shopping and booking travel.

**How does a mobile app work?**

Mobile apps are designed to run on specific mobile operating systems such as iOS, Android and Windows Phone. When a mobile app is downloaded and installed on a device, it is stored in the device's memory and is launched using the device's operating system .When a user opens a mobile app, the app communicates with the device's operating system and other built-in software components to access the device's hardware and services such as the camera, GPS and internet connection. The app then uses this information to provide its specific functions and services to the user.

## Advantages of mobile apps

There are numerous advantages to using mobile apps, namely:

1. Convenience: Mobile apps can be downloaded and installed on a device, allowing users to access the app's functions and services at any time, from anywhere.
2. Personalization: Mobile apps can be customized to meet the specific needs of individual users, providing a personalized experience.
3. Offline access: Many mobile apps can be used offline, providing access to important information and features even when an internet connection is not available.
4. Push notifications:  Mobile apps can send push notifications to users, providing real-time updates on important information and events.

**Disadvantages of mobile apps**

However, there are some disadvantages and considerations to keep in mind when using mobile apps. These include:

1. **Limited functionality:** Mobile apps are designed to provide specific functions and services, and they might not be able to provide the same level of functionality as a desktop application.
2. **Limited compatibility:** Mobile apps are designed to run on specific mobile operating systems, and they might not be compatible with all devices.
3. **Security concerns:** Mobile apps can access sensitive information on a device, and they might not have the same level of security as a desktop application.
4. **Limited updating capability:** Some mobile apps might not be easily updated, and users might need to download a new version of the app to access the latest features and bug fixes

While mobile apps offer convenience, personalization and offline access, they also have limitations in terms of functionality, compatibility, security and updating capability. Business software buyers should consider the advantages and disadvantages of mobile apps when deciding on a distribution method for their software. As the mobile market continues to grow, mobile apps are becoming an increasingly popular way for businesses to engage with customers and employees, providing a convenient, personalized and secure way to access important information and perform tasks on the go.

**2.4.2 Flutter**

**What is Flutter?**

Flutter is an open-source framework developed and supported by Google. Frontend and full-stack developers use Flutter to build an application’s user interface (UI) for multiple platforms with a single codebase. When Flutter launched in 2018, it mainly supported mobile app development. Flutter now supports application development on six platforms: iOS, Android, the web, Windows, MacOS, and Linux**.**

## How does Flutter help app development?

## Flutter simplifies the process of creating consistent, appealing UIs for an application across the six platforms it supports. Because Flutter is a cross-platform development framework, we’ll first compare it to native development. Then, we can highlight features that are unique to Flutter

### Native app development compared to cross-platform app development

Coding an application for one specific platform, such as iOS, is called native app development. By contrast, cross-platform app development is building an application for multiple platforms with a single codebase.

### Native app development

Because developers code for a specific platform in native app development, they have full access to native device functionality. This generally leads to higher performance and speed compared to cross-platform app development. On the other hand, if you want to launch an application on multiple platforms, native app development requires more code and more developers. In addition to these expenses, native app development can make it harder to launch on different platforms at the same time with a consistent user experience. This is where cross-platform app development frameworks like Flutter can be useful.

### Cross-platform app development

Cross-platform app development allows developers to use one programming language and one codebase to build an application for multiple platforms. If you’re releasing an application for multiple platforms, cross-platform app development is less costly and time-consuming than native app development. This process also lets developers create a more consistent experience for users across platforms. This approach can have drawbacks compared to native app development, including limited access to native device functionality. However, Flutter has features that make cross-platform app development smoother and highly performant.

### The advantages of Flutter

Here are some ways that Flutter stands out as a cross-platform development framework:

* Close-to-native performance. Flutter uses the programming language Dart and compiles into machine code. Host devices understand this code, which ensures a fast and effective performance.
* Fast, consistent, and customizable rendering. Instead of relying on platform-specific rendering tools, Flutter uses Google’s open-source Skia graphic library to render UI. This provides users with consistent visuals no matter what platform they use to access an application.
* Developer-friendly tools. Google built Flutter with an emphasis on ease-of-use. With tools like hot reload, developers can preview what code changes will look like without losing state. Other tools like the widget inspector make it easy to visualize and solve issues with UI layouts.

## What programming language does Flutter use?

Flutter uses the open-source programming language Dart, which was also developed by Google. Dart is optimized for building UIs, and many of Dart’s strengths are used in Flutter. For example, one feature of Dart used in Flutter is sound null safety. Dart’s sound null safety makes it easier to detect common bugs called null errors. This feature reduces the time developers spend on code maintenance and gives them more time to focus on building their applications.

## What are widgets in Flutter?

In Flutter, developers build UI layouts by using widgets. This means everything a user sees on a screen, from windows and panels to buttons and text, are made of widgets .Flutter widgets are designed so developers can easily customize them. Flutter achieves this through a composition approach. This means most widgets are made up of smaller widgets, and the most basic widgets have specific purposes. This allows developers to combine or edit widgets to create new ones. Flutter renders widgets using its own graphic engine instead of relying on a platform’s built-in widgets. This way, users will experience a similar look and feel in a Flutter application across platforms. This approach also provides flexibility to developers, because some Flutter widgets can carry out functions that platform-specific widgets can’t. Flutter also makes it easy to use community-developed widgets. Flutter’s architecture supports having multiple widget libraries, and Flutter encourages the community to build and maintain new ones.

### Types of Flutter widgets

Flutter comes with an extensive widget catalog from the time you download it. The catalog has 14 categories, which include styling, Cupertino (IOS- style widgets), and Material Components (widgets that follow Google's Material Design guidelines). Flutter also comes with layouts and themes included, helping developers to build right away.

## How is Flutter supported?

Flutter is supported by Google and an active open source community on Reddit, Discord, Slack, Stack Overflow, and Gitter ,Google has consistently updated Flutter since its release in 2018, including with a Flutter 3 update in 2022 which extended stable support to macOS and Linux. To make Flutter easier to learn, Google has written extensive documentation and tutorials on Flutter’s site. To engage with Flutter users, Google also hosts global events, promotes community projects, and sponsors developer challenges. Upcoming events can be found on Flutter's site. Flutter’s community has created thousands of third-party packages and excellent tools that streamline the developer experience. These libraries are available at pub.dev.

# Run apps on the Android Emulator

# The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device. The emulator offers these advantages:

* Flexibility: In addition to being able to simulate a variety of devices and Android API levels, the emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.
* High fidelity: The emulator provides almost all the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.
* Speed: Testing your app on the emulator is in some ways faster and easier than doing so on a physical device. For example, you can transfer data faster to the emulator than to a device connected over USB.

In most cases, the emulator is the best option for your testing needs. This page covers the core emulator functionalities and how to get started with it. Alternatively, you can deploy your app to a physical device.

### **Emulator system requirements**

For the best experience, you should use the emulator in Android Studio on a computer with at least the following specs:

* 16 GB RAM
* 64-bit Windows, macOS, Linux, or ChromeOS operating system
* 16 GB disk space.

### **Create an Android Virtual Device**

Each instance of the Android Emulator uses an Android virtual device (AVD) to specify the Android version and hardware characteristics of the simulated device. To effectively test your app, create an AVD that models each device your app is designed to run on. To create an AVD, see Create and manage virtual devices. Each AVD functions as an independent device with its own private storage for user data, SD card, and so on. By default, the emulator stores the user data, SD card data, and cache in a directory specific to that AVD. When you launch the emulator, it loads the user data and SD card data from the AVD directory.

**2.4.3 Firebase**

Firebase is a platform launched by Google that specializes in developing applications for smart devices. The platform includes a set of tools that cover a large part of the services that developers usually have to build themselves, but they prefer to focus on the application experience itself. This includes: analytics, authentication, databases, file storage, etc. The services are hosted in the cloud and can be expanded with little to no effort on the part of the developer. By cloud hosting, we mean that the products contain back-end services that are fully maintained and operated by Google. The Firebase SDK interacts with these back-end services directly, without requiring any middleware between your application and the service. So, If you use one of Firebase's database options, you will typically write code to query your application's database. This is different from traditional app development, which typically involves writing front-end and back-end software. The application's front-end code calls the final API routes exposed to the back-end, so the back-end code actually works. On the other hand, Firebase tools bypass the traditional backend and leave the task to the client, as each of these tools is accessible via the Firebase console.

**What type of applications are compatible with Firebase?**

In fact, there is no limit to the types of applications that are compatible with Firebase, only the platforms that can be used with it. IOS and Android are the two main Firebase SDK targets, and there is increased support for the web, the open source Flutter SDK, the Unity game engine, and the C++ programming language .On top of the development software packages, there is a library called FirebaseUI that provides a set of useful utilities to make developing with Firebase easier. There are also projects like AngularFire that subject web development software packages to use with the Angular framework for designing website interfaces. It's open source, by the way.

**What is Firebase?**

**Realtime Database**

Real-time data is the way of the future. Nothing compares to it. Most databases require you to call HTTP to get your data and synchronize it. So it gives you data "only" when requested/recalled. When you connect your app to Firebase, you're not communicating via regular HTTP. Rather, it is via the WebSockets protocol (which is a much faster protocol than HTTP). Not only that, Firebase automatically syncs all your data via a single WebSocket as fast as the user's network can handle it. Firebase sends new data as soon as it is updated. When your client (user) saves a data change, all connected clients receive the updated data in real time.

**File Storage**

Firebase Storage provides a simple way to save files to Google Cloud Storage directly from the client. And Firebase Storage has its own system of security rules to protect your GCloud bucket from the masses, while granting detailed write privileges to your authenticated clients. Firebase Storage has its own system of security rules to protect your GCloud cluster from public modifications, while granting separate write privileges to your authorized clients.

**Authentication system**

Firebase authentication has an email and password authentication system. It also supports two-factor authentication systems for Google, Facebook, and Twitter one of the advantages of Firebase Auth is its ability to integrate directly into the Firebase database, so you can use it to control access to your data.

**Hosting system**: Firebase includes easy-to-use hosting for all your static files. To make development easier, Firebase uses Firebase Superstatic hosting, which you can run on your local server for all your tests.

**Advantages of Firebase**

1. Email, password, Google, Facebook, and Github authentication.
2. Real time data.
3. Full API compatibility.
4. An internal security system at the level of nodes in a data tree.
5. File storage powered by Google Cloud Storage.
6. Static file hosting service.
7. Manipulating data (as streams) to create highly scalable applications.

**Disadvantages of Firebase**

1. Limited query capacity due to its data transfer model.
2. Move SQL functions not supported (due to data model mismatch with NoSQL)
3. Does not include mounting hardware.

**Structure of Firebase Database**

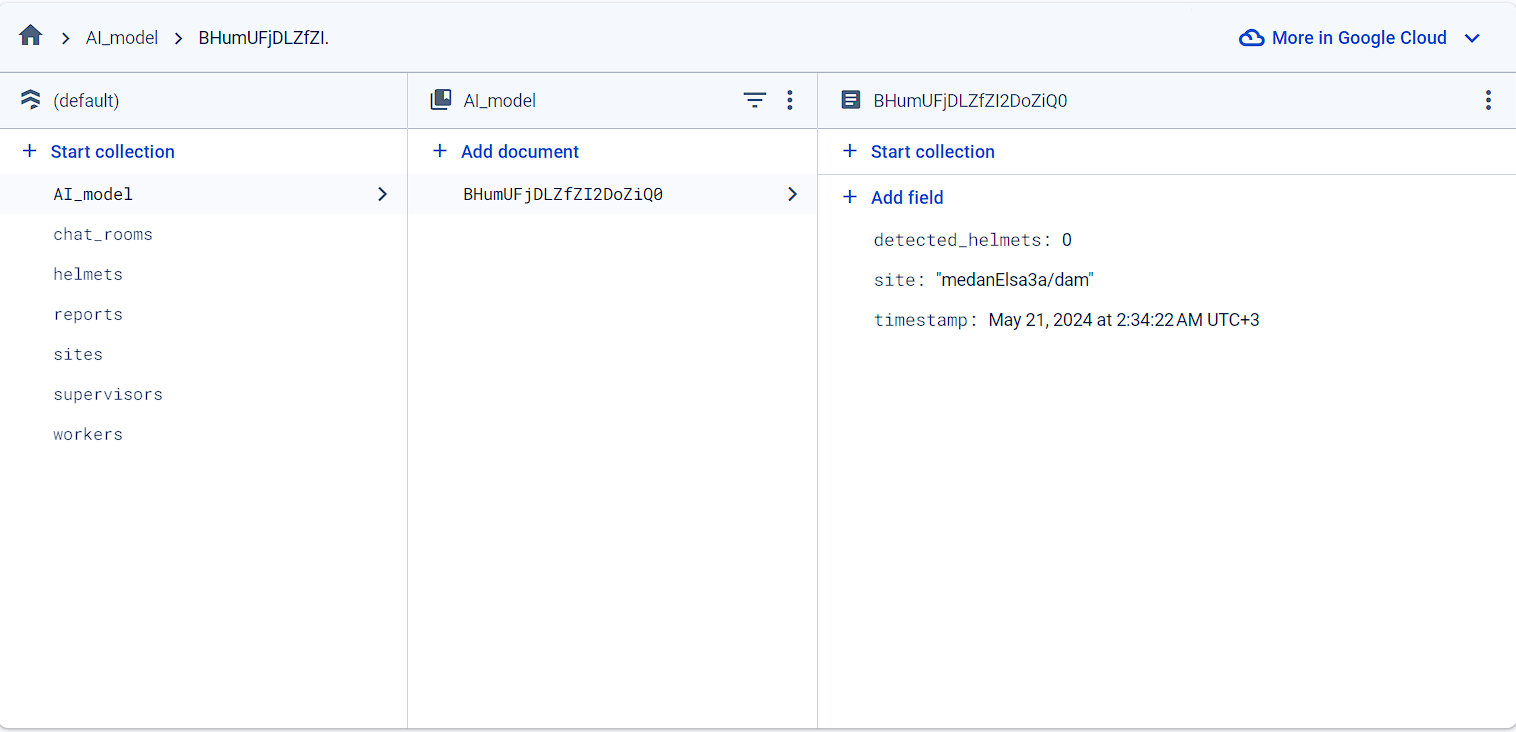


Figure 4 / Structure of firebase database

**2.4.4 Website**

**What is a website?**

A website is a collection of web pages and related content that is identified by a common domain name and is accessible over the internet or a private network. Websites are designed to be viewed through a web browser, and they can serve various purposes, such as providing information, offering services, or facilitating communication.

**The languages used to create a website can be broadly categorized into two types:**

**2.4.5 Front-End**

## What is frontend development?

## The frontend of a website is what you see and interact with on your browser. Also referred to as client-side, it includes everything the user experiences directly: from text and colors to buttons, images, and navigation menus .Let’s say you decide to start a business. You open a gourmet dog treat bakery, and need a professional website to present your company to customers and tell them where you’re located, Maybe you’ll include a few photos and some information about your products. All you need are frontend skills to build your website.

## We used this:

1. **HTML**

HTML is a markup language that defines the structure of your content. HTML consists of a series of elements, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicize words, can make the font bigger or smaller, and so on.

1. **CSS**

CSS or Cascading Style Sheets, is a style sheet language used to describe the presentation and formatting of a document written in HTML or XML (including XML dialects such as SVG or XHTML). In simpler terms, CSS is used to control the layout and appearance of web pages.

1. **Java script**

JavaScript in front-end development brings websites to life by adding interactivity, dynamic content, and responsiveness. It allows developers to respond to user actions, manipulate page elements, validate form inputs, handle events, implement AJAX functionality, and ensure browser compatibility. In essence, JavaScript enhances user experience and makes web pages more engaging and interactive.

## 2.4.6 Backend

## What is backend development?

The backend (or “server side”) is the portion of the website you don’t see. It’s responsible for storing and organizing data, and ensuring everything on the client-side actually works .The backend communicates with the frontend, sending and receiving information to be displayed as a web page .Whenever you fill out a contact form, type in a web address, or make a purchase (any user interaction on the client-side), your browser sends a request to the server side, which returns information in the form of frontend code that the browser can interpret and display. You can learn more about what these developers do in our full backend developer guide .Your new dog treat site will need to have additional backend components to make it a dynamic web application—a website whose content can change based on what is in its database, and that can be modified by user input .This is distinct from a static website, which doesn’t require a database because its content generally stays the same.

## We used this:

## 1.Python

How is Python used for web development? Python’s core features make it a popular option for web development. Firstly, Python is free**,**open-source, and widely available. More importantly, though, it is also highly adaptable. Python allows developers to create websites according to several different programming paradigms. For instance, it’s suitable for both object-oriented programming (OOP).  and functional programming (FP). You can learn about the differences between the two in our guide to FP vs OOP. It also boasts dynamic typing capabilities. In layman’s terms, this just means that Python scripts don’t require compiling (or translating) before execution. Instead, they’re executed at runtime. This is useful for web development, since it requires less coding and makes debugging easier. We could go and, and in fact, we have—we’ve created a beginner’s guide to the advantages of web developers learning Python. As it happens, there’s a reason these two languages are the most popular. Namely, JavaScript and Python are often used together. Web developers may use JavaScript to create the presentation layer (or frontend) and Python to create the server-side layer (or backend). Fortunately, backend coding using Python is much easier using web frameworks freely available on PyPI. Two of these, Django  and Flask , are especially popular and remove much of the heavy lifting from backend coding. These libraries are widely accepted as secure, scalable, and—once you get to grips with Python’s straightforward syntax—easy to use.

## 2.Django

## Django is a high-level Python web framework that enables rapid development of secure

## and maintainable websites. A framework is nothing more than a set of modules that make

## development easier. These modules are packaged together, and this allows you to create

## applications or websites from a known, consistent source, rather than writing code from

## scratch. Django is free and open-source, has a thriving and active community, great

## documentation, and many options for free and paid support. In simple terms, Django is a

## set of ready-made components that help you quickly build websites with a clean and

## functional structure. When designing websites, developers often need the same

## components for tasks such as authenticating users, uploading files, and implementing

## content management system

## We also used Firebase on the website, and we presented its features in the mobile application section, as mentioned above.



Chapter 3

System analysis



smart safety Helmet team

Faculty of computers and information

**3.1 Requirements:**

**3.1.1 User Requirements:**

A system shall help the user to constantly obtain helmet statistics and generate an alert when there is danger.

**3.1.2 System requirements:**

(SRS) that identifies the specific features , functions , nonfunctional requirements and requisite use cases for the software.

**Function Requirements:**

Functional requirements are statements or goals used to define system behavior. Functional requirements define what a software system must do or not do. They are typically expressed as responses to inputs or conditions.

• The manager shall log in for using the system.

• The manager shall enter the correct data for the supervisors so that they can log in and use the system.

• The manager shall add workers to the system

• The manager shall distribute helmets to the workplaces.

• The supervisor shall distribute workers to helmets and workplaces.

• Implement Bluetooth or Wi-Fi for wireless communication with external devices, enabling data exchange and connectivity features.

• Provide a user-friendly interface for controlling and customizing the smart helmet's features, including buttons, touch controls.

• Optimize power consumption to ensure a reasonable battery life for the smart helmet.

• Each sensor shall perform its function and send the results to the system.

• The system shall continuously provide statistics about helmets.

• If there is a possibility of a danger, the system shall send an alert to the supervisor.

• If it is confirmed that there is a danger, the supervisor shall call emergency services.

• Enable integration with mobile applications to enhance functionality, synchronization, and remote control.

• Design the smart helmet to be resistant to weather conditions, ensuring reliable performance in various environments.

**Non-functional requirements:**

Non-functional requirements are a set of specifications that describe the system’s operation capabilities and constraints and attempt to improve its functionality. These are basically the requirements that outline how well it will operate including things like speed, security, reliability, data integrity, etc.

1.Performance:

• Response Time: The smart helmet should provide real-time responses to inputs and deliver information with minimal latency.

• Throughput: The system should handle a specified number of transactions or data processing tasks per unit of time, ensuring efficient performance.

2.Reliability:

• Availability: The smart helmet should be available and operational for an acceptable percentage of the time, ensuring that critical functionalities are consistently accessible.

• Fault Tolerance: The system should be able to continue functioning or gracefully degrade in the face of hardware or software failures.

3.Security:

• Data Encryption: Communication between the smart helmet and external devices should be encrypted to ensure the confidentiality and integrity of transmitted data.

• User Authentication: Implement secure user authentication mechanisms to prevent unauthorized access to the smart helmet's features and data.

4.Usability:

• User Interface Design: Ensure that the user interface is intuitive, easy to use and accessible, enhancing ease of use for supervisors in the event of an emergency.

5.Scalability:

• Device Compatibility: Ensure that the smart helmet can integrate with a variety of external devices and platforms, supporting scalability in terms of connectivity.

• System Capacity: Design the system to handle an increasing number of users or devices as the deployment scales.

6.Maintainability:

• Modularity: Design the smart helmet system with modular components, facilitating easy updates, repairs, or replacements.

• Software Updates: Provide a mechanism for users to update the firmware or software of the smart helmet to incorporate new features and security patches.

7.Compatibility:

• Operating Systems: Ensure compatibility with popular operating systems to support a wide range of devices.

• Peripheral devices: The smart helmet must be compatible with various peripheral devices, such as smartphones, tablets, and other wearable devices.

8.Environmental Considerations:

• Weather Resistance: The smart helmet should be designed to withstand different weather conditions, ensuring reliable performance in various environments.

• Temperature Tolerance: Specify the acceptable temperature range within which the smart helmet should operate effectively.

9.Regulatory Compliance:

• Safety Standards: Ensure that the smart helmet complies with relevant safety standards and regulations, especially those pertaining to personal protective equipment (PPE).

10.Data Privacy:

• User data protection: Implement measures to protect user data, whether supervisor or manager, and comply with data protection regulations.

11.Interoperability:

• Communication Protocols: Specify interoperability standards for communication between the smart helmet and external devices, ensuring seamless integration.

Considering both functional and non-functional requirements is crucial for the successful development and deployment of a smart helmet system, meeting user expectations and industry standards.

**3.2 Review of relevant work:**

Given the importance of reducing the risk of injury, there are many ideas that have been implemented, including:

Figure 5 / KCN901

* + 1. **KCN901:**

KC Smart Helmet N901 is a wearable intelligent helmet for the actual combat needs of epidemic control and control in public places, This helmet is designed based on KC intelligent metamaterial technology, and integrates multiple advanced technologies such as infrared thermal imaging, artificial intelligence, AR (augmented reality) display, big data, communication sensing, and advanced aerospace technology. It is the first to implement mobile deployment-based control scenarios. The unaware and contactless temperature measurement is portable, safe and effective in epidemic prevention.

# 3.2.2 CrossHelmet X1:

CrossHelmet X1 is a wearable smart helmet for motorcycle drivers to keep them safe, this helmet can show the measures of body temperature, air humidity, Wind speed, Alcohol percentage and detect the Traffic jam.

Figure 6 / crosshelmet x1

**3.3 The difference between relevant work and our project:**

Most of the ideas on the market are directed at motorcycle drivers, while the ideas that focus on workers on sites have limited functionality and are limited to showing numbers to the worker, whether on a LCD screen or mobile application. While our project aims to create a complete system linking workers on sites with supervisors and responsible managers and providing them with reports and warnings in the event of a danger, whether a worker falls, an object collides with the worker, high temperature or humidity, or the detection of a toxic gas( CO / LPG ).



Chapter 4

System Design



smart safety Helmet team

Faculty of computers and information

**4.1 Diagrams:**

**4.1.1 Usecase diagram**



Figure 7 / Usecase diagram

**4.1.2 DFD**

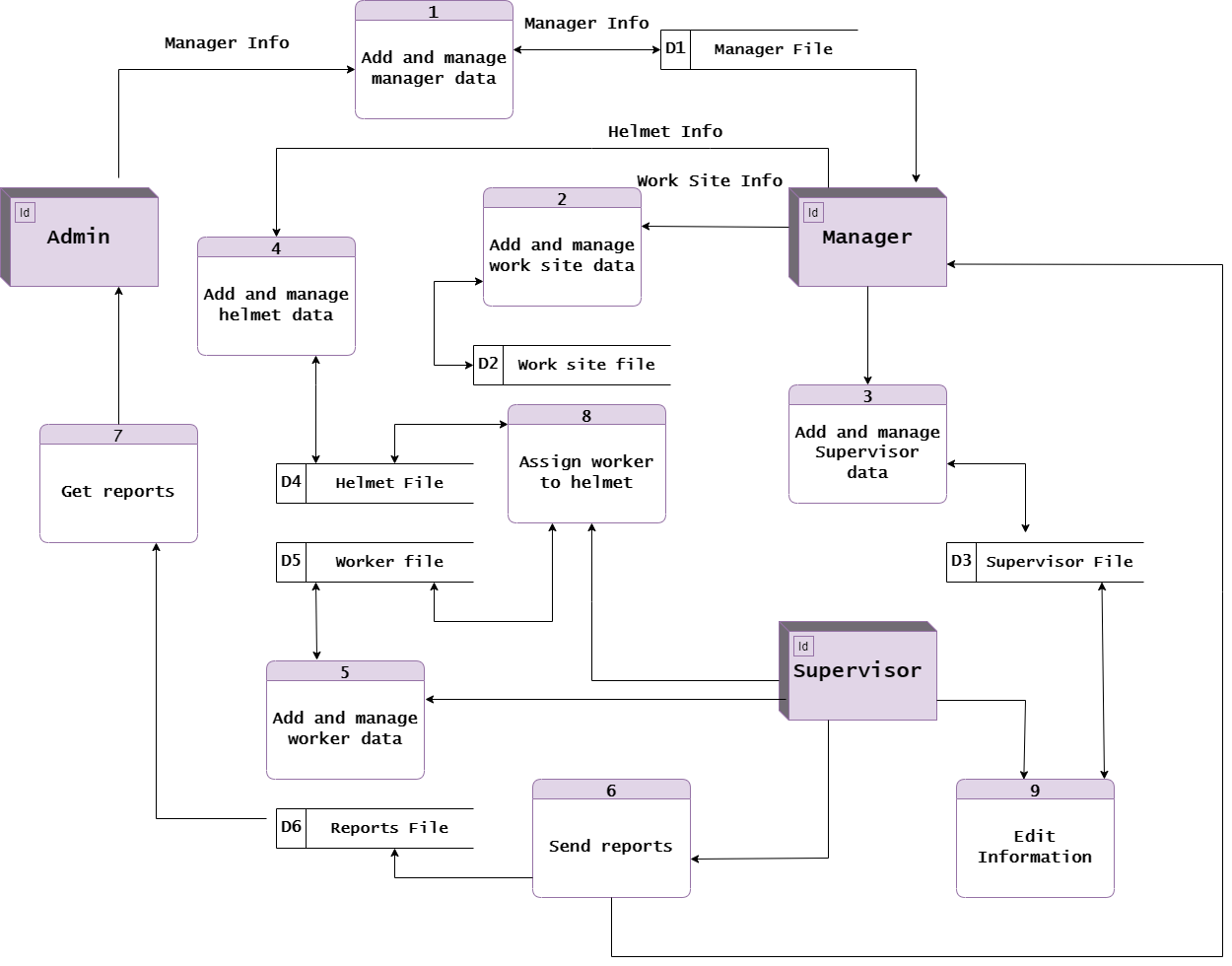
****

Figure 8 / DFD

**4.2 External Interfaces Design:**

**4.2.1 User Interfaces**

• We have two systems, application and website, which perform the same function

• Each part of the user interface intends to be as user friendly as possible.

• There are various pages for the system could be:

1) Sing in page as manager.

2) Home page.

3) Add supervisor page.

4) Add location page.

5) Add helmet page.

6) Add worker page.

7) User account page.

8) Information page.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1) Sign in page as supervisor

2) Home page.

3) Assign worker with helmet page.

4) Workers page.

5) Each worker page.

6) Statistics page.

7) Support page.

8) User account page.

9) Information page.

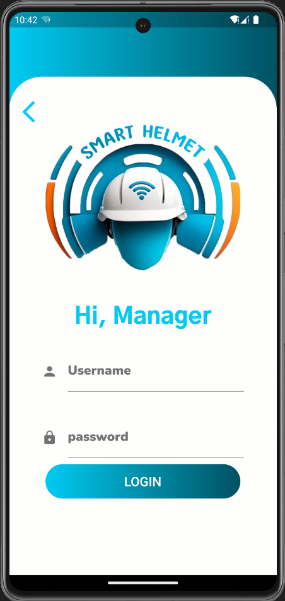
 Sign in page

Figure 9 / manager login (web)

Figure 10 / manager login (mobile)

First, there will be a page through which the user chooses whether he is a supervisor or manager so that the appropriate tasks appear to him, and after choosing, he will be moved to the sign in page.

The Sign in page requires the manager to enter the name and password so that he can move to the home page.

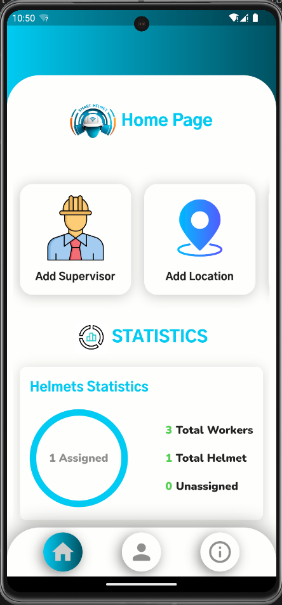
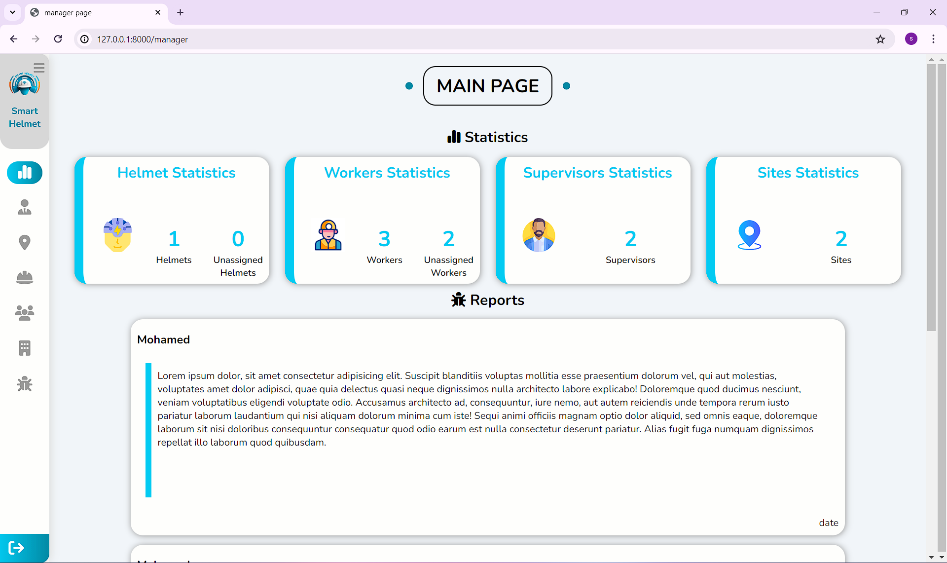
 Home page

Figure 11 / manager home (web)

Figure 12 / manager home (mobile)

The home page is considered the start after logging in, which can move the manager to many pages (add workers, statistics, add helmets, .....).

Through the home page, the manager can also see helmet statistics and the number of helmets connected to workers.

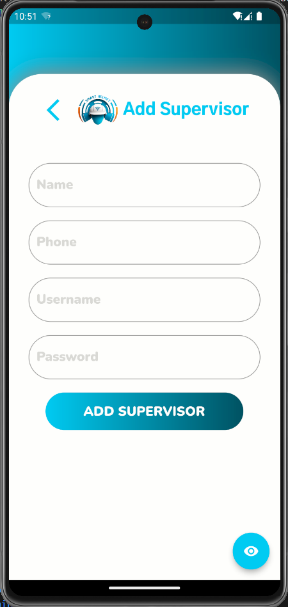
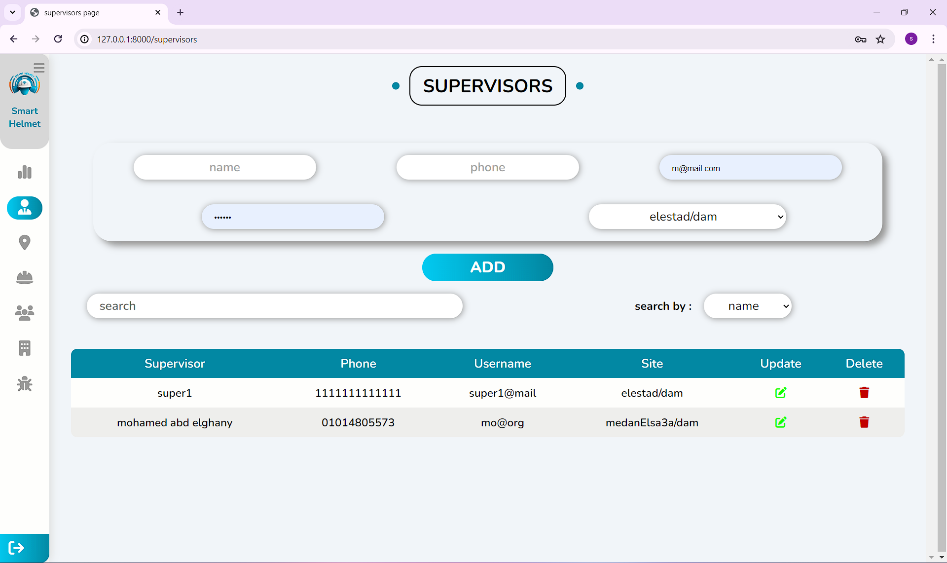
 Add supervisor page

Figure 13 / add supervisor (web)

Figure 14 / add supervisor (mobile)

Through the Add Supervisors page, the manager can enter the supervisor’s data and add him so that the supervisor can sign in.

He can also control the supervisors in terms of searching, adding, and modifying.

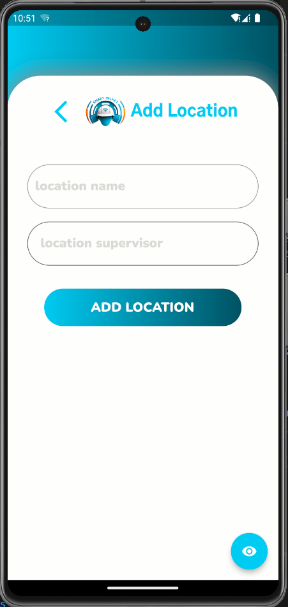
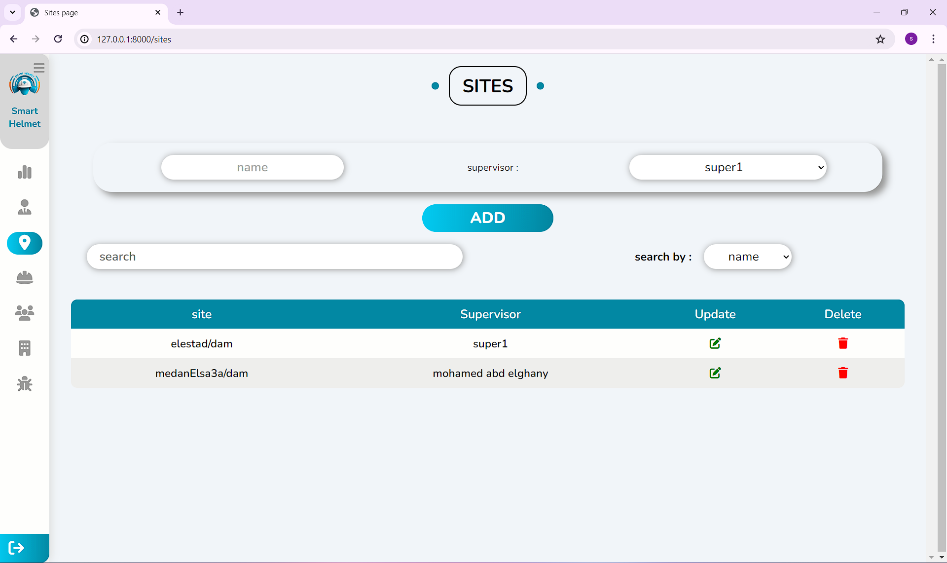
 Add location page

Figure 15 / add location (web)

Figure 16 / add location (mobile)

Through the Add location page, the manager can add a site through the x coordinate and the y coordinate and appoint a supervisor for this location.

The manager can also manage locations in terms of searching, editing, and deleting.

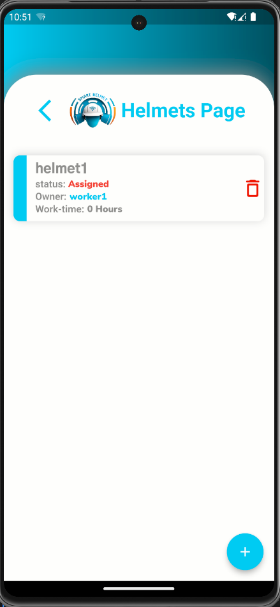
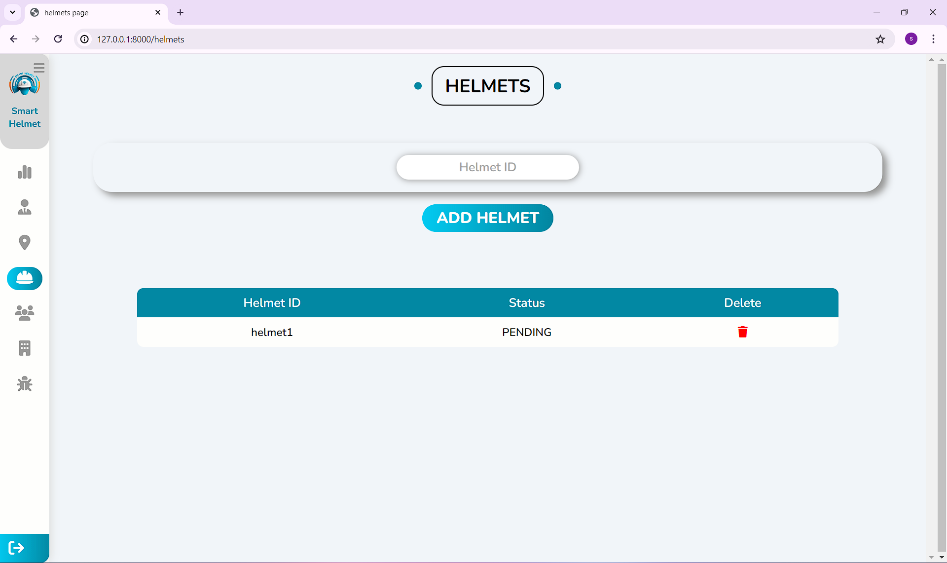
 Add helmet page

Figure 17 / helmets page (web)

Figure 18 / helmets page (mobile)

Through the Add Helmet page, the manager can add a helmet using a specific id.

The manager can also see the number of helmets, their status, whether they are connected or not, the owner of the helmet, and the number of working hours.

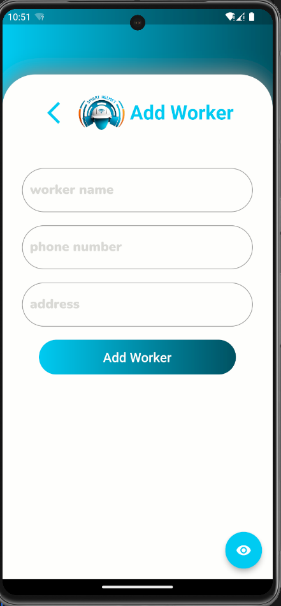
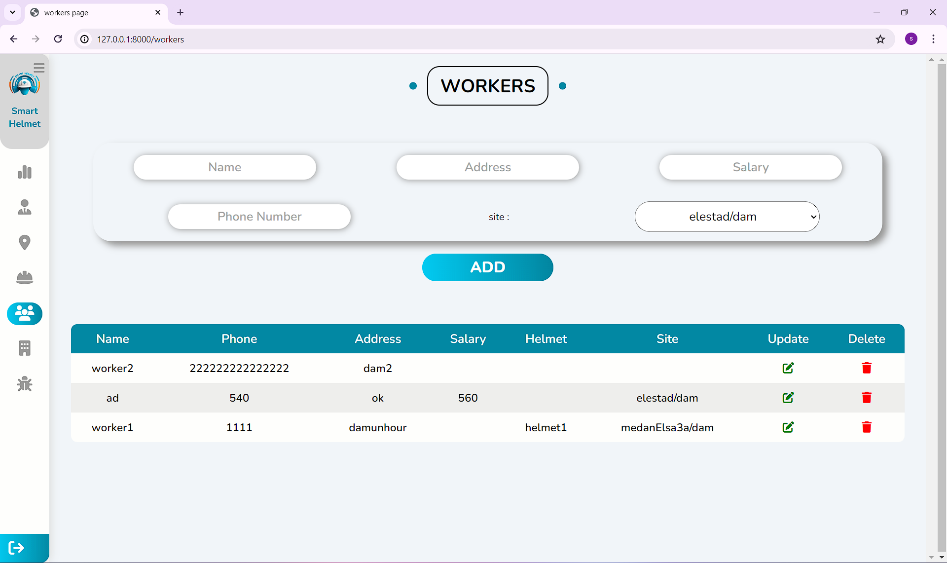
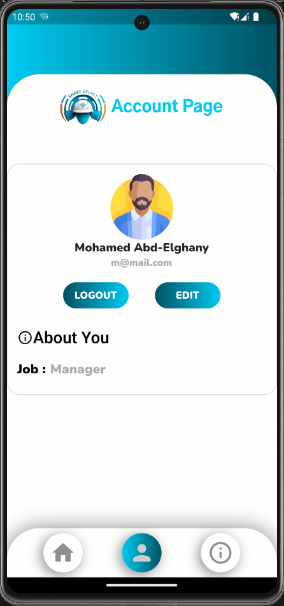
 Add worker

Figure 19 / add worker (web)

Figure 20 / add worker (mobile)

Through the Add Worker page, the manager can add the worker and his basic data.

The manager can also manage workers in terms of searching, modifying and deleting.

 User account page

The user account page is a very ordinary page that does not perform any function other than displaying the manager’s basic data, such as the name, image, ..... where the data can be modified or logged out.

Figure 21 / account page (mobile)

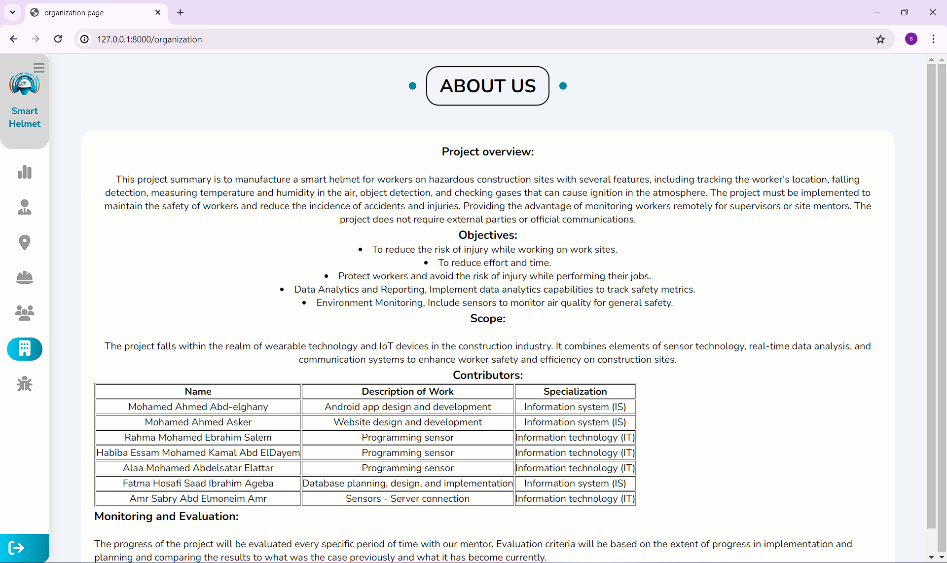
 Information page

Figure 22 / info page (web)

Figure 23 / info page (mobile)

The information page displays basic information about us.

**We will now move to the supervisor part.**

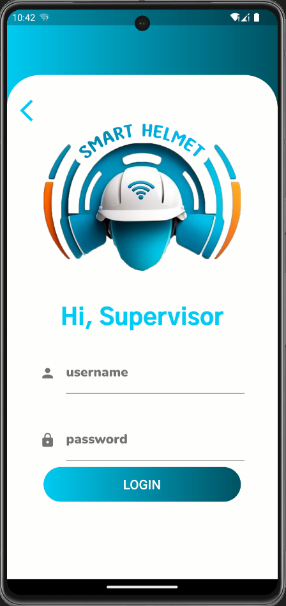
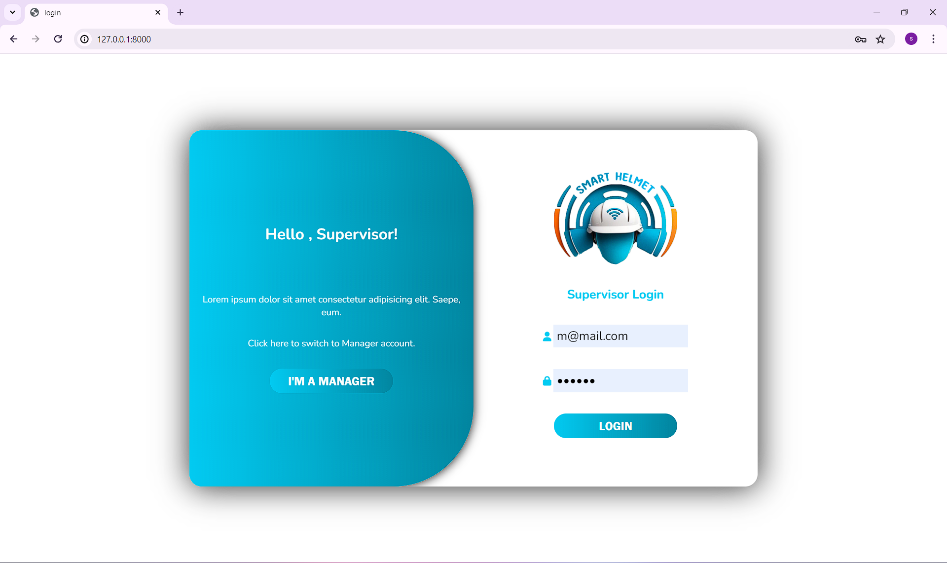
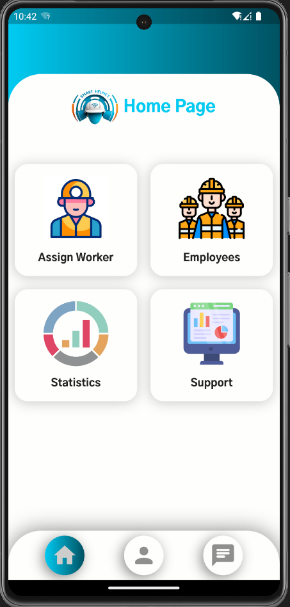
 Sign in page

Figure 24 / supervisor login (web)

Figure 25 / supervisor login (mobile)

As we mentioned before, the user chooses to be a supervisor and enters his data so that he can move to the home page.

 Home page

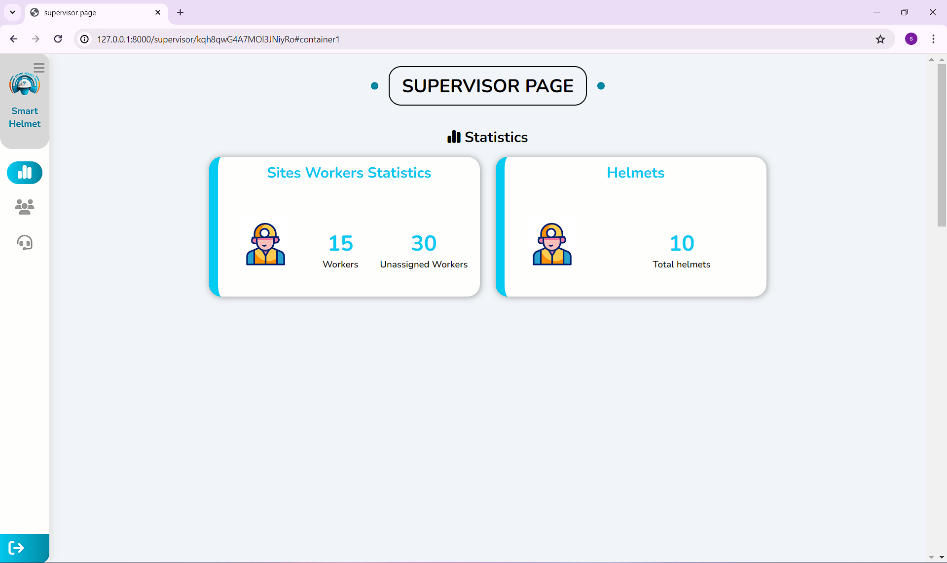
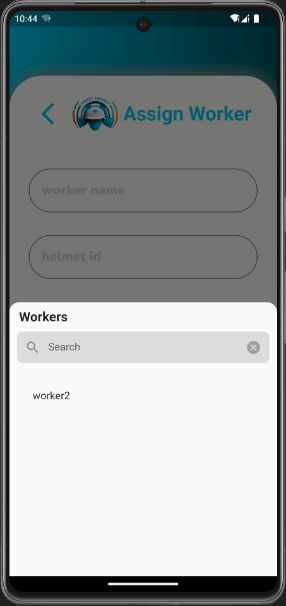


Figure 26 / supervisor home (web)

Figure 27 / supervisor home (mobile)

The supervisor's home page is not much different from the manager's, but the difference is in the sections it contains, for example, workers, support , …...

The page also shows helmet statistics and their number.

 Assign worker with helmet page

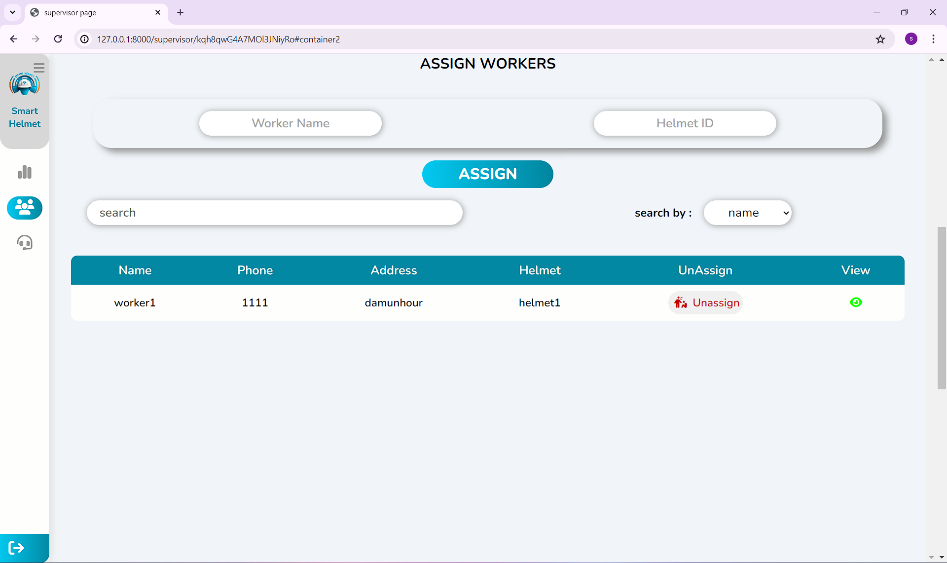
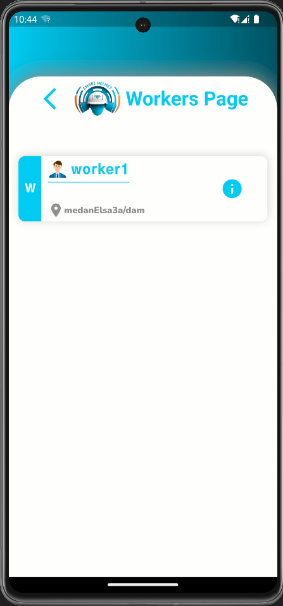


Figure 28 / assign page (web)

Figure 29 / assign page (mobile)

The Assign worker with helmet page is considered a very important page, as the supervisor links each worker to a helmet through its ID, which the manager puts when adding the helmet to the system.

 Workers page

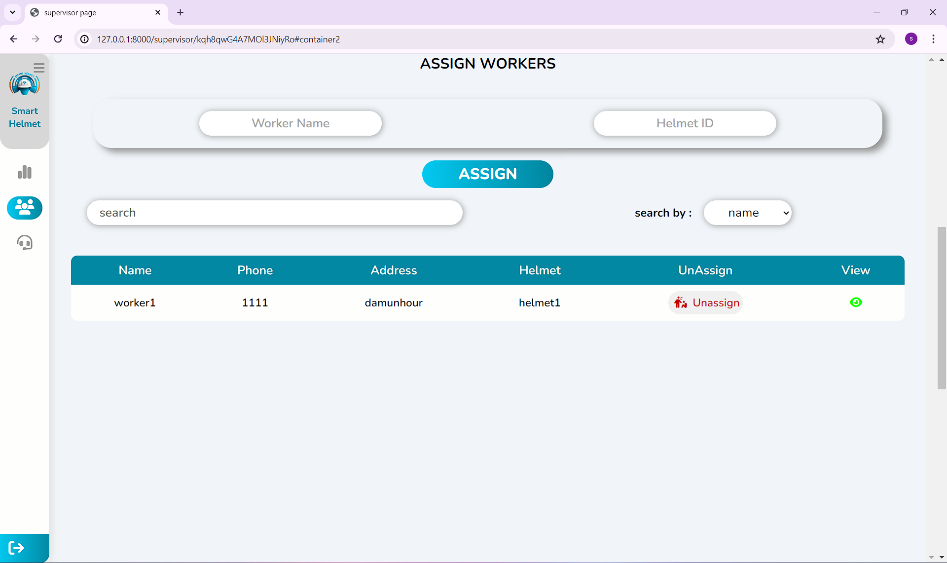
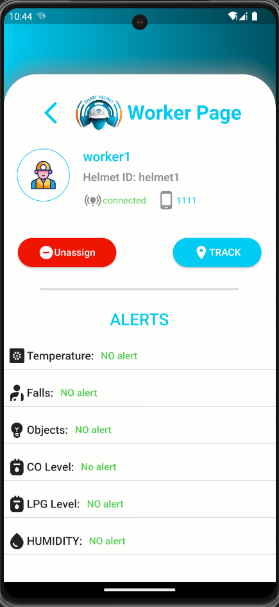


Figure 30 / workers page (web)

Figure 31 / workers page (mobile)

The workers page displays the workers, their number, and the worker’s status, whether he is connected or not connected.

 Each worker page

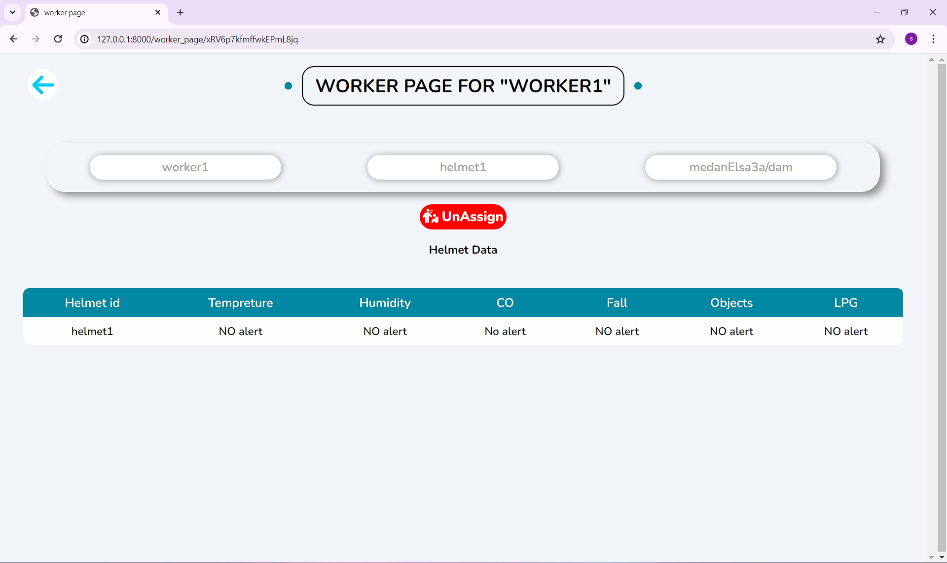


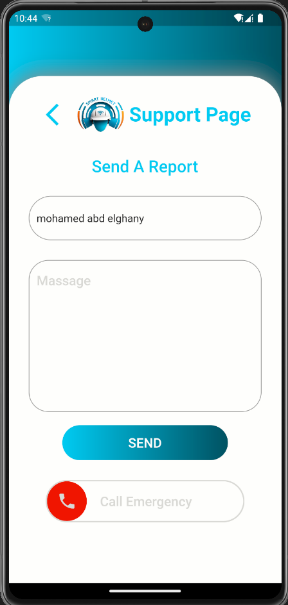
Figure 32 / worker page (web)

Figure 33 / worker page (mobile)

After viewing the workers page, if the supervisor wants to see more information about the worker or even remove him from the workplace,

he can click on see more.

Through this page, he can also see whether there is an alarm or not.

 Support page

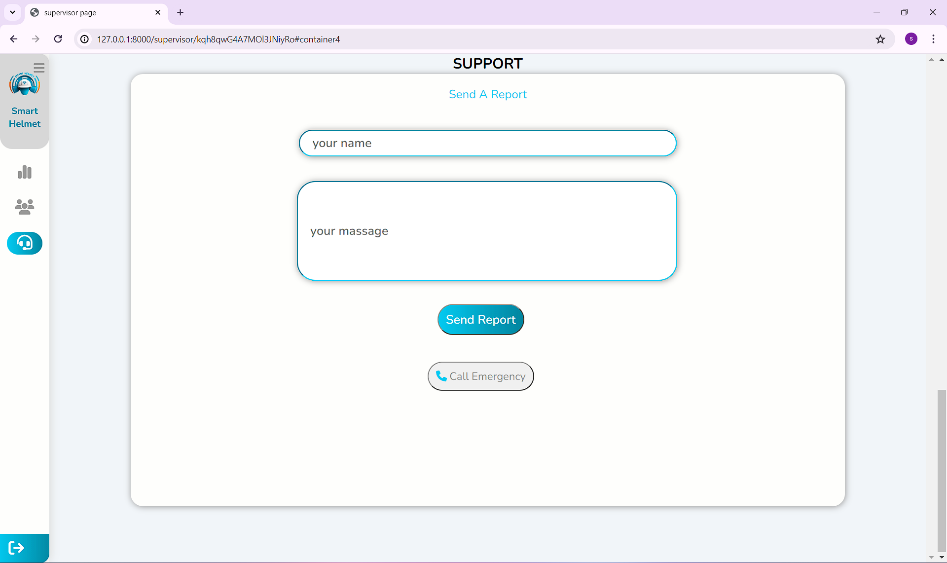


Figure 34 / support page (web)

Figure 35 / support page (mobile)

Through the support page, the manager can send reports or call emergency services in case of danger.

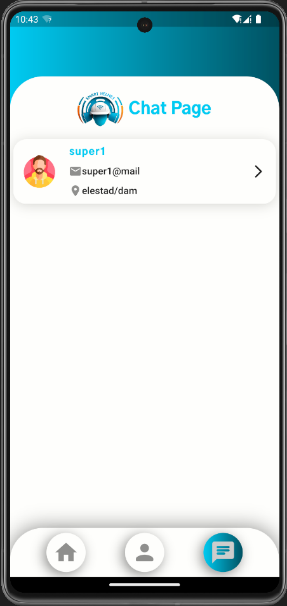
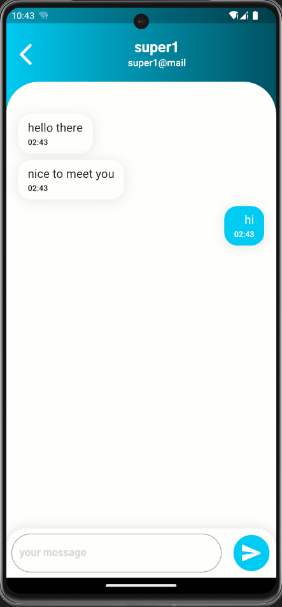
 chat pages

Figure 36 / chat page (mobile)

Figure 37 / chats page (mobile)

The chat pages displays the all supervisors for each site so they can communicate with each other to ask for supports or any other services.

It also provides a real-time chat with notifications to ease the communication between them.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Software Interface

• Server: python anywhere.

• Database: firebase.

• Tool: django, javascript, c language.

• Platform: vs code, android studio.

Hardware Interface

For pc:

• RAM: minimum of 512 MB.

• Hard Disk: any space.

• Processor: dual core processor.

For mobile:

• RAM: minimum of 2 GB of ram.

• Hard Disk: minimum of 1 GB free space.

• OS: at least android 8.

Communication Interface

• Django uses URL to communicate between pages of the website.

• Flutter uses session variables to communicate between pages of the application.

• The user must connect to the Internet to access the Website and the application for the communication purpose.

• Cost-Effective Solution: Despite its advanced features, the ESP32 remains cost-effective, making it an attractive choice for both hobbyist and commercial projects. Its affordability, combined with its rich feature set and performance capabilities, has contributed to its widespread adoption in various IoT applications, including home automation, industrial monitoring, wearable devices, and more.

**4.3 Hardware Design**

**Overview**

As we mentioned at the beginning, the safety of workers is very important in the workplace due to the amount of risks to which they may be exposed. Therefore, safe equipment must be provided and the durability of this equipment must be monitored. Since safety helmets are almost a necessary requirement in all industries, our project aims to develop mandatory safety equipment in a way that allows the supervisor to monitor workers and the working environment through a mobile application or website.

Our smart safety helmet is equipped with multiple sensors controlled by a microcontroller that constantly senses changes and sends them to the application or website. This will enable supervisors to better monitor their workers and take action if necessary.

The helmet is designed to sense changes in the work environment in terms of temperature, humidity, and some gases that can be very harmful if inhaled by workers in high concentrations and can even lead to death, track workers’ locations at work sites, and sense when a foreign body approaches workers or in cases of falls. When abnormal readings are sensed in the work environment, it alerts the worker and notifies the supervisor through the mobile application so that he is aware of the problem and can then take countermeasures.

The causes of most work-related deaths in industry include electrocution, falls, and collision with objects. In accidents such as falls, especially if from great heights or collisions with heavy objects, the timing of medical care provided to the worker plays a critical role in determining the worker's chance of survival.

Ensuring the possibility of providing the necessary medical care to workers in a timely manner. Our smart safety helmet is also designed to detect if a worker falls, if an object falls on them, or if they are hit by an object from above, and notify the supervisor of the possibility of such incidents along with the worker's location and in case medical attention is required, they will be able to send quickly.

**Microcontroller**

The ESP32 is a versatile microcontroller chip developed by Espressif Systems, a semiconductor company based in Shanghai, China. It's widely recognized for its powerful processing capabilities, built-in wireless connectivity options, and low power consumption.

The ESP32 comes equipped with built-in Wi-Fi and Bluetooth capabilities. Utilize Wi-Fi connectivity to establish communication between the smart safety helmets and the central monitoring system, whether it's a mobile application or a web-based dashboard. Bluetooth can be used for short-range communication between helmets and nearby devices, facilitating features such as location tracking and proximity detection.

**Why ESP32**

We chose the ISP 32 because, firstly, it contains built-in Wi-Fi and Bluetooth, and this will help us a lot in achieving the project goals, in addition to many features, which are:

1. Dual-Core Processor: One of the key features of the ESP32 is its dual-core Xtensa LX6 microprocessor, which operates at up to 240 MHz. Having two cores allows for multitasking and simultaneous execution of multiple tasks, making the ESP32 suitable for handling complex applications requiring real-time processing.

2. Wireless Connectivity: The ESP32 integrates both Wi-Fi (802.11 b/g/n) and Bluetooth (Bluetooth Low Energy or BLE) connectivity options, making it ideal for Internet of Things (IoT) projects requiring wireless communication capabilities. It supports a wide range of Wi-Fi security protocols, including WPA, WPA2, and WEP, as well as Bluetooth profiles such as A2DP, SPP, and BLE Mesh.

3. Low Power Consumption: Despite its powerful performance, the ESP32 is designed to operate efficiently in low-power scenarios, making it suitable for battery-powered and energy-conscious applications. It offers multiple low-power modes, including deep sleep, light sleep, and modem sleep, which help minimize power consumption during periods of inactivity.

4. Rich Peripheral Set: The ESP32 features a wide array of peripherals, including GPIO (General-Purpose Input/Output) pins, UART (Universal Asynchronous Receiver-Transmitter) interfaces, SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), ADC (Analog-to-Digital Converter), DAC (Digital-to-Analog Converter), and more. These peripherals enable interfacing with various sensors, actuators, displays, and other external devices.

5. Security Features: Security is a priority for IoT devices, and the ESP32 includes features to enhance data security and integrity. It supports hardware-accelerated encryption algorithms such as AES (Advanced Encryption Standard) and SHA (Secure Hash Algorithm), as well as secure boot and flash encryption mechanisms to protect against unauthorized access and firmware tampering.

6. Security Features: Security is a priority for IoT devices, and the ESP32 includes features to enhance data security and integrity. It supports hardware-accelerated encryption algorithms such as AES (Advanced Encryption Standard) and SHA (Secure Hash Algorithm), as well as secure boot and flash encryption mechanisms to protect against unauthorized access and firmware tampering.

7. Cost-Effective Solution: Despite its advanced features, the ESP32 remains cost-effective, making it an attractive choice for both hobbyist and commercial projects. Its affordability, combined with its rich feature set and performance capabilities, has contributed to its widespread adoption in various IoT applications, including home automation, industrial monitoring, wearable devices, and more.



Chapter 5

System implementation



smart safety Helmet team

Faculty of computers and information

**5.1 Mobile Application:**

**5.1.1 Introduction**

The implementation phase is a critical stage in the development of the mobile application, where the theoretical design is translated into a functional and user-friendly product. This section of the graduation book delves into the intricate process of building the mobile app, covering the tools, technologies, and methodologies employed to bring the project to fruition.

The mobile implementation part begins with a detailed overview of the development environment, including the programming languages, frameworks, and integrated development environments (IDEs) utilized. Understanding the technological foundation is essential as it sets the stage for the subsequent coding and integration efforts.

Next, we explore the structural design of the application, outlining the architecture that supports its functionality. This includes a discussion on the Model-View-Controller (MVC) or other architectural patterns adopted to ensure a robust and maintainable codebase. Each component of the architecture is examined, highlighting its role and interaction with other parts of the system.

Following the architectural overview, the part delves into the development process, presenting a step-by-step account of how each feature was implemented. Key modules and algorithms are described in detail, providing insights into the logic and reasoning behind their development. This section not only demonstrates the technical skills applied but also showcases the problem-solving approaches used to overcome challenges encountered during implementation.

Additionally, the integration of backend services and third-party APIs is covered. This includes the implementation of features such as user authentication, data storage, and real-time updates, all of which are essential for a seamless user experience. The challenges of integrating these services and the solutions devised are discussed to provide a comprehensive view of the development process.

In conclusion, the mobile implementation chapter not only documents the technical execution of the project but also reflects on the learning and growth experienced during the development journey. This detailed account serves as a testament to the technical proficiency and problem-solving capabilities demonstrated throughout the project.

**5.1.2 Connection with database**

Connecting Flutter App to Firebase

Firebase is a comprehensive platform that provides various tools and services to help you develop high-quality apps. In this section, we will detail the steps to connect a Flutter app to Firebase using Dart.

1. Setting Up Firebase

Before writing any code, you need to set up Firebase for your Flutter project.

Step 1: Create a Firebase Project

Go to the Firebase Console.

Click on "Add project" and follow the prompts to create a new Firebase project.

Step 2: Register Your App with Firebase

In the Firebase Console, click on "Add app" and select the Android or iOS platform based on your project.

Follow the instructions to register your app. You'll need to provide the package name (for Android) or the bundle identifier (for iOS).

Download the google-services.json (for Android) or GoogleService-Info.plist (for iOS) and place it in the appropriate directory of your Flutter project.

Step 3: Add Firebase SDK

Adding all SDKs that we need to communicate with firebase in pubspec.yaml file:

We used the terminal to install all packages that have been used in our project throw getting its commands from “ pub.dev ” website and after installing this packages it will be added into pubspec.yaml file.

2. Initializing Firebase in Your Flutter App

To initialize Firebase in your Flutter app, you need to configure it in the main entry point of your app which is the file called “ main.dart ”, this file is responsible of launching the mobile app and initializing all dependencies that your app need to perform its job that is built to do.

Future<void> main() async {

  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(

    options: DefaultFirebaseOptions.currentPlatform,

  );

  //other functions that need to be called .

  runApp(MyApp(prefs: prefs));

}

**5.1.3 Dependencies**

What is dependencies ?

Dependencies is the packages that is used in the project to perform its job ,its been installed throw the terminal or adding the version of this package into the pubspec.yaml file manually, this is all dependencies in our mobile app :

dependencies:

  flutter:

    sdk: flutter

  cupertino\_icons: ^1.0.2

  firebase\_auth: ^4.16.0

  firebase\_core: ^2.24.2

  cloud\_firestore: ^4.14.0

  awesome\_dialog: ^3.2.1

  drop\_down\_list: ^0.0.6

  connectivity: ^3.0.6

  shared\_preferences: ^2.2.3

  firebase\_messaging: ^14.6.5

  url\_launcher: ^6.2.4

  firebase\_analytics: ^10.8.0

  http: ^1.1.0

  flutter\_local\_notifications: ^17.1.2

  audioplayers: ^5.2.1

  permission\_handler: ^11.0.1

  flutter\_sound: ^9.2.13

  record\_mp3: ^3.0.0

  firebase\_storage: ^11.6.0

  intl: ^0.19.0

  get: ^4.6.6

  fluttertoast: ^8.2.5

in this dependencies there are the packages of communicating with firebase like:

**1.firebase\_auth:** this package is responsible of authentication functions as login and logout using many options such as google authentication or facebook or github or ordinary username and password ‘in our project we used the ordinary auth using username and password’.

**2.firebase\_core:** this package is responsible of connecting the mobile app with firebase database as it gets the database id ‘from the firebase\_options.dart file which contains all configs to access firebase database’.

**3.cloud\_firestore:** this package is responsible of getting all data stored on the database, in NOSQL firebase database the data is stored in the form of collections and each collection has its documents where the data is stored, this data may be string, Boolean or number.

**4.firebase\_messaging:** this package is responsible of sending notifications from the database to mobile app when it receives a request throw an api (thunder client), this notifications is being sent in two methods: (appToken or Topic).

**5.firebase\_analytics:** this package is responsible of analytics such as getting the total numbers of documents in a collection, for example: getting the total number of workers in the collection called workers who has the same site name.

**6.firebase\_storage:** this package is responsible of storing images, videos or audios and creating a url to use it when you need to call this image, video or audio.

There is another dependency to show dialogs to user and deliver messages to him, this package is:

**Awesome\_dialog:** this package is responsible of make the application interactive with users throw send them a dialogs with messages in different processes , there is multiple types of dialogs like error , info , success and warning each type has its animation and used in different situations.

There is another dependency to check the connection of the device when the user use our application , this package is:

**Connectivity:** this package is responsible of checking the connection status of the device such as ( mobile network / wifi ) because the application needs an internet connection to access the database , if there is no connection the app shows awesomedialog to tell the user that there is no internet connection and ask him to retry connecting to internet .

There is another dependency to make a selection widget to use in our app in the cases that we need to make the user select something such as ‘assigning worker with helmet ’,in this function the app needs to show all available workers and helmets so the user can assign each one worker with one helmet, this package is:

**Drop\_down\_list:** this package is responsible of show a menu to user and in this menu the data that we fetch into this menu so user can interact with this menu and use the data that he wants to use with the availability of searching in this menu data to find his option that he is looking for.

There is another dependency to define the URLs that we used in the application ,this package is :

**Https:** this package is responsible of defining the URL that we use in the application such as API’s URLs so the app can determine if that url is a string or link.

There is another dependency to launch the URLs after defining it using the previous package in the application ,this package is :

**Url\_launcher:** this package is responsible of launch the URL that we use in the application such as API’s URLs so the app can communicate with this api with the tasks to do.

There is another dependency to take permissions from the user to access his device for higher performance, this package is :

**Permission\_handler:** this package is responsible of asking user for permissions like notification access , run in background access and get to camera or mic access.

There is another dependency to run notifications when the app is terminated, this package is :

**flutter\_local\_notifications:** this package is responsible of run the app in background after taking the permission and show notifications even if the application is terminated.

There is another dependency to run a sound when some processes happens ‘sending message / receiving notifications’, this package is :

**Flutter\_sound:** this package is responsible of run sound when the user sends a message or receive notifications, this sound can be defined throw put the sounds in assets and make the app access this sounds.

There is another dependency to get variables from other files in the app, this package is :

**Get :** this package is responsible of getting variables from other files to use in the file we want like importing it .

**5.1.4 Login**

This part includes the funcions of login for supervisors and managers :

1.Manager login:

To make the function of manager login we used firebase authentication by username and password as it is a built in function available in firebase to ease the process of singing in.

class \_LoginMState extends State<LoginM> {

  GlobalKey<FormState> formState = GlobalKey<FormState>();

  TextEditingController email = TextEditingController();

  TextEditingController password = TextEditingController();

 @override

Widget build(BuildContext context) {

if (formState.currentState!.validate()) {

        try {

// ignore: unused\_local\_variable

             final credential = await FirebaseAuth.instance

             .signInWithEmailAndPassword(

             email: email.text.toLowerCase().trim(),

             password:password.text.toLowerCase().trim());

  await Navigator.pushReplacementNamed(context, "/homeM");

           } on FirebaseAuthException catch (e) {

               AwesomeDialog(

                context: context,

                  animType: AnimType.scale,

                dialogType: DialogType.error,

                  title: 'ERROR',

                  desc: 'No user found for that email..',

            ).show();

              print('No user found for that email.');

              if (e.code == 'user-not-found') {

              } else if (e.code == 'wrong-password') {

              AwesomeDialog(

                  context: context,

                    animType: AnimType.scale,

                dialogType: DialogType.error,

                     title: 'ERROR',

                  desc: 'Wrong password provided for that user..',

                  ).show();

            print('Wrong password provided for that user.');

              }};};

According to this code we used validation to ensure that text fields are not empty then it checks if the username and password that the manager enters are in database or not ,if this process output is positive the manager can login and go to the manager home page ,else a dialog appears to tell the manager that the username or password is invalid.

This dialog ‘awesomedialog’ is a package had been installed in the dependencies to use in the cases of error , warning ,information or success .

We used this package in every process that need to deliver a message to the user ,so it can be considered as an interactive action.

2.Supervisors login:

To make the function of supervisor login we built a function to make supervisor able to login by username and password as each supervisor has amout of data so their data stored in firestore so we couldn’t use firebase authentication.

class \_LoginState extends State<Login> {

  final TextEditingController \_usernameController = TextEditingController();

  final TextEditingController \_passwordController = TextEditingController();

  GlobalKey<FormState> formState = GlobalKey<FormState>();

  // ignore: unused\_field

  Map<String, dynamic>? \_supervisorData;

  Future<void> \_login() async {

    try {

      final username = \_usernameController.text;

      final password = \_passwordController.text; // Hash password

      final supervisorCollection =

          FirebaseFirestore.instance.collection('supervisors');

      final querySnapshot = await supervisorCollection

          .where('email', isEqualTo: username)

          .where('password', isEqualTo: password) // Check hashed password

          .get();

      final documentSnapshot = querySnapshot.docs.first.reference;

      if (querySnapshot.docs.isNotEmpty) {

        \_supervisorData = querySnapshot.docs.first.data();

        if (\_supervisorData?['site'] != "") {

          final prefs = await SharedPreferences.getInstance();

      await prefs.setString('supervisorData',jsonEncode(\_supervisorData));

          // Login successful (handle navigation or further actions)

          String? mytokenVal = await FirebaseMessaging.instance.getToken();

      await documentSnapshot.update({'superToken': mytokenVal,});

          print('Login successful!');

          Navigator.pushReplacementNamed(context, "/home");

        } else {

          AwesomeDialog(

            context: context,

            animType: AnimType.scale,

            dialogType: DialogType.error,

            title: 'ERROR',

            desc: 'this username iisnot assigned to site .',

          ).show();

        }

      } else {

        AwesomeDialog(

          context: context,

          animType: AnimType.scale,

          dialogType: DialogType.error,

          title: 'ERROR',

          desc: 'Invalid username or password.',

        ).show();

        // Login failed (show error message)

        print('Invalid username or password.');

      }

    } catch (e) {

      AwesomeDialog(

        context: context,

        animType: AnimType.scale,

        dialogType: DialogType.error,

        title: 'ERROR',

        desc: 'Invalid username or password.',

      ).show(); // Handle errors

    }

  }

This function looks like the manager login function but instead of accessing the firebase auth we access the fire store collection called ‘supervisors’ and check if the username and password are correct or not ,and if the data user entered is right the app gets the signed in device ‘token’ so the app use this token in other processes such as sending notifications , but according to the device token is so important and sensitive data we programmed a function to encrypt the device token and store it in firestore and when the app needs this token it must decrypt the token before using it.

**5.1.5 Stream builder function**

This function is responsible of fetching the data in page in real-time so any update happen in the database will be fetched in the page without need to refresh this page.

We used this function in every page needs to be real-time (worker page / chat page / reports page / statistics page / ..etc).

Code example for this function:

  Stream<DocumentSnapshot> \_getModelDataStream() {

    if (\_documentId == null) {

      return const Stream

          .empty(); // Return an empty stream if no document ID is found

    }

    return FirebaseFirestore.instance

        .collection('AI\_model')

        .doc(\_documentId)

        .snapshots();

  }

In this example we used stream funcrion to get all statistics of the site that our AI model detects and fetch this data continuingly without need to refresh the page.

 StreamBuilder<DocumentSnapshot>(

       stream: \_getModelDataStream(),

       builder: (context, snapshot) {

       if (snapshot.connectionState == ConnectionState.waiting) {

         return const Center(child: CircularProgressIndicator());

       }

       if (snapshot.hasError) {

         return Center(child: Text('Error: ${snapshot.error}'));

       }

       if (!snapshot.hasData || !snapshot.data!.exists) {

         return const Center(child: Text('No data found'));

       }

       final data = snapshot.data!.data() as Map<String, dynamic>?;

       final detectedHelmets = data?['detected\_helmets'];

…………………..//the structure of the page and designing if the page

After building the stream function we used the ‘stream.builder’ function to show the data from the function into the page body.

We defined the stream to the name of the function we built and the builder to snapshot so it can get any data from stream function and fetch it in the page body in real-time .

The function must be dynamic so the stream builder widget can get the data from this function.

**5.1.6 Chat function**

Chat functions includes many processes to ensure sending and receiving messages successfully ,this processes are :

1.fetch supervisors in page body:

  Future<void> \_fetchSupervisorData() async {

    final prefs = await SharedPreferences.getInstance();

    final encodedData = prefs.getString('supervisorData');

    if (encodedData != null) {

      \_supervisorData = jsonDecode(encodedData) as Map<String, dynamic>;

    }

    setState(() {}); // Update UI with retrieved supervisor data

  }

  @override

  void initState() {

    \_fetchSupervisorData();

    super.initState();

  }

In this future we fetch the signed in supervisor data to use this data to hide his name from the chats page.

  Map<String, dynamic>? \_supervisorData;

  Stream<QuerySnapshot<Object?>> getUsersDataStream() {

    return FirebaseFirestore.instance.collection('supervisors').snapshots();

  }

We used this stream to get all documents in supervisors collection.

StreamBuilder<QuerySnapshot<Object?>>(

stream: getUsersDataStream(),

builder: (context, AsyncSnapshot<QuerySnapshot> snapshot) {

if (snapshot.hasError) {

return Text('Error: ${snapshot.error}');

}

if (snapshot.hasData) {

final data = snapshot.data!.docs;

if (data.isEmpty) {

return const Center(child: Text('No chats'));

}

return SizedBox(

height: 500,

child: ListView.builder(

itemCount: snapshot.data!.docs.length,

itemBuilder: (context, index) {

final name = snapshot.data!.docs[index]['name'];

final email = snapshot.data!.docs[index]['email'];

final site = snapshot.data!.docs[index]['site'];

if (email != currentSupervisorEmail &&site != "") {

…………//fetch data in page body

}

In this code we used steam builder to get the data from the future ‘getUsersDataStream’ and check if there is no data it returns no chats text and if there is data it returns the supervisors in cards except the current signed in user that uses the device and send messages and the supervisors that not assigned to a site as this chat is built to communicate between supervisors in their sites to ask for supplies or any other things between the site’s supervisors.

2. send message:

class Message {

  final String senderID;

  final String senderEmail;

  final String receiverID;

  final String message;

  final String date;

  final String time;

  final Timestamp timestamp;

  Message({

    required this.senderID,

    required this.senderEmail,

    required this.receiverID,

    required this.message,

    required this.timestamp,

    required this.date,

    required this.time,

  });

  Map<String, dynamic> toMap() {

    return {

      "SenderID": senderID,

      "SenderEmail": senderEmail,

      "receiverID": receiverID,

      "message": message,

      "timeStamp": timestamp,

      "Date": date,

      "Time": time,

    };

  }

}

Map<String, dynamic>? \_supervisorData;

Future<void> sendMessages(String receiverID, message) async {

  final prefs = await SharedPreferences.getInstance();

  final encodedData = prefs.getString('supervisorData');

  if (encodedData != null) {

    \_supervisorData = jsonDecode(encodedData) as Map<String, dynamic>;

  }

  final \_currentUserID = \_supervisorData?['name'];

  final \_currentUserEmail = \_supervisorData?['email'];

  final String currentUserID = "$\_currentUserID";

  final String currentUserEmail = "$\_currentUserEmail";

  final Timestamp timestamp = Timestamp.now();

  final DateTime now = DateTime.now();

  int year = now.year;

  int month = now.month;

  int day = now.day;

  int hour = now.hour + 1;

  int minute = now.minute;

  String formattedDate =

      "${day.toString().padLeft(2, '0')}/${month.toString().padLeft(2, '0')}/${year}";

  String formattedTime =

      "${hour.toString().padLeft(2, '0')}:${minute.toString().padLeft(2, '0')}";

  Message newMessage = Message(

      senderID: currentUserID,

      senderEmail: currentUserEmail,

      receiverID: receiverID,

      message: message,

      timestamp: timestamp,

      date: formattedDate,

      time: formattedTime);

  List<String> ids = [currentUserID, receiverID];

  ids.sort();

  String chatRoomID = ids.join("\_");

  await FirebaseFirestore.instance

      .collection('chat\_rooms')

      .doc(chatRoomID)

      .collection("messages")

      .add(newMessage.toMap());

}

First we created the structure of the message which contains senderID, receiverID, timestamp, message content and sender name.

Then we created ‘sendMessages’ future to get the sender name and id and takes the message content and receiverID from the request when the sender pressed the send button in this code:

onPressed: () async {

if (formstate.currentState!.validate()) {

QuerySnapshot querySnapshot = await FirebaseFirestore.instance.collection('supervisors').where('name', isEqualTo: widget.receiverID).get();

late String receiverToken = "";

if (querySnapshot.docs.isNotEmpty) {

final DocumentSnapshot documentSnapshot =

querySnapshot.docs.first;

final Map<String, dynamic>? data =

documentSnapshot.data() as Map<String, dynamic>?;

if (data != null && data.containsKey('superToken')) {

setState(() {

receiverToken = data['superToken'] as String;

print(receiverToken);

});

} else {

print('Field "superToken" not found or is null');

}}

final String messageContent = messagecontroller.text;

sendNotificationForMessage(

messageContent, receiverToken, senderName);

sendMessage();

getMessagesStream(senderName, widget.receiverID);

}

3.get messages:

  Stream<List<Map<String, dynamic>>> getMessagesStream(

      String userID, otherUserID) {

    List<String> ids = [userID, otherUserID];

    ids.sort();

    String chatRoomID = ids.join("\_");

    final messagesRef = FirebaseFirestore.instance

        .collection('chat\_rooms')

        .doc(chatRoomID)

        .collection("messages")

        .orderBy('timeStamp', descending: false);

    return messagesRef.snapshots().map((snapshot) {

      final messages = snapshot.docs

          .map((doc) => {

                ...doc.data(), // Include all existing data fields

                'id': doc.id, // Add the document ID

              })

          .toList();

      return messages;

    });

  }

This stream function defines the steps to build it as we used this function in chat page to get all messages between the sender and receiver only by their id

child: ChatBubble(

textalign: isCurrentUser

? CrossAxisAlignment.end

: CrossAxisAlignment.start,

In this code checks if the will show on right or left of the page to give a specified design.

4.message info & delete message:

onlongpress: () {

final MessageID =

"${messages[index]['id']}";

isCurrentUser

? AwesomeDialog(

context: context,

animType: AnimType.scale,

dialogType: DialogType.info,

title: 'message info',

desc:'date:${messages[index]['Date']}',

btnOkIcon:

Icons.delete\_rounded,

btnOkText: ' ',

btnCancelText: 'OK',

btnOkColor:const Color(0xfff00f00),

btnCancelColor:const Color(0xff02CBF2),

btnCancelOnPress: () {},

btnOkOnPress: () {

deleteMessage(

MessageID,

senderName,

widget.receiverID);

},

).show()

: AwesomeDialog(

context: context,

animType: AnimType.scale,

dialogType: DialogType.info,

title: 'message info',

desc:'date:${messages[index]['Date']}',

btnCancelText: 'OK',

btnCancelColor:

const Color(0xff02CBF2),

btnCancelOnPress: () {},

).show();},

On long press on the message bubble it checks if this message was sent by the current user or not.

If this message sent be this user, he can see the info of this message like date and time also he can delete this message, else if he didn’t send this message he can see the info of the message but he can’t delete this message.

**5.1.7 Notifications function**

This function is responsible of sending notification request to the database throw thunder client api.

We used this function to send notification to supervisors when there is an alert from their workers helmets or they received a new message.

sendNotifications(mytoken, messageTitle, messageBody) async {

  var headersList = {

    'Accept': '\*/\*',

    'User-Agent': 'Thunder Client (https://www.thunderclient.com)',

    'Content-Type': 'application/json',

    'Authorization':

        'key=AAAAdgXpk7o:APA91bHaU1GMUcSze-UN7vxnsjRHFyrdXh-d3NeSYCid9alxT-kGLJtBLVSG4BTxMHOYDAVXFHpL3miwGR8fJY4RlTcN6\_VWSu0jV3Nn2uR3UzHNAmW2zuZcy3vIzUc3V4HSn4AlPfQA'};

  var url = Uri.parse('https://fcm.googleapis.com/fcm/send');

  var body = {

    "to": mytoken,

    "notification": {

      "title": messageTitle,

      "body": messageBody,

      "mutable\_content": true,

      "sound": "Tri-tone"

    },

    "data": {

      "url": "<url of media image>",

      "dl": "<deeplink action on tap of notification>"

    }

  };

  var req = http.Request('POST', url);

  req.headers.addAll(headersList);

  req.body = json.encode(body);

  var res = await req.send();

  final resBody = await res.stream.bytesToString();

  if (res.statusCode >= 200 && res.statusCode < 300) {

    print(resBody);

  } else {

    print(res.reasonPhrase);

  }}

This code is the api that receives the token, message content and title of notification

Then it sends the request to firebase to send the notification to the device that have the same token.

getNotifications() async {

  String? mytokenVal = await FirebaseMessaging.instance.getToken();

  final helmetsCollection = FirebaseFirestore.instance.collection('helmets');

  final snapshotStream = helmetsCollection.snapshots();

  await for (final snapshot in snapshotStream) {

    for (final doc in snapshot.docs) {

      final workerName = doc.get('owner');

      final helmetName = doc.get('helmetID');

      final coValue = doc.get('co');

      final lpgValue = doc.get('lpg');

      final tempValue = doc.get('temp');

      final humidityValue = doc.get('humidity');

      final fallValue = doc.get('fall');

      // final objectsValue = doc.get('objects');

      if (coValue != null) {

        if (coValue == "alert") {

          sendNotifications(mytokenVal, 'CO Alert from $helmetName',

              "worker($workerName)'s helmet detected a high level of CO");

          print('co alert');

        }

      }

      if (lpgValue != null) {

        if (lpgValue == "alert") {

          sendNotifications(mytokenVal, 'LPG Alert from $helmetName',

              "worker($workerName)'s helmet detected a high level of LPG");

          print('LPG alert');

        }

      }

      if (tempValue != null) {

        if (tempValue == "alert") {

          sendNotifications(mytokenVal, 'Temprature Alert from $helmetName',

              "worker($workerName)'s helmet detected a high level of Temperature");

          print('temp alert');

        }

      }

      if (humidityValue != null) {

        if (humidityValue == "alert") {

          sendNotifications(mytokenVal, 'Humidity Alert from $helmetName',

              "worker($workerName)'s helmet detected a high level of Humidity");

          print('humidity alert');

        }

      }

      if (fallValue != null) {

        if (fallValue == "alert") {

          sendNotifications(mytokenVal, 'fall Alert from $helmetName',

              "worker($workerName)'s helmet detected a fall of worker");

          print('fall alert');

       }}}}}

This code is the future that checks helmets data and if any field of any helmet sends alert the future sends the data throw api that contacts with firebase to send message to supervisor.

void sendNotificationForMessage(

    String messageContent, String receiverToken, String senderName) async {

  if (messageContent != '') {

    sendNotifications(receiverToken, 'you have a new message from $senderName',

        messageContent);

    print('notification sent');

  }

}

This future works when calling it in sending message so it takes the sender name ,message content and receiver token then it uses the api to contact with firebase and send notification to the receiver supervisor by his token.

We called this two send notification functions in each function that requires notification.

**In the conclusion**, we presented the main and important parts of mobile implementation that we needed to focus on.

**5.2 Website**

**5.2.1 Overview**

The website serves as a vital tool for managers, offering a comprehensive suite of functionalities tailored to streamline operations and enhance managerial oversight. While supervisors also have access, the primary focus lies on empowering managers with robust tools to efficiently manage resources and monitor performance.

The manager section of the website is thoughtfully divided into two distinct areas, each catering to different aspects of managerial responsibilities:

1. Smart Helmet:

Within this section, managers gain access to a wealth of insightful data regarding workforce activities, site operations, and equipment management. Upon logging in, managers can seamlessly navigate through various statistics, offering valuable insights into worker performance, site productivity, supervisor activities, and the status of deployed smart helmets. Moreover, managers can effortlessly review reports submitted by supervisors, providing valuable feedback and facilitating informed decision-making.

2. Financial Management:

This section serves as a financial nerve center, empowering managers to effectively oversee and manage the financial resources of their organization, be it a construction company or any other enterprise. Here, managers have the tools to meticulously manage payroll, track purchases, monitor expenses, and analyze income streams. Additionally, the financial management section offers detailed reports and visualizations, allowing managers to gain a comprehensive understanding of their organization's financial health. Whether assessing monthly expenditures, evaluating yearly profits, or planning budget allocations, managers can rely on this section to make data-driven decisions that drive business success.

By offering these functionalities, the website equips managers with the tools they need to optimize operations, ensure regulatory compliance, and drive organizational growth. From workforce management to financial oversight, the website serves as an indispensable asset for managers striving for excellence in their roles.

**5.2.2 Databases used**

Within the website setup, we've incorporated two types of database systems to handle our data needs efficiently. Let's take a closer look at each one and why we've chosen to use them:

1. Firebase:

We've opted for Firebase's Firestore database for the Smart Helmet section of our website. Firestore excels in providing real-time data synchronization, which perfectly suits our need for instant updates. With Firestore, we can easily manage data related to workers, supervisors, reports, helmets, chat logs, and AI model data. This ensures that our information is always up-to-date and consistent across all devices.

To connect our website with Firebase, we've utilized Firebase's API along with some private keys and configurations. This setup involves installing necessary libraries and packages within our Python environment to enable smooth communication between our website and the Firebase backend.

2. SQLite:

In the financial management section of our website, we've employed SQLite as our database solution. SQLite is lightweight and straightforward, making it a great fit for handling structured financial data like income records, employee details, expenses, purchases, and net profit calculations.

SQLite operates on a language called SQL, which makes it easy for us to manage table structures without getting bogged down in complex programming. This simplicity allows our team to focus on analyzing financial data and making informed decisions.

We've seamlessly integrated SQLite into our website using the Django framework, ensuring a secure and reliable connection between our application and the SQLite database.

In summary, by utilizing Firebase and SQLite databases, we're able to efficiently manage our data needs across different sections of our website while ensuring real-time updates and easy data analysis.

**5.2.3 Backend requirements**

Actually there is more than 65 library installed in the project put here is the most important 7:

Django (Framework) == 3.2.23 the python web framework used .

firebase-admin (Library) == 6.5.0 SDK for interacting with Firebase services from server-side.

google-cloud-firestore (Library) == 2.16.0 Client library for Firestore,,database for web from Google.

SQLite (Library) transactional SQL database engine.

channels (Library) == 4.0.0 Asynchronous support for Django, including WebSockets.

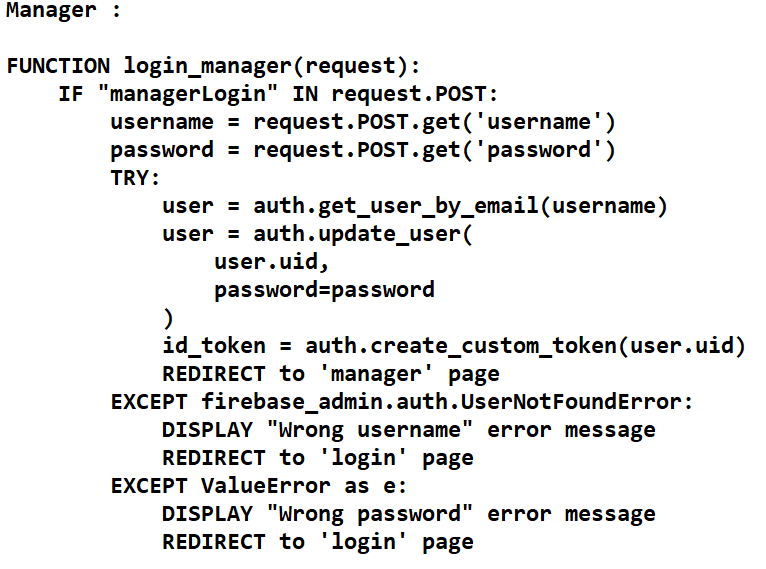
httplib2 (Library) == 0.22.0 Comprehensive HTTP client library for Python.

redis (Library) == 5.0.4 Python client for Redis, an in-memory data structure store.

**5.2.4 Backend authentication**

The authentication mechanism employed within the website exhibits notable similarities to its mobile counterpart. Its principal function is to facilitate the login process for both managers and supervisors. Leveraging the Firebase Authentication system, managers are authenticated using a combination of username and password, whereas supervisors utilize their email credentials, which are incorporated into the database upon managerial instantiation of supervisor accounts. Upon successful authentication, managers are directed to the main manager page, constituting the primary interface for managerial functionalities. Conversely, supervisors are seamlessly redirected to a dedicated supervisor page, uniquely identified by the supervisor's associated identifier (ID). This stratified approach ensures the segregation of managerial and supervisory functionalities, thereby optimizing user experience and system integrity.

Here is a here's a pseudo code representation for the two login functions :

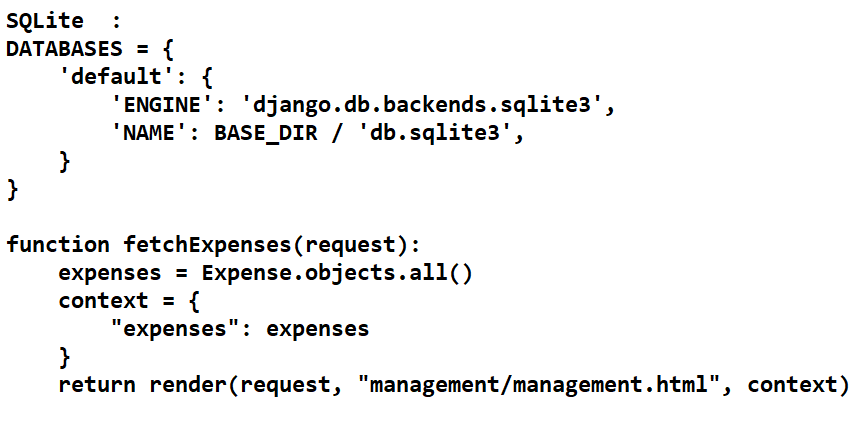


**5.2.5 Fetch data functionalities**

Section 5.2.5 delves into the intricacies of data retrieval functionalities within the web environment. As previously elucidated, data retrieval occurs from two distinct databases. While the fundamental logic governing this retrieval process remains consistent, nuances arise in the syntactic constructs utilized when interfacing with Firebase or SQLite databases. These differences stem from the distinctive query languages inherent to each database system.

Here is a small example in how to fetch workers and expenses data in the two databases :





**5.2.6 CRUD functionalities**

CRUD stands for Create, Read, Update, and Delete, and it represents the four basic functions of persistent storage. These operations are commonly used in database management systems, and they form the backbone of most data-driven applications. Let's break down each operation:

Create: This operation involves the creation of new data records in a database. It typically involves inserting new data into the database.

Read: The read operation involves retrieving data from the databases. This operation allows users to access and view existing data stored in the databases. Reading data can involve querying the database for specific records or retrieving all records that meet certain criteria..

Update: This operation involves modifying existing data in the database. It allows user to make changes to data that already exists.

Delete: The delete operation involves removing data from the database. It allows users to eliminate records that are no longer needed or that contain incorrect information.

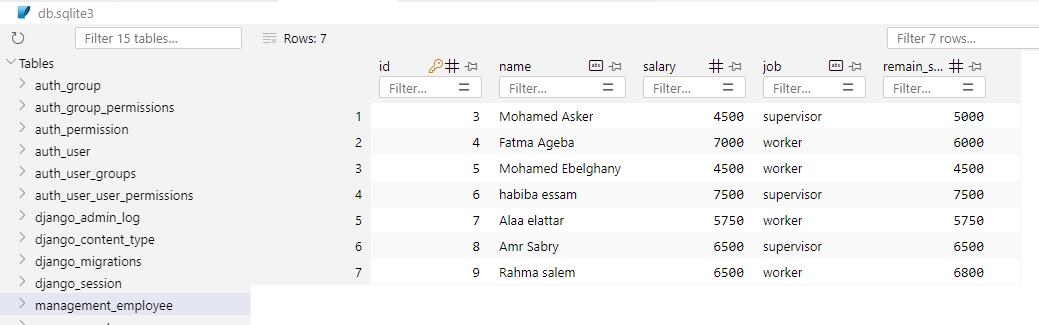
Together, these four operations provide a comprehensive set of functionalities for managing data in a database. They are fundamental to the development of applications that rely on persistent storage, and they are used extensively in various domains, including web development, mobile app development, and enterprise software development.

Figure 38/ SQLite records

**5.2.6 WebSocket Real-time**

This section will explore the procedures for configuring real-time data retrieval from Firebase within a Django framework. Django, by default, lacks built-in real-time capabilities. Hence, I had to manually set up an open connection between the server and the database. This was achieved through the utilization of "WebSocket" technology. To facilitate this process, I employed "Django Channels" to establish WebSocket connections instead of relying on the traditional HTTP protocol, which involves a request and response mechanism.

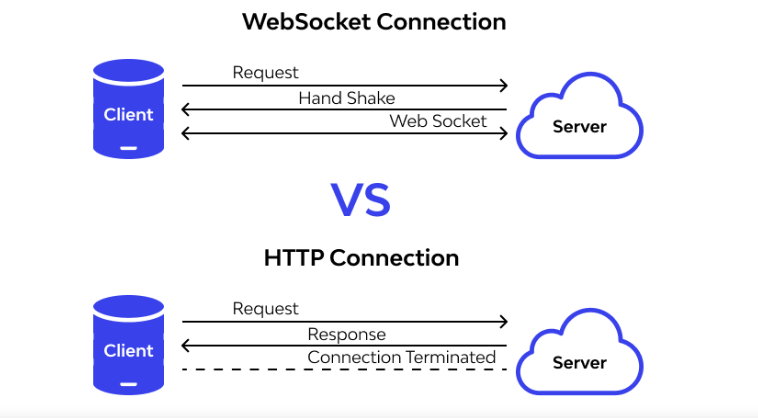


Figure 39/ Comparison between websocket & HTTP

And here is a small brief of how this function works :

1. Setting up the Environment: Initially, it imports necessary modules and sets the DJANGO\_SETTINGS\_MODULE environment variable.

2. Routing Configuration: The ProtocolTypeRouter is configured to handle both HTTP and WebSocket protocols. For WebSocket connections, it employs the AuthMiddlewareStack along with a URLRouter to direct requests to appropriate consumers based on URL patterns.

3. URL Configuration: The websocket\_urlpatterns list contains URL patterns for WebSocket connections. In this case, it defines a path for WebSocket connections to a worker page, with the worker's ID included in the URL.

4. WebSocket Consumer: The WorkerPageConsumer class is a WebSocket consumer responsible for handling WebSocket connections to the worker page. It connects to a specific document in the Firestore database corresponding to the worker's ID and listens for changes using the on\_snapshot method. When changes occur, it sends the updated worker data to the client.

5.Connection Management: The connect method is called when a client establishes a WebSocket connection. It accepts the connection, subscribes to changes in the worker document, and initiates the data exchange. The disconnect method is invoked when the connection is closed, and it unsubscribes from further updates. The receive method is not utilized in this context and remains empty.

**5.2.7 Financial Analysis and statistics**

Manager Function:

This function retrieves various data related to the management of workers, helmets, supervisors, sites, and reports from a Firestore database.

It calculates statistics such as the total number of helmets, workers, supervisors, and sites, as well as the number of unassigned helmets and workers.

Additionally, it computes the total salary of all workers.

Finally, it constructs a context containing all the retrieved data and statistics, and renders a template named "manager.html" with this context.

Management Function:

This function is responsible for generating financial reports and statistics based on expenses and incomes within a specific time frame.

It first calculates the start and end dates of the current month and year.

Then, it retrieves all expenses and incomes from the database and calculates various financial metrics such as total salaries, total incomes, total expenses, and total profit.

It further calculates monthly and yearly incomes, expenses, and profits.

Additionally, it counts the number of workers and supervisors.

Finally, it constructs a context containing all the calculated financial metrics and statistics, and renders a template named "management.html" with this context.

In summary, these functions are crucial for financial analysis and management within the project. They provide insights into various financial aspects such as expenses, incomes, profits, and workforce statistics, enabling effective decision-making and monitoring of financial health.

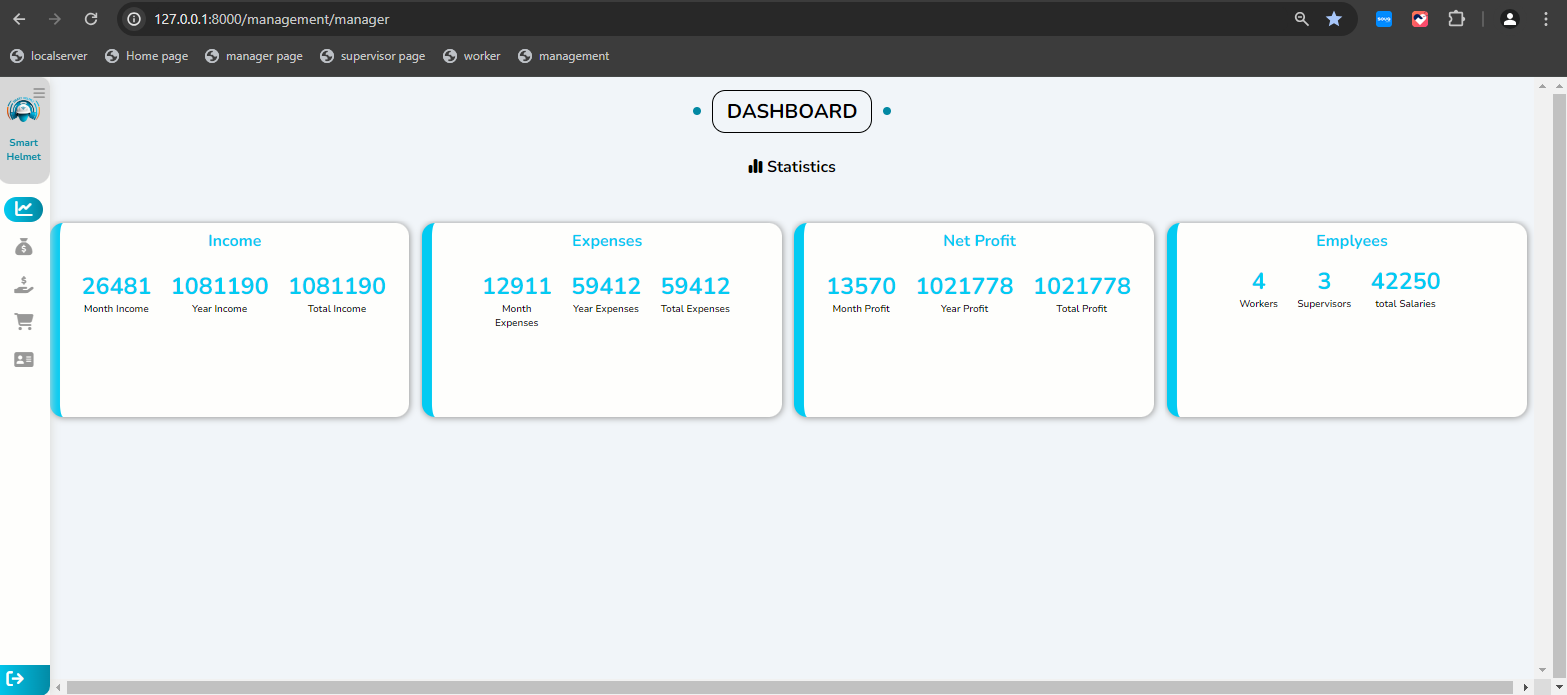


Figure 40/ Financial Statistics



Chapter 6

Helmet implementation



smart safety Helmet team

Faculty of computers and information

**6.1 Temperature and Humidity**

**6.1.1 Overview of the Sensor**

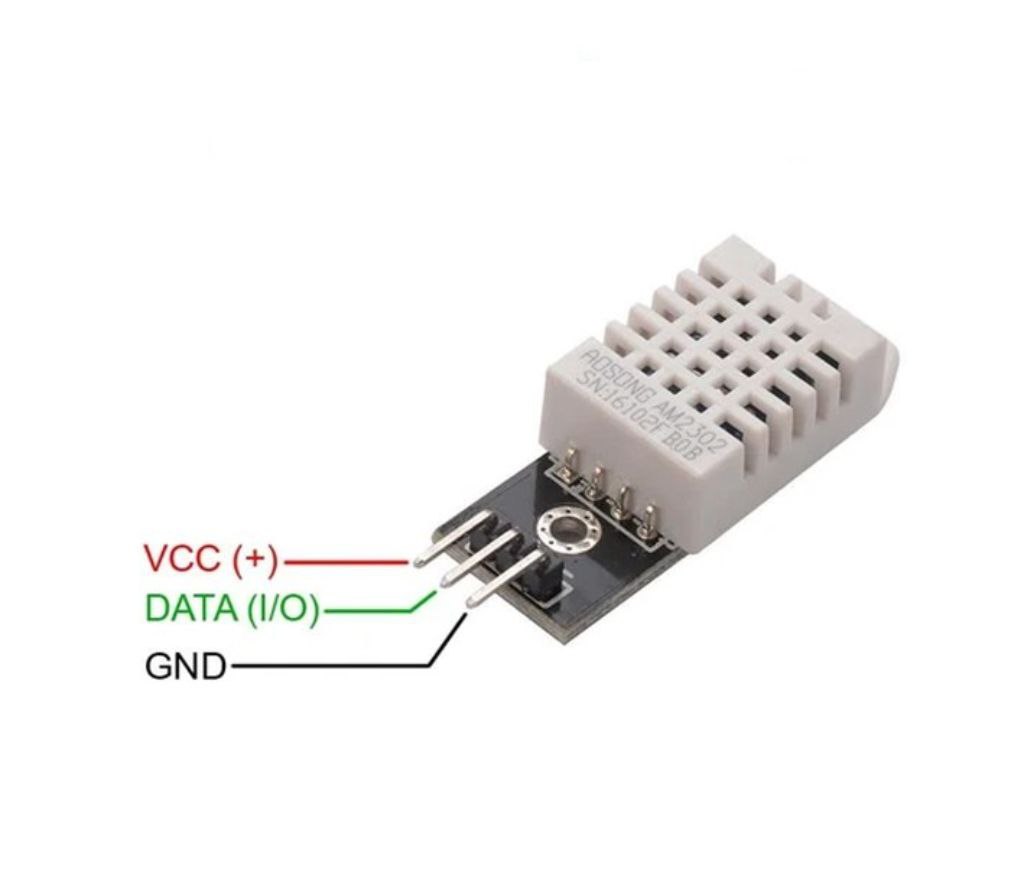


Figure 41 / DTH22 sensor pinout

The DHT22 sensor, is a widely used digital sensor for measuring temperature and humidity. It is known for its accuracy, reliability, and ease of integration into microcontroller projects. Below are the key features and technical details of the DHT22 sensor.

**Key Features**

1. **Temperature Measurement:**
   * Range: -40 to 80 °C
   * Accuracy: ±0.5 °C
   * Resolution: 0.1 °C
2. **Humidity Measurement:**
   * Range: 0 to 100% RH
   * Accuracy: ±2-5% RH
   * Resolution: 0.1% RH
3. **Output:**
   * Digital signal via a single-wire protocol
   * Easy to interface with microcontrollers such as Arduino, Raspberry Pi, etc.
4. **Power Supply:**
   * Operating Voltage: 3.3 to 6V
   * Low power consumption: 2.5mA max during measurement
5. **Response Time:**
   * Temperature: ≤ 2 seconds
   * Humidity: ≤ 5 seconds
6. **Size:**
   * Compact size making it easy to embed in various projects

**Pin Configuration**

The three pins on your DHT22 sensor are likely configured as follows:

1. **VCC (VDD):** Power supply (3.3V to 6V)
2. **Data:** Combined Data and Power pin
3. **GND:** Ground

**Working Principle**

The DHT22 sensor uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and it outputs a digital signal on the data pin. The data is transferred using a proprietary single-wire protocol that requires precise timing to decode.

**Communication Protocol**

The communication protocol between the DHT22 sensor and the microcontroller involves:

1. **Initialization:** The microcontroller sends a start signal to the sensor.
2. **Response:** The sensor responds with a low signal followed by a high signal.
3. **Data Transmission:** The sensor sends 40 bits of data, consisting of 16 bits for humidity, 16 bits for temperature, and 8 bits for a checksum.

**Pros and Cons**

**Pros:**

* High accuracy for both temperature and humidity
* Simple to interface with microcontrollers
* Wide operating range

**Cons:**

* Slower response time compared to some other sensors
* Requires careful timing to read data correctly

**6.1.2 Wiring diagram and Working method**

**6.1.2.1 Wiring diagram**

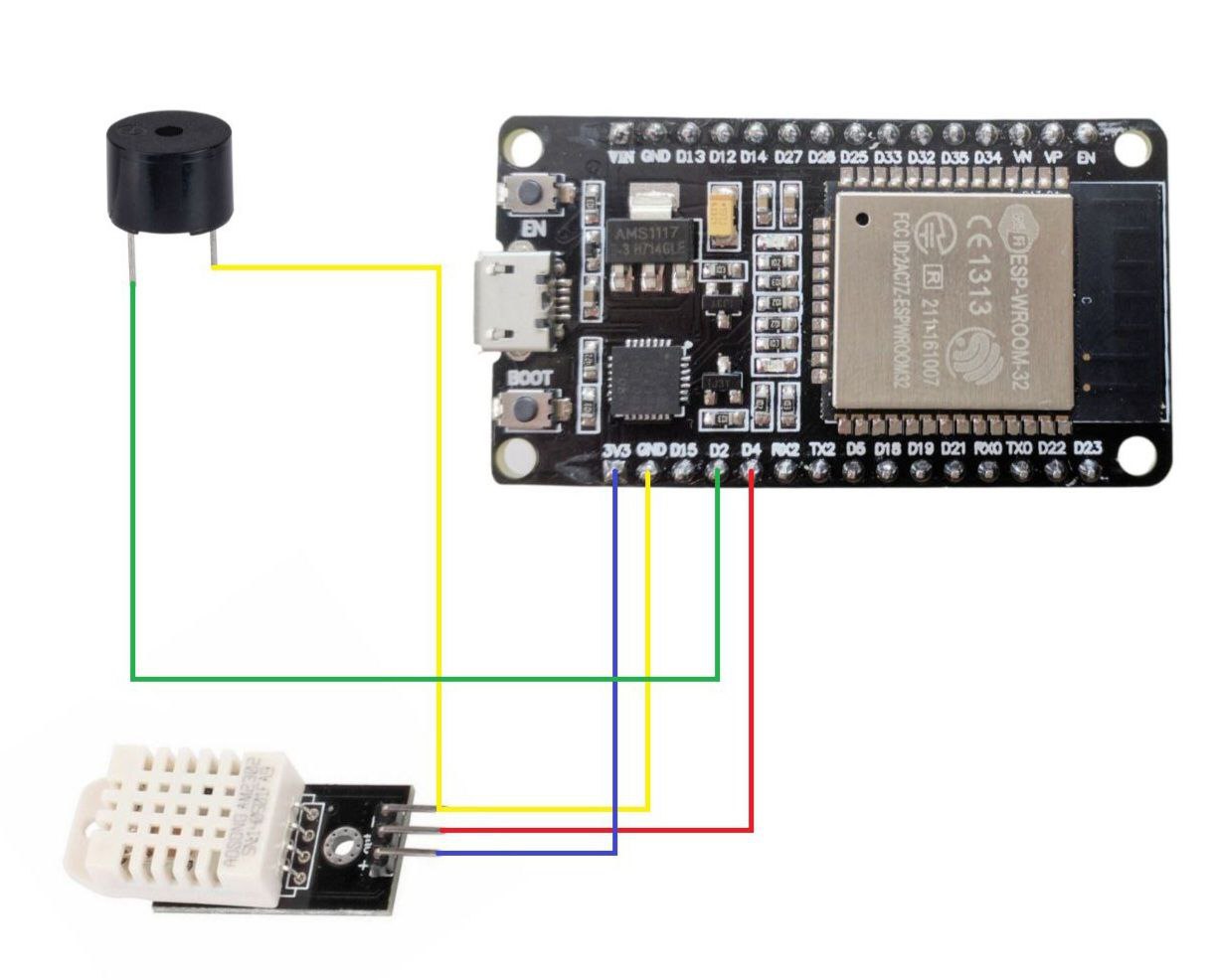


Figure 42 / DHT22 & ESP 32

The image shows a wiring diagram connecting a DHT22 sensor to an ESP32 microcontroller board. Here's a detailed explanation of the connections and how to set up the sensor with the ESP32:

**DHT22 Sensor Pinout:**

1. **VCC:** Power supply (3.3V to 5V)
2. **Data:** Digital signal output
3. **GND:** Ground

**ESP32 Pinout:**

* **3V3:** 3.3V power supply
* **GND:** Ground
* **GPIO Pins (D4, D2, etc.):** General-purpose input/output pins

**Connections:**

* DHT22 VCC (Red wire) to ESP32 3V3
* DHT22 Data (Yellow wire) to ESP32 GPIO D4
* DHT22 GND (Black wire) to ESP32 GND

**Additionally, there is a buzzer connected to the ESP32:**

* Buzzer Positive (Yellow wire) to ESP32 GPIO D2
* Buzzer Negative (Green wire) to ESP32 GND

The Occupational Safety and Health Administration (OSHA) provides guidelines to ensure comfortable and safe indoor environments, particularly for workplaces. However, it doesn't prescribe exact temperature and humidity ranges. Instead, OSHA refers to recommendations made by other organizations, such as the American Society of Heating, Refrigerating, and Air-Conditioning Engineers (ASHRAE). Here's a summary of these recommendations:

**Temperature Range**

* **Winter:** 68°F to 76°F (20°C to 24.4°C)
* **Summer:** 68°F to 78°F (20°C to 25.6°C)

**Humidity Range**

* **Relative Humidity:** 20% to 60%

**6.1.2.2 working method**

1. **Sensor Measurements:**
   * The DHT22 sensor takes readings of the temperature and humidity every 2 seconds. This continuous monitoring allows for real-time tracking of environmental conditions.
2. **Threshold Comparison:**
   * After obtaining the temperature and humidity readings, the system compares these values to predefined threshold values. These thresholds represent acceptable ranges for temperature and humidity. If the readings fall within these thresholds, it indicates that the environment is within the desired conditions.
3. **Displaying Normal Readings:**
   * If the temperature and humidity readings are within the acceptable range, a normal text message is displayed in the mobile application. This message likely informs the user that the conditions are normal and no action is required.
4. **Sending Alerts for Out-of-Range Readings:**
   * If the temperature or humidity readings fall outside the predefined thresholds, an alert is triggered. This alert is sent to a supervisor or relevant personnel to notify them of the out-of-range readings. The alert indicates that there may be an issue with the environmental conditions that require attention.

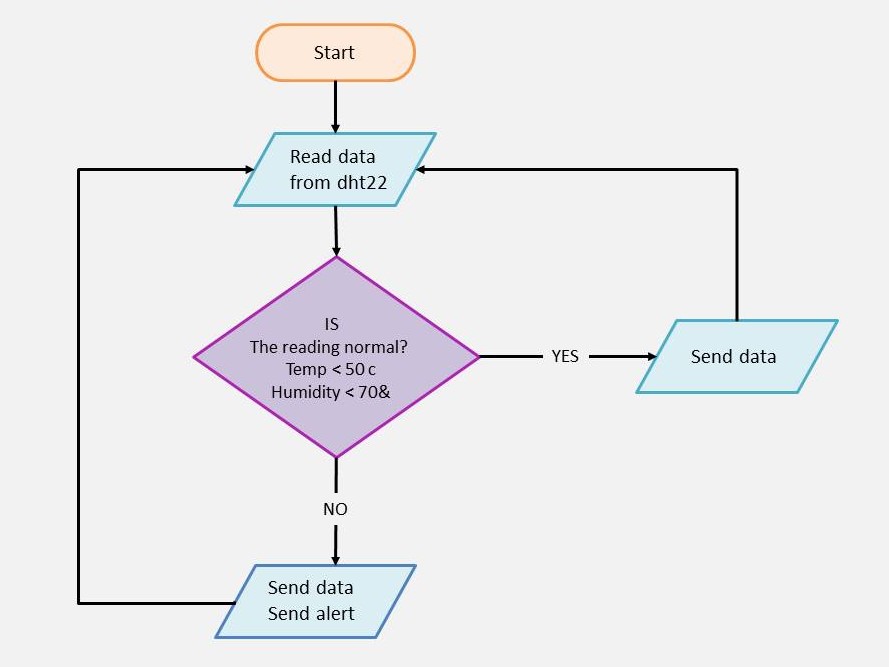


Figure 43 / Temperature and Humidity flowchart

**6.1.3 Code and test result**

**6.1.3.1 Code**

#include <DHT.h>

#define DHTPIN 4          // Digital pin connected to the DHT sensor

#define DHTTYPE DHT22     // DHT 22

DHT dht(DHTPIN, DHTTYPE);

#define BUZZERPIN 2      // Digital pin connected to the buzzer

* **Include the DHT Library:** which provides functions to interact with DHT sensors. You need to have this library installed in your Arduino IDE. It facilitates reading temperature and humidity from the sensor.
* **Define Sensor Pin and Type:**
  + **DHTPIN 4:** This defines a constant DHTPIN set to 4, indicating that the DHT22 sensor's data pin is connected to digital pin 4 on the microcontroller.
  + **DHTTYPE DHT22:** This defines a constant DHTTYPE set to DHT22, specifying the type of DHT sensor you are using. In this case, it's the DHT22.
* **Define Buzzer Pin:**
  + **BUZZERPIN 2:** This defines a constant BUZZERPIN set to 2, indicating that the buzzer is connected to digital pin 2 on the microcontroller. This pin will be used to control the buzzer, turning it on or off based on certain conditions (like out-of-range temperature or humidity).

void setup() {

  Serial.begin(9600);

  pinMode(BUZZERPIN, OUTPUT);

  dht.begin();

}

The setup function runs once when the microcontroller is powered on or reset. Its purpose is to initialize settings and configurations that are needed before entering the main loop. In this case, it prepares the serial communication, sets up the buzzer pin, and initializes the DHT sensor.

Once the setup function is complete, the microcontroller will move on to the loop function, which contains the main code that will run repeatedly.

1. **Serial Communication Initialization:**
   * **Purpose:** initializes the serial communication at a baud rate of 9600 bits per second.
   * **Explanation:** Serial communication allows the microcontroller to send and receive data to and from the computer via the USB connection. The **9600** is a common baud rate, which defines the speed of communication. This is essential for debugging and monitoring sensor data through the Serial Monitor in the Arduino IDE.
2. **Buzzer Pin Configuration:**
   * **Purpose:** This line sets the mode of the **BUZZERPIN** (which is pin 2) to output.
   * **Explanation:** In Arduino, pins can be configured as either inputs or outputs. By setting the **BUZZERPIN** to **OUTPUT**, you configure the microcontroller to send signals to this pin, which will control the buzzer. When the pin is set to **HIGH**, the buzzer will turn on, and when set to **LOW**, the buzzer will turn off.

**3. DHT Sensor Initialization:**

* + **Purpose:** This line initializes the DHT sensor.
  + **Explanation:** The dht.begin() function starts the DHT sensor, preparing it to read temperature and humidity data. This is necessary before you can take any readings from the sensor. The dht object was previously created and configured with the pin number and sensor type (DHT22).

void loop() {

  float humidity = dht.readHumidity();            // Read humidity from DHT22

  float temperature = dht.readTemperature();      // Read temperature from DHT22

  failed(humidity, temperature, "DHT22 sensor");  // Call failed function

  Serial.print("Humidity: ");

  Serial.print(humidity);

  Serial.print("%  ");

  check(humidity, "Humidity", 70);                // Call check function for humidity

  Serial.print("Temperature: ");

  Serial.print(temperature);

  Serial.print("°C  ");

  check(temperature, "Temperature", 50);          // Call check function for temperature

delay(1000); // Delay for 1 second between each measurement }

The loop function runs continuously on the microcontroller. It performs the main tasks of the program, such as reading sensor data, processing it, and taking actions based on the data.

1. **Reading Sensor Data:**
   * **Purpose:** These lines read the humidity and temperature values from the DHT22 sensor.
   * **Explanation:** The dht.readHumidity() and dht.readTemperature() functions retrieve the current humidity and temperature readings from the sensor. These values are stored in the variables humidity and temperature.
2. **Calling the failed Function:**
   * **Purpose:** This line calls the failed function to check if the sensor readings are valid.
   * **Explanation:** The failed function likely checks if the readings from the sensor are NaN (Not a Number), indicating a failure in reading the sensor. It might also handle error logging or retries. The string "DHT22 sensor" is passed to identify which sensor's data is being validated.
3. **Printing Humidity:**
   * **Purpose:** These lines print the humidity reading to the Serial Monitor.
   * **Explanation:** The Serial.print functions output the humidity value and its unit (%) to the Serial Monitor for debugging and monitoring purposes.
4. **Calling the check Function for Humidity:**
   * **Purpose:** This line calls the check function to evaluate if the humidity reading is within acceptable limits.
   * **Explanation:** The check function likely compares the humidity value to a threshold (70% in this case). If the humidity exceeds this threshold, the function might trigger an alert or take other actions. The string "Humidity" is passed to identify the type of data being checked.
5. **Printing Temperature:**
   * **Purpose:** These lines print the temperature reading to the Serial Monitor.
   * **Explanation:** The Serial.print functions output the temperature value and its unit (°C) to the Serial Monitor for debugging and monitoring purposes.
6. **Calling the check Function for Temperature:**
   * **Purpose:** This line calls the check function to evaluate if the temperature reading is within acceptable limits.
   * **Explanation:** The check function likely compares the temperature value to a threshold (50°C in this case). If the temperature exceeds this threshold, the function might trigger an alert or take other actions. The string "Temperature" is passed to identify the type of data being checked.
7. **Adding a Delay:**
   * **Purpose:** This line adds a delay of 1000 milliseconds (1 second) between each measurement cycle.
   * **Explanation:** The delay(1000) function pauses the execution of the program for 1 second. This ensures that sensor readings are taken at regular intervals, preventing the sensor from being polled too frequently and allowing for more meaningful data collection. It also helps to avoid potential issues with sensor overheating or erroneous readings due to rapid consecutive reads.

void failed(float value1, float value2, String sensor) {

  if (isnan(value1) || isnan(value2)) {

    Serial.println("Failed to read from " + sensor);

  }

}

**Failed Function**

* **Purpose:** Checks if the sensor readings are valid by verifying that they are not NaN (Not a Number).
* **Implementation:** If either **value1** or **value2** is NaN, it prints an error message indicating that the sensor reading failed.

void check(float value, String Name, int max) {

  if (isnan(value)) {

    Serial.println("Failed to read " + Name + " level");

    return;

  }

  if (value > max) {

    Serial.println("High " + Name + " level! Danger");

    buzzer();   // Call buzzer function

  } else {

    Serial.println(Name + " level is normal");

  }

}

**Check Function**

* **Purpose:** Validates the sensor reading against a predefined threshold and takes appropriate actions based on the reading.
* **Implementation:**
  + If the reading is NaN, it prints an error message and returns.
  + If the reading exceeds the threshold (max), it prints an alert message indicating a high level and calls the buzzer function to activate the buzzer.
  + If the reading is within acceptable limits, it prints a message indicating normal levels.

void buzzer() {

  digitalWrite(BUZZERPIN, HIGH);

  delay(1000);  // Buzz for 1 second

  digitalWrite(BUZZERPIN, LOW);

}

**Buzzer Function**

* **Purpose:** Activates the buzzer for 1 second to signal an alert.
* **Implementation:**
  + Sets the buzzer pin high to turn the buzzer on.
  + Delays for 1 second to keep the buzzer on.
  + Sets the buzzer pin low to turn the buzzer off.

This complete code provides a continuous loop that reads temperature and humidity from the DHT22 sensor, checks if the readings are valid, prints them to the Serial Monitor, compares them against predefined thresholds, and activates a buzzer if the readings exceed those thresholds.

**6.1.3.2 Test Result**

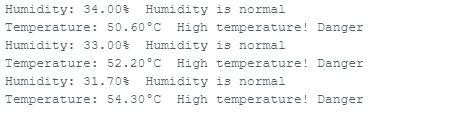


Figure 44 / Test temperature and Humidity

1. **Humidity:** 34.00% Humidity is normal
   * **Humidity:** 34.00%, which is within the normal range.
   * **Status:** "Humidity is normal."
2. **Temperature:** 50.60°C High temperature! Danger
   * **Temperature:** 50.60°C, which is above the safe threshold.
   * **Status:** "High temperature! Danger."
   * **Action:** The system should trigger an alert due to the high temperature.
3. **Humidity:** 33.00% Humidity is normal
   * **Humidity:** 33.00%, which is within the normal range.
   * **Status:** "Humidity is normal."
4. **Temperature:** 32.20°C Temperature is normal
   * **Temperature:** 32.20°C, which is within the normal range.
   * **Status:** "Temperature is normal."

So, the output shows a sequence of readings from the DHT22 sensor, indicating normal levels for both humidity and temperature initially, followed by alerts for high humidity and high temperature readings, and then returning to normal levels for humidity.

**6.2 Gas Monitoring**

**6.2.1 Overview of the Sensor**

Figure 45 / MQ5 sensor pinout

The MQ5 sensor is a gas sensor commonly used to detect various gases such as LPG, natural gas, methane, and other combustible gases in the atmosphere. Here's an overview of the MQ5 sensor:

1. **Principle of Operation:** The MQ5 sensor operates on the principle of chemiresistance, where its resistance changes in response to the concentration of the target gas in the air. It contains a sensing element composed of a tin dioxide (SnO2) semiconductor, which reacts with the target gas, causing a change in resistance.
2. **Structure:** The MQ5 sensor typically consists of a sensing element enclosed in a ceramic tube, with electrodes for electrical connections. It requires both a heater coil and a sensing element to function properly.
3. **Heating Element:** The sensor includes a heating coil that is used to heat the sensing element to an optimal temperature for gas detection. This ensures stable and accurate readings.
4. **Detection Range:** The MQ5 sensor is sensitive to a range of combustible gases, including liquefied petroleum gas (LPG), natural gas, and methane. Its sensitivity and response time may vary depending on the gas being detected.
5. **Calibration:** Calibration may be necessary to ensure accurate readings, especially when detecting different types of gases or in varying environmental conditions. This can involve exposing the sensor to known concentrations of the target gas and adjusting its sensitivity accordingly.
6. **Output:** The sensor typically provides an analog output voltage that varies in proportion to the concentration of the detected gas. This output can be interfaced with microcontrollers or other electronic devices for further processing and control.
7. **Limitations:** While the MQ5 sensor is effective for detecting certain gases, it may not be suitable for detecting all types of gases or for use in highly corrosive or extreme environments. Additionally, it may require periodic calibration and maintenance to ensure accuracy and reliability.

The MQ5 gas sensor typically comes with four pins for connection: VCC, GND, Analog Output (AO), and Digital Output (DO). Here's a basic pinout description:

1. **VCC:** This pin is connected to the positive supply voltage (usually 5V) of the microcontroller or power source.
2. **GND**: This pin is connected to the ground of the microcontroller or power source.
3. **Analog Output (AO):** This pin provides an analog voltage output that varies depending on the concentration of the detected gas. The voltage typically increases as the gas concentration increases.
4. **Digital Output (DO):** Some versions of the MQ5 sensor also include a digital output pin that provides a digital signal (usually high or low) based on a predefined threshold level of gas concentration. This can be useful for triggering alarms or other actions when the gas concentration exceeds a certain level.

As for accuracy and range, it's important to note that the accuracy and sensitivity of the MQ5 sensor may vary depending on factors such as the target gas, environmental conditions, and calibration. However, here's a general overview:

1. **Accuracy:** The accuracy of the MQ5 sensor can vary, and it may not provide precise quantitative measurements of gas concentration. Instead, it is commonly used for qualitative detection and indication of the presence of gases.
2. **Range:** The MQ5 sensor is typically sensitive to a range of combustible gases, including liquefied petroleum gas (LPG), natural gas, methane, and other hydrocarbons. The detection range can vary depending on the gas being detected and the specific model of the sensor.
3. **Sensitivity:** The sensitivity of the MQ5 sensor can also vary depending on the gas being detected and environmental conditions. It's important to note that the sensor's sensitivity may degrade over time due to factors such as contamination or aging of the sensing element.
4. **Calibration:** Calibration may be necessary to ensure accurate and reliable detection of gases with the MQ5 sensor. This involves exposing the sensor to known concentrations of the target gas and adjusting its sensitivity or threshold accordingly.

**Pros and Cons**

**Pros:**

1. **Cost-Effective:** The MQ5 sensor is relatively inexpensive compared to some other gas detection technologies, making it accessible for hobbyists, students, and small-scale projects.
2. **Ease of Use:** It's straightforward to interface with microcontrollers or other electronic devices using the analog output, making it suitable for beginners and DIY projects.
3. **Versatility:** The MQ5 sensor is sensitive to a range of combustible gases such as LPG, natural gas, and methane, making it suitable for various gas detection applications.
4. **Compact Size:** The sensor is compact and lightweight, making it easy to integrate into different devices and systems without taking up much space.
5. **Real-Time Monitoring:** It provides real-time monitoring of gas concentrations, allowing for timely detection of gas leaks or hazardous conditions.

**Cons:**

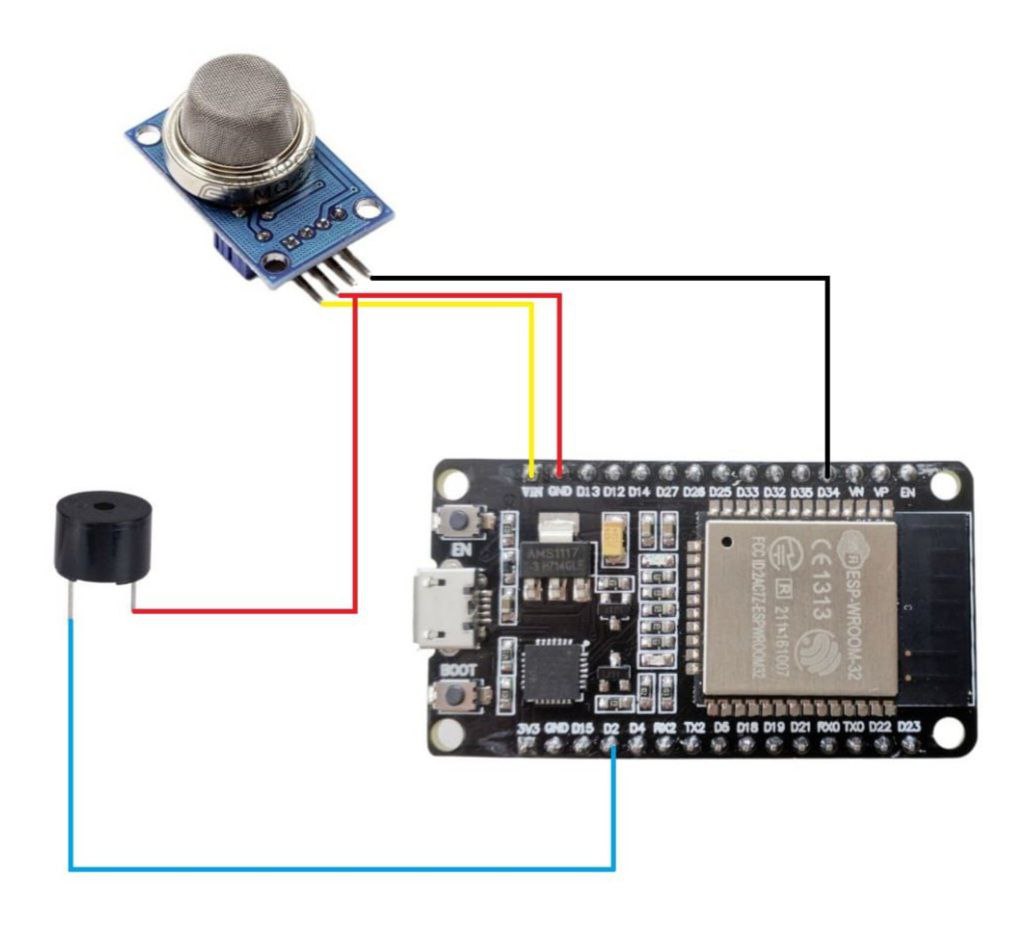
1. **Limited Accuracy:** The MQ5 sensor may not provide highly accurate quantitative measurements of gas concentrations. Its output is often used for qualitative detection rather than precise measurement.
2. **Calibration Requirements:** Calibration may be necessary to ensure accurate and reliable operation, especially when detecting different types of gases or in varying environmental conditions.
3. **Sensitivity to Environmental Factors:** The sensor's sensitivity and performance may be affected by factors such as temperature, humidity, and air flow. It may require shielding or additional calibration to mitigate these effects.
4. **Limited Detection Range:** While sensitive to a range of combustible gases, the MQ5 sensor may not detect certain gases or may have limitations in detecting gases at very low concentrations.

Figure 46 / MQ5 & ESP 32

1. **Lifetime and Maintenance:** The sensor's sensitivity may degrade over time due to factors such as contamination or aging of the sensing element. Regular maintenance and replacement may be required to maintain optimal performance.

**6.2.2 Wiring diagram and Working method**

**6.2.2.1 Wiring diagram**

**Components:**

1. MQ Gas Sensor
2. Buzzer
3. ESP32 Microcontroller

**Connections:**

**MQ Gas Sensor:**

* **VCC (Red wire):**
  + **Connection:** Connect to the VIN pin on the ESP32.
  + **Purpose:** Powers the gas sensor with the required voltage.
* **GND (Black wire):**
  + **Connection:** Connect to one of the GND (Ground) pins on the ESP32.
  + **Purpose:** Provides a common ground reference for the circuit.
* **OUT (Yellow wire):**
  + **Connection:** Connect to an analog GPIO pin (e.g., D34) on the ESP32.
  + **Purpose:** Sends the sensor's analog output signal (proportional to gas concentration) to the ESP32 for reading.

**Buzzer:**

* **Positive (Red wire):**
  + **Connection:** Connect to a digital GPIO pin (e.g., D15) on the ESP32.
  + **Purpose:** Receives a high/low signal from the ESP32 to turn the buzzer on/off.
* **Negative (Blue wire):**
  + **Connection:** Connect to one of the GND (Ground) pins on the ESP32.
  + **Purpose:** Completes the circuit for the buzzer.

**6.2.2.2 Working method**

**Overview:**

The system continuously monitors the gas concentration in the environment using the MQ sensor. When the gas concentration exceeds a set threshold, the buzzer is activated to alert the user.

**Detailed Steps:**

1. **Power Up:**
   * When the ESP32 is powered on (e.g., via USB or an external power source), it supplies 3.3V to the VCC pin of the MQ sensor, powering it up.
   * The buzzer is also connected to the ESP32 but remains off until activated by the ESP32.
2. **Gas Detection:**
   * The MQ gas sensor has a heating element inside that changes its resistance in the presence of different gases.
   * The sensor converts the resistance change into a corresponding analog voltage signal, which represents the gas concentration.
3. **Signal Reading:**
   * The ESP32 continuously reads the analog signal from the sensor via the connected analog GPIO pin (e.g., D34).
   * The analog-to-digital converter (ADC) of the ESP32 converts this analog voltage into a digital value (typically ranging from 0 to 4095, depending on the resolution of the ADC).
4. **Processing:**
   * The ESP32 processes the digital value and compares it against a predefined threshold.
   * If the sensor value exceeds this threshold, it indicates a high concentration of gas.
5. **Alerting:**
   * If the gas concentration is above the threshold, the ESP32 sets the connected digital GPIO pin (e.g., D15) to HIGH.
   * This HIGH signal turns the buzzer on, producing an audible alert to warn about the high gas concentration.
   * If the gas concentration is below the threshold, the ESP32 sets the GPIO pin to LOW, ensuring the buzzer remains off.

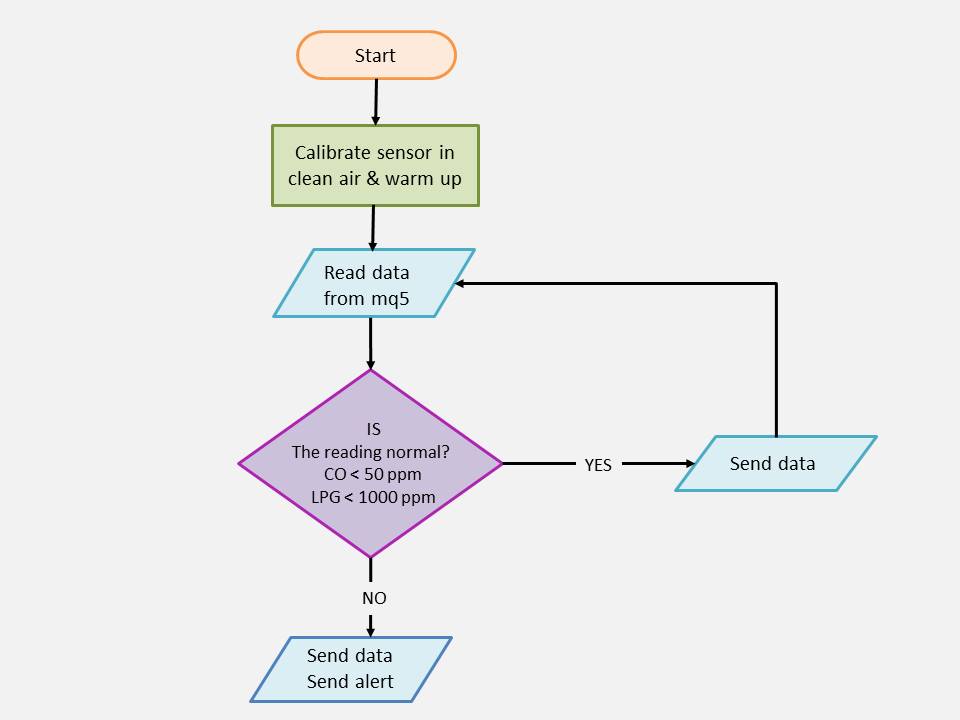


Figure 47 / CO and LPG flowchart

**6.2.3 Code and Test result**

**6.2.3.1 Code**

#define MQ5PIN  34       // Analog pin connected to MQ5 sensor

#define BUZZERPIN 2      // Digital pin connected to the buzzer

void setup() {

  Serial.begin(9600);

  pinMode(BUZZERPIN, OUTPUT);

  dht.begin();

}

* **Constants:**
  + **MQ5PIN:** is defined as the analog pin number where the MQ5 sensor is connected.
  + **BUZZERPIN:** is defined as the digital pin number where the buzzer is connected.
* **Setup Function:**
  + **pinMode(BUZZERPIN, OUTPUT):** Configures the pin connected to the buzzer as an output pin.
  + **Serial.begin(9600):** Initializes serial communication at a baud rate of 9600, which is used for sending data to the Serial Monitor for debugging purposes.

void loop() {

  // Read CO and LPG levels from MQ5 sensor

  float coValue = analogRead(MQ5PIN);

  float lpgValue = analogRead(MQ5PIN);

  failed(coValue, lpgValue, "MQ5 sensor");        // Call failed function for MQ5 sensor

  Serial.print("CO Level: ");

  Serial.print(coValue);

  Serial.print(" ppm  ");

  check(coValue, "CO", 50);                       // Call check function for CO level

  Serial.print("LPG Level: ");

  Serial.print(lpgValue);

  Serial.print(" ppm  ");

  check(lpgValue, "LPG", 1000);                   // Call check function for LPG level

  delay(1000); // Delay for 1 second between each measurement

}

**Main Loop**

* **Reading Sensor Value:**
  + **float sensorValue = analogRead(MQ5PIN)**: Reads the analog value from the MQ5 sensor pin, which gives a voltage that corresponds to the concentration of gases.
* **Check for Failed Readings:**
  + **failed(sensorValue, "MQ5 sensor"):** Calls the **failed()** function to check if the sensor reading is valid.
* **Display and Check CO Level:**
  + **Serial.print("CO Level: "); Serial.print(sensorValue); Serial.print(" ppm ");**: Prints the CO level to the Serial Monitor.
  + **check(sensorValue, "CO", 50):** Calls the **check()** function to determine if the CO level exceeds the threshold of 50 ppm.
* **Display and Check LPG Level:**
  + **Serial.print("LPG Level: "); Serial.print(sensorValue); Serial.print(" ppm ");**: Prints the LPG level to the Serial Monitor.
  + **check(sensorValue, "LPG", 1000):** Calls the **check()** function to determine if the LPG level exceeds the threshold of 1000 ppm.
* **Delay:**
  + **delay(1000):** Waits for 1 second before taking the next measurement, providing a stable reading interval.

void failed(float value1, float value2, String sensor) {

  if (isnan(value1) || isnan(value2)) {

    Serial.println("Failed to read from " + sensor);

  }

}

**Failed Function**

* **Error Checking:**
  + **if (isnan(value)):** Checks if the sensor value is not a number (NaN), which indicates a failed reading.
  + **Serial.println("Failed to read from " + sensor):** Prints an error message to the Serial Monitor if the reading failed.

void check(float value, String Name, int max) {

  if (isnan(value)) {

    Serial.println("Failed to read " + Name + " level");

    return;

  }

  if (value > max) {

    Serial.println("High " + Name + " level! Danger");

    buzzer();   // Call buzzer function

  } else {

    Serial.println(Name + " level is normal");

  }

}

**Check Function**

* **Error Checking:**
  + **if (isnan(value)):** Checks if the sensor value is NaN.
  + **Serial.println("Failed to read " + Name + " level"):** Prints an error message if the reading is NaN.
* **Threshold Checking:**
  + **if (value > max):** Compares the sensor value to the maximum threshold value.
  + **Serial.println("High " + Name + " level! Danger"):** Prints a danger message if the sensor value exceeds the threshold.
  + **buzzer():** Calls the buzzer() function to sound the alarm.
  + **else: If the value is within the normal range:**
    - **Serial.println(Name + " level is normal"):** Prints a normal level message.

void buzzer() {

  digitalWrite(BUZZERPIN, HIGH);

  delay(1000);  // Buzz for 1 second

  digitalWrite(BUZZERPIN, LOW);

}

**Buzzer Function**

* **Buzzer Activation:**
  + **digitalWrite(BUZZERPIN, HIGH):** Turns the buzzer on.
  + **delay(1000):** Keeps the buzzer on for 1 second.
  + **digitalWrite(BUZZERPIN, LOW):** Turns the buzzer off after 1 second.

**6.2.3.2 Test result**

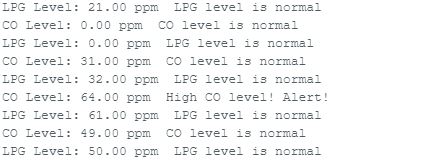


Figure 48 / Test CO and LPG

1. **LPG Level:** 21.00 ppm LPG level is normal
   * **LPG Level:** 21.00 ppm, which is below the threshold of 1000 ppm.
   * **Status:** "LPG level is normal."
2. **CO Level:** 31.00 ppm CO level is normal
   * **CO Level:** 31.00 ppm, which is below the threshold of 50 ppm.
   * **Status:** "CO level is normal."
3. **LPG Level:** 32.00 ppm LPG level is normal
   * **LPG Level:** 32.00 ppm, which is below the threshold of 1000 ppm.
   * **Status:** "LPG level is normal."
4. **CO Level:** 64.00 ppm High CO level! Alert!
   * **CO Level:** 64.00 ppm, which is above the threshold of 50 ppm.
   * **Status:** "High CO level! Alert!"
   * **Action:** The buzzer should sound an alarm because the CO level is dangerous.
5. **LPG Level:** 61.00 ppm LPG level is normal
   * **LPG Level:** 61.00 ppm, which is below the threshold of 1000 ppm.
   * Status: "LPG level is normal."
6. **CO Level:** 49.00 ppm CO level is normal
   * **CO Level:** 49.00 ppm, which is below the threshold of 50 ppm.
   * **Status:** "CO level is normal."



Chapter 7

Conclusion and future work



smart safety Helmet team

Faculty of computers and information