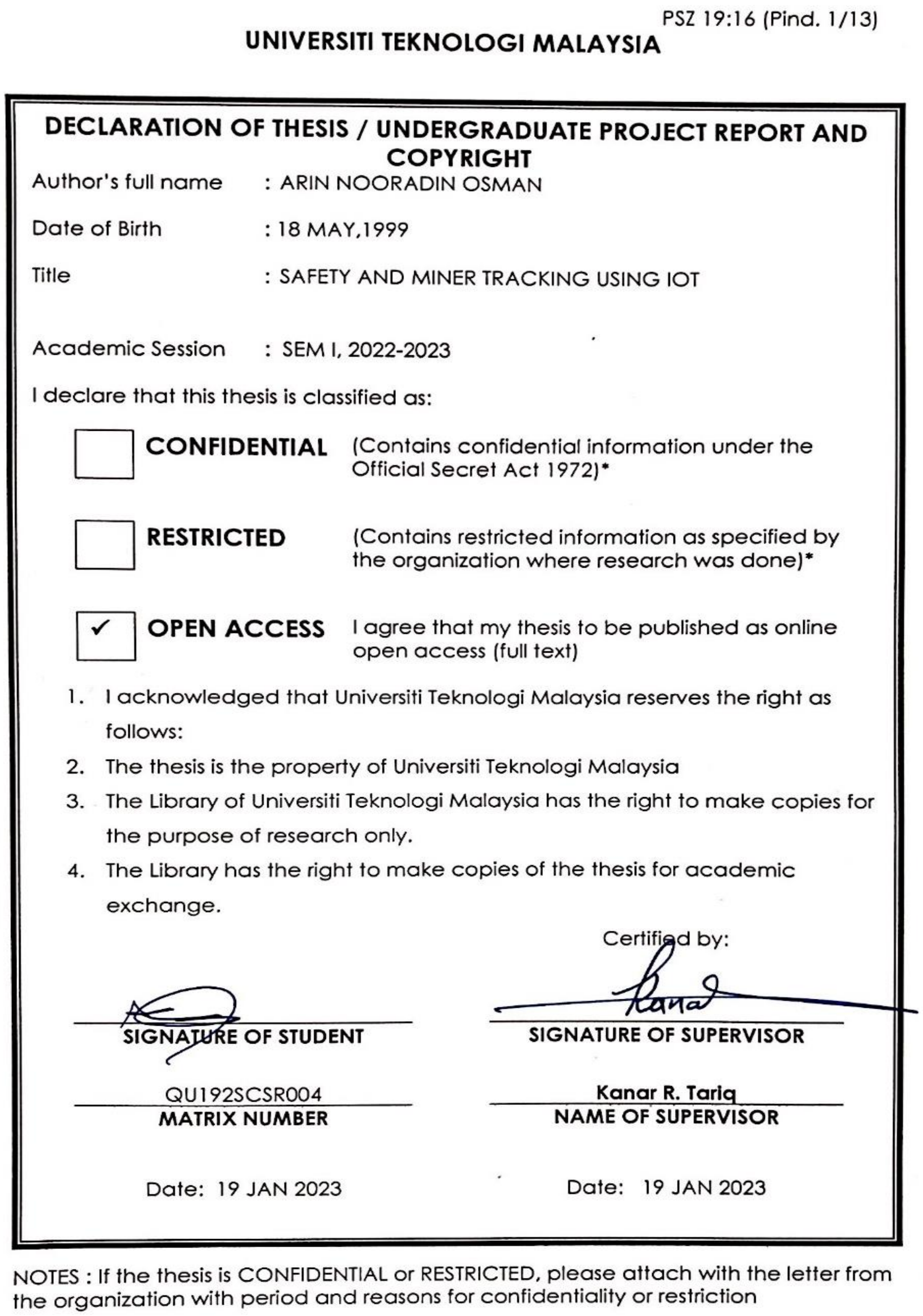
UNIVERSITI TEKNOLOGI MALAYSIA

PSZ 19:16 (Pind. 1/13)

UNIVERSITI TEKNOLOGI MALAYSIA



PSZ 19:16 (Pind. 1/13)

UNIVERSITI TEKNOLOGI MALAYSIA

DECLARATION OF THESIS / UNDERGRADUATE PROJECT REPORT AND COPYRIGHT

Author's full name	: ARIN NOORADIN OSMAN
Date of Birth	: 18 MAY,1999
Title	: SAFETY AND MINER TRACKING USING IOT
Academic Session	: SEM I, 2022-2023

I declare that this thesis is classified as:

☐	**CONFIDENTIAL**	(Contains confidential information under the Official Secret Act 1972)*
☐	**RESTRICTED**	(Contains restricted information as specified by the organization where research was done)*
✓	**OPEN ACCESS**	I agree that my thesis to be published as online open access (full text)

1. I acknowledged that Universiti Teknologi Malaysia reserves the right as follows:
2. The thesis is the property of Universiti Teknologi Malaysia
3. The Library of Universiti Teknologi Malaysia has the right to make copies for the purpose of research only.
4. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

[illegible]	[illegible]
SIGNATURE OF STUDENT	**SIGNATURE OF SUPERVISOR**
QU192SCSR004	**Kanar R. Tariq**
MATRIX NUMBER	**NAME OF SUPERVISOR**
Date: 19 JAN 2023	Date: 19 JAN 2023

NOTES : If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organization with period and reasons for confidentiality or restriction

NOTES : If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organization with period and reasons for confidentiality or restriction

19th / January/ 2023

Librarian

Perpustakaan UTM

UTM, Skudai Johor

Sir,

CLASSIFICATION OF THESIS AS OPEN ACCESS

Smart Safety Helmet

ARIN NOORADIN OSMAN

Please be informed that the above mentioned thesis entitled ″ Smart Safety Helmet” be classified as OPEN ACCESS.

Thank you.

Sincerely yours,

Mr. KANAR TARIQ, As Sulaymaniyah Iraq, +964 770 143 6938

“I hereby declare that we have read this thesis and in my
opinion this thesis is suffcient in term of scope and quality for the
award of the degree of Bachelor of Computer Science (Computer Networks &
Security)”

Signature	:	[illegible]
Name of Supervisor I	:	KANAR TARIQ
Date	:	19 January 2023

Smart Safety Helmet

ARIN NOORADIN OSMAN

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Bachelor of Computer Science (Computer Networks & Security)

School of Computing
Faculty of Engineering
Universiti Teknologi Malaysia

JANUARY 2023

DECLARATION

I declare that this thesis entitled “*Smart Safety Helmet”* is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	[illegible]
Name	ARIN NOORADIN OSMAN
Date	18 January 2023

DEDICATION

To my parents, for their unwavering love and support throughout my entire life. Their sacrifices and dedication to my well-being have been the foundation of my success. They have always been there for me, through every challenge and every triumph. Their love and guidance have been my constant compass, guiding me through the twists and turns of my academic journey. Without their unwavering belief in me, I would not be where I am today. This project is a culmination of my academic endeavors and it would not have been possible without their guidance and encouragement. I am eternally grateful for all that they have done for me and for the sacrifices they have made to ensure my success. This project is dedicated to my parents, with all my love, gratitude and appreciation for everything they have done for me.

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my computer teacher as well as our principal who gave me the golden opportunity to do this wonderful project on the topic, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them.

Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

ABSTRACT

The construction sector continues to be one of the high-risk industries. This may be attributed to changing industrialization patterns as well as the scale and complexity of development projects growing. Therefore, when compared to other high-risk industries, the development industry has the highest percentage of fatal accidents as well as disabling injuries. These mishaps have real financial and societal repercussions. Aside from loss of life and property, financial losses might be attributed to medical expenses for treating and restoring injured people. [2] Because of how high the buildings are, it is more risky for workers to be there, accidents occur frequently at work, and because of how high the structures are, it is very difficult for management to prevent mishaps. In the event of an emergency, they should call for assistance. While construction workers attempt to create subway tunnels, there are several injuries occurring. It is extremely dangerous to impact, penetrating, exhuming, and pouring concrete in settings that are frequently dull, damp, and dirty when building and maintaining tunnels. [3] Construction accidents and injuries can occur for a variety of reasons; for example, falling from a ladder, becoming sickened by too much subway dust, sliding and falling, having a heavy fall to the head, etc. Such mishaps can cause severe injuries, including crushed and shattered bones, removed appendages, traumatic brain injuries, burns, electric shock, and, in rare circumstances, death. Bosses have a responsibility to ensure secure conditions of employment for their employees. Their spokespeople can persevere when they fail to do so. If you're ready to pursue legal action, contact our Modern York cave worker attorney right once.

ABSTRAK

Le secteur de la construction continue d'être l'une des industries à haut risque. Cela peut être attribué à l'évolution des modèles d'industrialisation ainsi qu'à l'échelle et à la complexité croissantes des projets de développement. Par conséquent, par rapport à d'autres industries à haut risque, l'industrie du développement a le pourcentage le plus élevé d'accidents mortels ainsi que de blessures invalidantes. Ces mésaventures ont de réelles répercussions financières et sociétales. Outre les pertes en vies humaines et en biens, les pertes financières peuvent être attribuées aux dépenses médicales liées au traitement et à la restauration des personnes blessées. [2] En raison de la hauteur des bâtiments, il est plus risqué pour les travailleurs d'y être, les accidents se produisent fréquemment au travail et, en raison de la hauteur des structures, il est très difficile pour la direction d'éviter les accidents. En cas d'urgence, ils doivent demander de l'aide. Alors que les travailleurs de la construction tentent de créer des tunnels de métro, plusieurs blessures se produisent. Il est extrêmement dangereux d'impacter, de pénétrer, d'exhumer et de couler du béton dans des environnements souvent ternes, humides et sales lors de la construction et de l'entretien des tunnels. [3] Les accidents de construction et les blessures peuvent survenir pour diverses raisons. par exemple, tomber d'une échelle, devenir malade à cause d'une trop grande quantité de poussière dans le métro, glisser et tomber, tomber lourdement sur la tête, etc. brûlures, électrocution et, dans de rares circonstances, la mort. Les patrons ont la responsabilité d'assurer des conditions d'emploi sûres pour leurs employés. Leurs porte-parole peuvent persévérer quand ils ne le font pas. Si vous êtes prêt à intenter une action en justice, contactez immédiatement notre avocat spécialisé dans les caves de Modern York.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	-	Artificial Neural Network
GA	-	Genetic Algorithm
PSO	-	Particle Swarm Optimization
MTS	-	Mahalanobis Taguchi System
MD	-	Mahalanobis Distance
TM	-	Taguchi Method
UTM	-	Universiti Teknologi Malaysia
XML	-	Extensible Markup Language
ANN	-	Artificial Neural Network
GA	-	Genetic Algorithm
PSO	-	Particle Swarm Optimization

LIST OF SYMBOLS

δ	-	Minimal error
D, d	-	Diameter
F	-	Force
v	-	Velocity
p	-	Pressure
I	-	Moment of Inersia
r	-	Radius
Re	-	Reynold Number

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The construction industry proceeds to rank as one of the most elevated chance businesses. This could be credited to rising patterns in industrialization and the increasing size and complexity of development ventures. Consequently, the development industry has the most elevated rate of lethal mishaps as well as debilitating injuries compared to other high-risk businesses. These mischances have genuine financial and social consequences. Other than the misfortune of life and property, monetary misfortunes may be credited to restorative bills for the treatment and restoration of harmed individual.[2] Workers work in high building which makes it more dangerous for them to work in them and accidents happens all the time in works places and with the workers working in high buildings its making it very hard for them to call out for help in case of an emergency. There many injuries occurring in subway tunnels while workers trying to build subway tunnels, Building and keeping up tunnels is exceptionally unsafe work including impacting, penetrating, exhuming, and pouring concrete in situations that are regularly dull, soggy, and messy.[3] Development mishaps and wounds can effectively emerge due to an assortment of components, the injuries could have are falling from a ladder, to many dust in the subway that causes sickness, slipping and falling taking a bad fall in the head, Such mischances can result in serious wounds, counting pulverized and broken bones, excised appendages, traumatic brain wounds, burns, electric shock, and, in extraordinary cases, passing. It is the duty of bosses to guarantee secure working conditions for their workers. When they come up short to do so, their representatives can endure. Contact our Modern York burrow laborer legal counselor nowadays in case you're prepared to require lawful activity.[5]

1.2 Problem Background

Mining is a multidimensional business that involves complex activities in tunnels, underground, and so on. This includes a number of risk factors that have an impact on miners' health. The Chasnala mining tragedy in the Indian state of Jharkhand, near Dhanbad, nearly killed 372 miners. This was regarded as one of the greatest mining catastrophes in history. Miners may be unaware of exterior factors like as temperature, pressure, and so on. Miners are sometimes killed when they collide with large things such as mines and hard rock. [4]Another problem is that the miners are inhaling dangerous gases, putting their lives in peril. Miners are unable to contact with the outside world under this condition. That's why I proposed a system to make the life workers in easier so that they can do their job with a safe mind and we reach them soon enough to save their lives either in construction sites or subway tunnels. [2] each worker helmet circuit is integrated with a panic/emergency button. This button when pressed shows an emergency sign over the IOT application about the worker emergency. This can be used for any emergencies.

1.3 Project Aim

The main goal of this project is to provide a clear and point-to-point view of the underground mining system, as well as dependable communication via IoT. As a result, it benefits the miners present within the mine and assures their safety. Assuring miner safety in the event of a mining accident caused by an increase in temperature, pressure, or force. [6And to assist coal miners within mines in communicating with the outside world. And to monitor the conditions inside the mines and notify miners in the event of an emergency

1.4 Project Objectives

The objectives of the project are:

(a) To monitor the Position and the status of the worker through the application.

(b) To create an emergency button installed on the smart helmets comes to aid when there are some emergency cases.

(c) To reduce the risks of massive injuries.

1.5 Project Scope

The scopes of the project are:

(d) The target area is crowded public areas and legal road sides.

(e) The system has three type of user which is Construction companies, Construction fields and Subway tunnels.

(f) The software's that will use for this project is (Visual studio code,C++,Python …etc.).

(g) The hardware's that will use in this project is (Arduino Wi-Fi module,buzzeer,RF,TX,RX,LCDdisplay,crystal,oscillator,resistors,capacitors,transistors,cables,and,connecters,diodes,PCB,and,breadboard,LED,transformer,panic button,switch,IC,IC socket....etc.).

1.6 Project Importance

A smart helmet will be developed to assist workers in avoiding different hazardous risk factors while working in the mining sectors. While creating this project, the integrated features of the hardware components were preserved in concordance and sequence.

Mines of today use IoT for particular activities such as detection, placing equipment, identifying workers, and real-time tailing dam monitoring at various phases of mining. This initiative is critical now because IoT can make mines safer for employees, more cost-effective, and efficient to run. For the time being, IoT is being utilized in the mining sector to make energy consumption more efficient, create a smart environment on mining sites, optimize risk management and mitigation measures, and so on. Finally, the system is dependable, with basic and easily accessible components that make this project relatively light weight, portable, and usable.

1.7 Report Organization

In this chapter, the problem has been covered, objectives and what was our scopes, and the goal of the project with all the components of the project that will be used.

Chapter 2 Literature Review: which is literature view will be discussed about developing system and getting other relates systems also articles and research-based or related to the topic.

Chapter 3 Methodology: The aim of the research, the research technique will desire to employ, the equipment to be utilized, and will collect data, the sorts of data collection, and how you obtained it.

Chapter 4 Proposed System Design: This is the software development stage, which is based on user requirements and a comprehensive examination of a system based on the system analysis.

Chapter 5 Conclusion: This is the dissertation's final and most distinctive chapter. Previously, most chapters could be simply created by following a template with specific standards for each part.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Generally, a literature review is a systematic analysis of current existing research related to the proposed system. In other words, it shows the process of collecting, obtaining, and analyzing data required for the system's development. It also provides explanations and justifications that support in the resolution of research challenges. In this chapter, serval topic will be discussed. These include background about the physical activity exercises and diet system, existing system analysis and the comparison with existing system, finally this chapter end with chapter summery.

2.1.1 Background

The tracking and safely helmet application system could potentially be used to improve safety and efficiency in a variety of settings, such as construction sites, factories, warehouses, or any other environment where workers are required to wear helmets for safety purposes. The system will use GPS or other location tracking technology to monitor the location of each worker wearing a helmet. This could be used to ensure that workers are in the designated work area and to alert supervisors if a worker strays outside of the designated area. The system will use sensors to detect potential safety hazards, such as falling objects or other hazards, and send alerts to the worker and/or their supervisor in real-time. The system will include a built-in communication system, such as a two-way radio or a mobile phone, to allow employees to stay in touch with each other and with their supervisors while on the job. The system will collect and store data on the usage and performance of the helmets, as well as any safety incidents that occur. This data could then be analyzed to identify trends and areas for improvement in terms of safety and efficiency.

2.2 Existing system analysis

The existing system analysis will be discussed in this part. There are a number of tracking and safely helmet related mobile applications available, this chapter will include some case study for the existing system.

2.2.1 Case Study 1

In this case study, researchers implemented a GPS-enabled tracking and safety helmet system on a large construction site in order to assess the impact on safety and efficiency. The system included sensors to detect potential hazards, such as falling objects, and send alerts to workers and supervisors in real-time. The study found that the use of the system resulted in a significant reduction in safety incidents on the construction site, as well as an increase in overall efficiency. The researchers attributed these improvements to the ability of the system to provide real-time alerts and allow for quick communication between workers and supervisors. [1]

2.2.2 Case Study 2

In this case study, researchers evaluated the effectiveness of a two-way radio-based communication system for safety helmet users in a factory setting. The system allowed workers to stay in touch with each other and with their supervisors while on the job, and also included a panic button feature that could be used to alert supervisors in the event of an emergency. The study found that the use of the system resulted in a significant reduction in safety incidents on the factory floor, as well as improved communication and coordination among workers. [2]

2.2.3 Case study 3

In this case study, researchers analyzed the use of wearable technology, including tracking and safety helmet systems, in a large warehouse setting. The study found that the use of these systems resulted in a significant reduction in safety incidents and an increase in efficiency, as workers were able to receive real-time alerts about potential hazards and communicate with each other more effectively. The study also identified some challenges with the implementation of the systems, including the need for ongoing maintenance and the need to ensure that workers were properly trained on how to use the technology.

2.3 Current System Analysis

In order to more accurately analyze the current status of tracking and helmet application systems, it would be helpful to gather information on the following points:

Current Use: How widespread is the use of tracking and safety helmet systems in various industries? Are they used in a wide range of settings, or are they used more in certain industries or locations?

Impact: What is the track record of these systems in terms of improving safety and efficiency? Has any research or evaluation been done to assess the effectiveness of these systems?

Cost: What is the cost of implementing and maintaining a tracking and safety helmet system? Is the investment worth it, given the potential benefits in terms of improved safety and efficiency?

User satisfaction: What is the feedback from workers and supervisors who use these systems? Do they find the systems easy to use and helpful in their work, or are there problems with usability or functionality?

Technology Limitations: Are there any technological limitations or challenges with the current generation of tracking and safety helmet systems? Are there opportunities for improvement or innovation in this area?

By collecting and analyzing data on these points, a clear understanding of the current status of tracking and safety helmet systems can be gained and areas for improvement or further research can be identified.

2.4 Comparison between existing systems

Setting: The case study on wearable technology in warehouses focused on the use of these systems in a large warehouse setting, while the other two case studies focused on the use of tracking and safety helmet systems in construction and factory settings, respectively.

Technologies and features: The warehouse case study focused on the use of wearable technology in general, while the other two case studies specifically mentioned the use of GPS-enabled helmet systems and two-way radio-based communication systems.

Effectiveness: All three case studies found that the use of tracking and safety helmet systems resulted in a significant reduction in safety incidents and an increase in efficiency. However, the specific impacts on safety and efficiency may have differed depending on the specific technologies and features used, as well as the specific context and setting in which the systems were implemented.

Challenges and limitations: All three case studies identified some challenges and limitations with the implementation of tracking and safety helmet systems. For example, the warehouse case study mentioned the need for ongoing maintenance and worker training, while the other two case studies mentioned the need for careful planning and coordination in order to ensure the effectiveness of the systems.

Overall, it is important to consider the specific context and setting in which tracking and safety helmet systems are being used, as well as the specific technologies and features that are implemented, when evaluating their effectiveness in improving safety and efficiency.

2.5 Literature Review of Technology Used

There are several tools and technologies that could be used to implement a tracking and safety helmet application system. Some possible options include:

GPS: Global positioning system (GPS) technology could be used to track the location of each worker wearing a helmet. This could be done using GPS-enabled devices such as smartphones or dedicated GPS tracking devices.

Sensors: Sensors could be used to detect potential safety hazards, such as falling objects or other hazards, and send alerts to the worker and/or their supervisor in real-time.

Communication: A two-way radio or a mobile phone could be used to allow workers to stay in touch with each other and with their supervisors while on the job.

Data storage and analysis: The system could use a database or other data storage system to collect and store data on the usage and performance of the helmet, as well as any safety incidents that occur. This data could then be analyzed using software tools to identify trends and areas for improvement in terms of safety and efficiency.

User interface: The system could include a user interface, such as a website or mobile app, to allow employees and supervisors to access and interact with the system. This could include features such as the ability to view location data, receive alerts, and report safety incidents

2.6 Survey

Brief introduction: Smart helmets are hard hats made specifically for those who have jobs on construction sites. Construction workers and engineers use the smart helmet for both safety reasons and site planning purposes. Smart helmets have the same basic appearance as a standard hard hat, but they have built-in technology that includes tracking tools, sensors and augmented reality. The technology can warn workers of dangerous conditions and alert the facility to get help and to keep track of the workers where they are at all times so they are easy to find when there is an emergency.

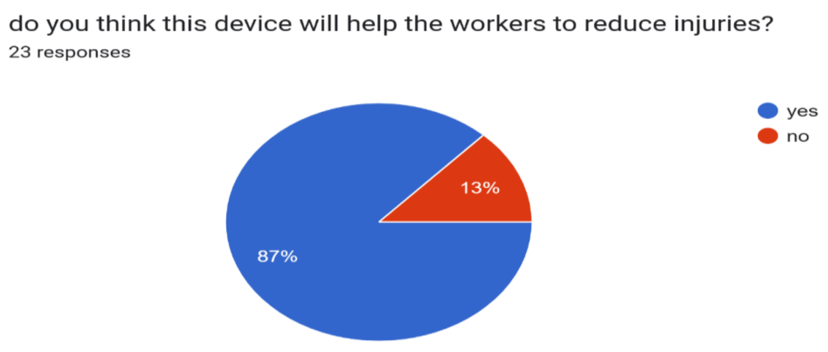


Figure 2.6.1

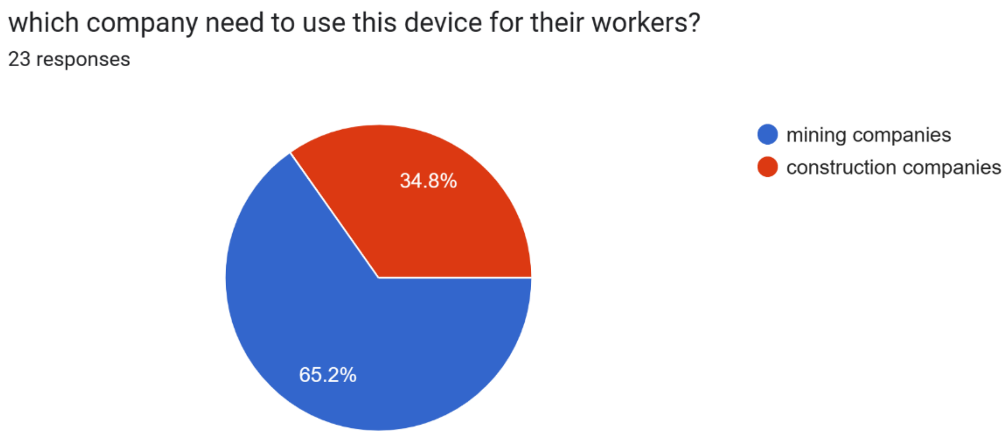


Figure 2.6.2

this smart helmet will keep track of the workers and provide safety at all times, do you think this is useful?

23 responses

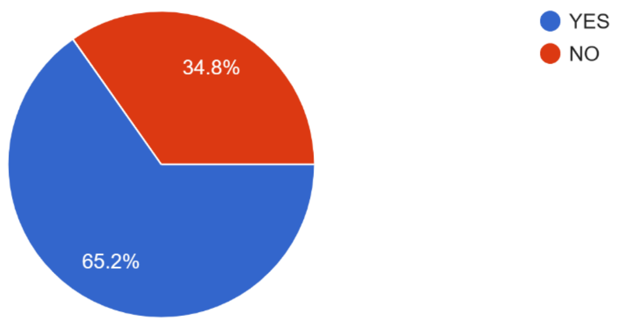


Figure 2.6.3

this smart helmet will keep track of all workers , will this help workers more efficient in their jobs?

23 responses

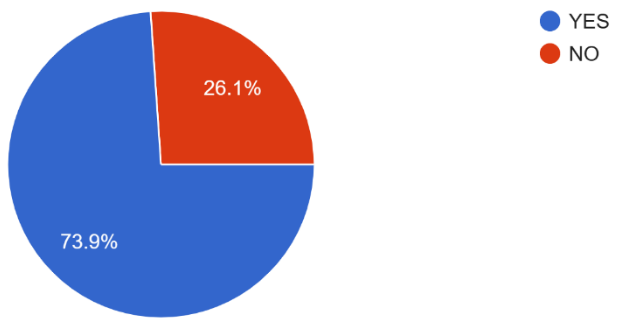


Figure 2.6.4

in case of emergencies this smart hat have a built in panic button and the worker can press it to get help and instant care, will this reduce massive risks?

23 responses

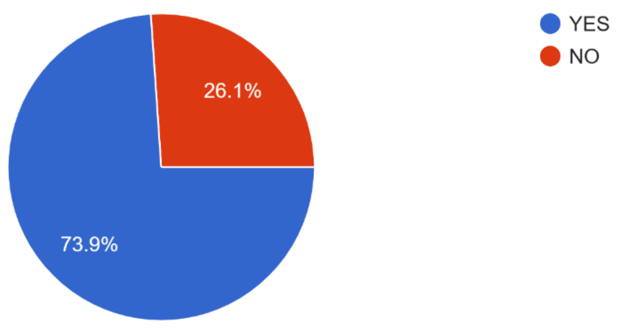


Figure 2.6.5

when the worker is working and he takes off the hat we are alerted so we can take action and to tell the worker to put on the helmet

23 responses

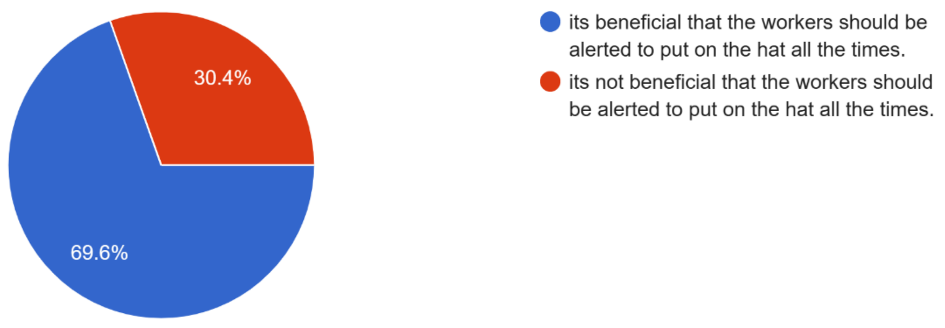


Figure 2.6.6

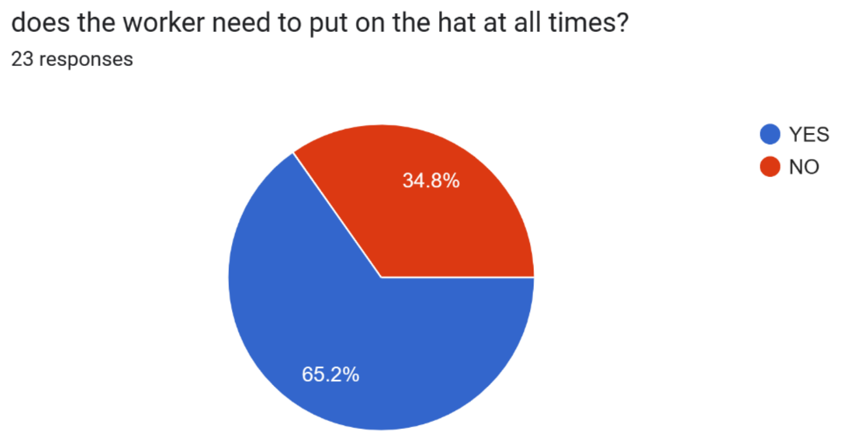


Figure 2.6.7

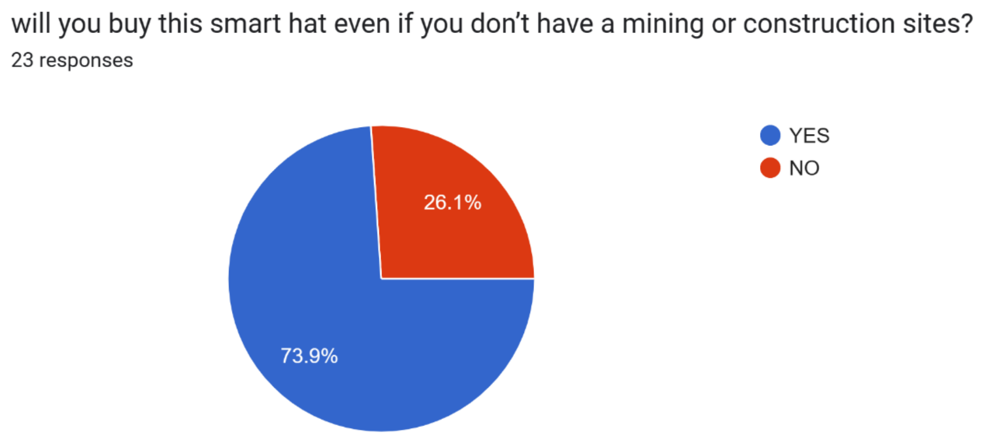


Figure 2.6.8

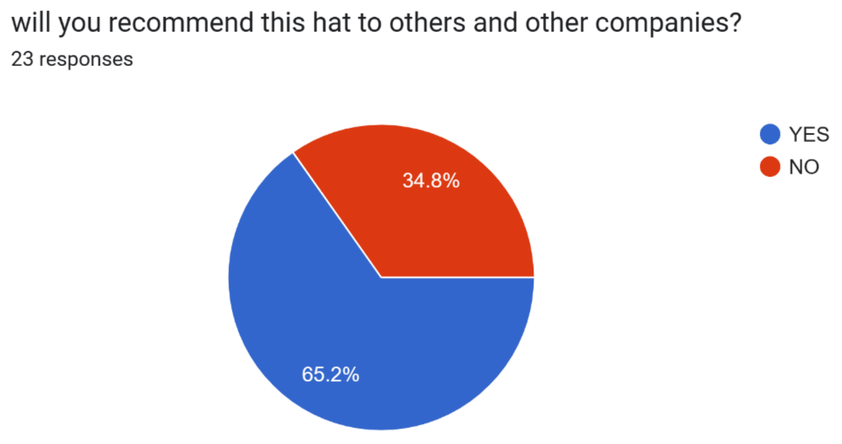


Figure 2.6.9

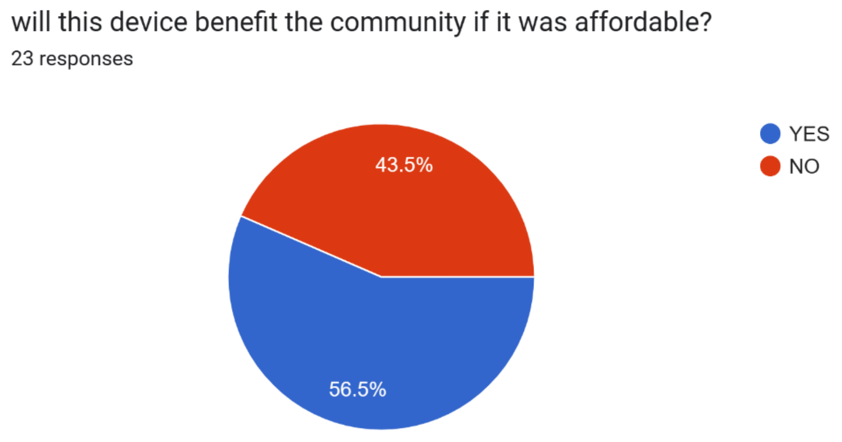


Figure 2.6.10

2.7 Chapter Summary

In this chapter, the inter organization case study was stated. The comparison between existing systems also clearly discussed. Next, the result and analysis of the survey are also well identified. Lastly, this chapter ends with the technology and tools used to develop the system. The focus of the next chapter will be on the approach employed during the development process.

CHAPTER 3

SYSTEM DEVELOPMENT METHODOLOGY

3.1 Introduction

In this chapter methodology will be discussing what methodology is and which methodology will be used, a methodology is a way of implementing the project to show how the project is structured and processed each project or company use different methodology it can be divided into two part which is traditional and modern methodology each uses a different approach to the process, the purpose of methodology in software development is to have better estimates, deliver stable systems, keep the customer informed also clear understanding and view of the tasks that the team or person will work on also each methodology have it is own rules to follow and initialize some popular example of the methodology are Agile, DevOps, waterfall, Rapid application...etc.

3.2 Methodology Choice and Justification

Agile methodology is known as modern software development methodologies is a way or method of project management it divides tasks into short phases each phase can take up to 1 – 4 weeks based on the skills and knowledge of the team that is working on or individual, agile continuously collect feedback from the users based on the feedback it makes changes to the system agile is a set of frameworks and practices to raise the collaboration quality and speed, agile work on people doing the task and how they can do it together, an example of some companies that use agile for their project management PlayStation Network m CISCO, OpenLink...etc (Null).

This methodology is suitable for the OMS system there are 12 principles from the agile manifesto (Principles behind the Agile Manifesto, 2001) to understand why to choose agile methodology One of the goals is to satisfy customers for continuous andearly delivery of the software this get from direct feedback from the customer and response to those changes without the year of plans, agile always getting stakeholders requirement at any time even if it is in late of software development, Always deliver changes and tasks continuously in a short amount of time because it uses short tasks which called sprint also another important point is a collaboration between developers and business people daily, it will provide a suitable and comfortable environment for the team to work on also the best architectures, requirements, and designs emerge from self-organizing teams, some of the advantages of the agile is:

1. Higher product quality.
2. Reduced risk.
3. Better visibility into project performance.
4. Increased project control.
5. Better project predictability.

3.3 Phases of the Chosen Methodology

3.3.1 P1: define requirement

The first phase in agile that comes is requirement gathering in some organization before requirement there are a couple of steps one of them is a training of agile which provide some agile training for the client but in this system does not go to implement that, the requirement is provided goals and guides for the projects developers and staff lake of requirement can make delay over coast and decrease of user satisfaction, as mentioned in agile requirement can be changed during the development process it agile methodology will accept that according to agile manifesto says that responding to change instated of the following plan, each approach has a different method to gather requirement and the requirements is not only from stakeholders can be from people in a certain area or specific users like lawyers gamers and more.

one of the processes in the requirement are supply user stories development team canhave more detail on the user such as use cases, collaborate with stakeholders daily to get most of the requirements is a good technique to stakeholders are the main sourcesto get the requirement, requirement gathering have different approach one which are Interviews, Questionnaires or Surveys. User Observation, Document Analysis, Brainstorming…etc.

Questionnaires or surveys and brainstorming is a great way to implement in the OMS system getting user/citizen feedbacks surveys and brainstorming with stakeholders toget all necessary requirements implemented as code.

3.3.2 P2: Design

Agile design is a highly collaborative way of design when big tasks break into groups of subtasks to be shown as virtual with team design is the same as the developers tasks it will still separate into subtasks when the project goes into design development and integration process all the task to each phase can be in progress as once and each time any new design implemented or changed developers directly work on those changes or new designs in each iteration design will be more improved iteration means is a basic building block of which I have a fixed timebox means a deadline for each block, getting changes and improving design continuously will help to get quick feedback from clients and users you can adapt what are the needs of the users/clients also change management and faster development delivering subtasks of the design to the developers to work and finish the implementation in a short amount of time.

3.3.3 P3: Development and Testing

In a traditional way after the development test has been started in the agile test can be in the first approach during the development Agile testing and coding are done incrementally and interactively, continuously providing values to the system until it goes into the production process of this phase is to Continuous doing like Continuous Build, Continuous Integration (CI), Continuous Delivery (CD), and Continuous Deployment, mostly DevOps can be implemented in here which automated all these steps which called pipelines that build test and deploy in just a couple of mins, one of the tools it used for Continuous Integration is GitHub which keep all the code under one repository before the code be updated in the repository it will process into the building and testing the new code, any failure in the test process it cannot merge to production it needs works until it passed all the tests.

In each iteration all the team works under one specific task until one task is finished then can move to the next one, testing has three different types which is a Unit test which is testing inside the logic of the system how the code is implemented any logicalerror another one is Integration Tests this test is more likely the system testing those technologies or services that used to implement this code such as database, API and servers, the last one is System/E2E testing the complete system by simulating a customer.

3.3.4 P4: Implementation and Deployment

After testing has been completed deployment will come in deployment might need a period of time to be ready and fully deployed in agile deployment calls Continuous Deployment which is the purpose that makes agile deployment much more popular is the process that takes validated Features into staging and then merge them into production, typically after testing QA will jump in to do a test of the system before going to production environment there are some good practices to have a great result of the deployment, using checklist deployment each deployment to set some certain tasks before and after the deployment, providing best deployment tools such as Jenkins, GitLab CI, GitHub Actions these are tools to use for deployment based on the project these tools will speed up your deployment also automate them for each time that you want to deploy any changes.

Normally any changes to the system it automatically been built tested and deployed to the staging environment and this because of the CI/CD pipeline which automate all thesteps and each step will be complete before going to the next step, sometimes even tested fully completed passed and do not find any issue from our local machine but when implementing it in the server, in this case, is better to have a rollback strategy tobe ready for any kind of worst-case that system will face it.

The four activities in the deployment are: Deploy to production, Verify the solution, monitor for problems, Respond and recover.

3.3.5 P5: Review and Feedback

Finally, when the system is released and it is used by stakeholders with users, users often get bugs errors in the system or suggestions so this feedback, feedback is not only from stakeholders any users that use the system can have review and feedback on it, in agile it calls feedback loop it serves to increase productivity for user performance feedback will be useful to identify and catch those components that need improvement in the system, apply those feedbacks on the system there are three stages of the feedback loop first gather feedback from users and client, second analyzing those feedbacks that has been collected from stage one finally making a decision based on the analyze that has been done in stage two and this loop going on again and again that why it called feedback loops.

There are ways to collect feedback from end-users, a client cannot understand the structure of the code so there is an approach called low code app development and modern driven development for defining an application interface main point on feedback loop is a collaboration between developers and users or stakeholder to gather everything efficiently.

3.3.6 PSM: GANTT Chart

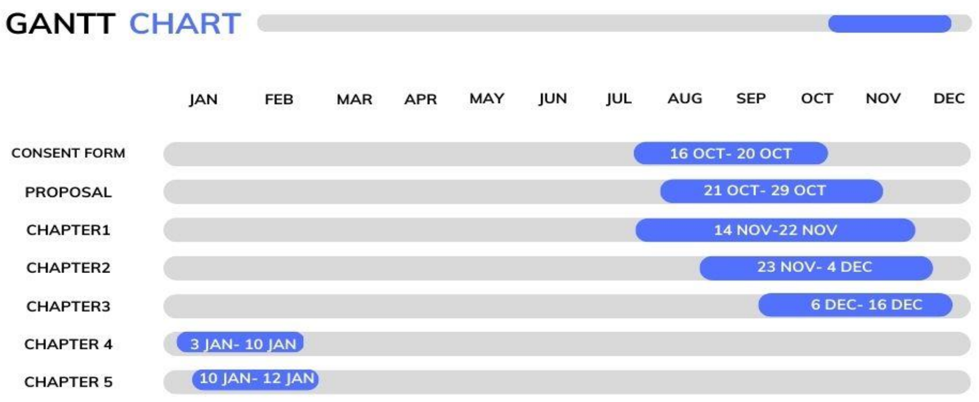


Figure 3.3.6.1 GANTT Chart

3.3.7 Design Model

Using case diagram model will help to capture the requirement of the system easily it Is a high-level function and scope of a system also shows the relation between the functionality and the users it describes how the users use the system functionality, the use case can be described in general it is not necessary to show how each function works, you can specify a relation between the functions too if a function is included or extended, for this system to design model will use Enterprise Architect software it is a comprehensive UML analysis and design tool for UML.

3.4 Technology Used Description

3.4.1 postgreSQL

PostgreSQL is an open-source powerful relation databaseusing SQL (Structured Query Language) structure which means it stores information in data tables also support JSON non-relational query it highly stables database nowadays Postgres became the default and primary database for many websites it supports performance optimization, Postgres allows you store large data safely mostly it Is been used is complex application tables canbe joint and use data from others table using join clause, the transaction is another feature of Postgres in any update either success or fails data will be preserved when talking about performance Postgres is the top one to discuss because it can handle the most compelling request and always optimize it is performed when releasing new versions some of the other important features are integration with other sources Postgres can extract data from other databasemanagement systems also can be used with most popular used programming language. (Borozenets, 2021).

3.4.2 Flutter

Flutter is an open-source mobile UI framework created by Google it is used to create a native mobile or web application, back in a day when a mobile application come to implementation developers need to create the same application using different programming for each operating system like IOS and Android flutter fix this issue you only have one code base that you can use for both IOS and Android system, flutter use Dart as programminglanguage dart has been improved these couple of years, flutter is easy to learn that makes more popular among the developers because they don't spend a lotof time to learn a new language to create an application flutter have a great compilation that maximum productivity, the old programming language it is very difficult to find good documentation to learn the program but with flutter,it makes the job easier from flutter documentation (Thomas, 2019).

3.4.3 Node Js

node js Is an open-source cross-platform use JavaScript language to run an application in the back end outside of client view, the nodehas a very high performance that makes run the node js fast node js use NPM node package manager for packaging the application with using more than 50,000 bundles, along with running fast node reduce the processing time for video and audio so there is no buffering for the data's node js is like nginx thatuse single-threaded for multiple requests so it can handle a large request, nodejs can responds in a non-blocking way that makes is it highly scalable.(Sufiyan, 2021).

3.4.4 AWS

AWS is an amazon web service is a cloud computing solution service that provides services like servers, database, software, and many morecloud will help to access services anywhere that the system is working in any devices, AWS provide a lot of services and automation for both developers andoperation, popular companies that use AWS for their services Netflix, Adobe,Coinbase…etc. In this system, AWS will be used for some services like Database which is called RDS, and running the application in the server for user testing instead install them in each device that I get from users. (Simplilearn, 2022)

3.5 System Requirement Analysis

3.5.1 Hardware requirement

Minimum hardware requirement for Safety and Miner Tracking Helmet Using IoT

1. Arduino
2. Buzzeer
3. LCD display
4. Crystal oscillator
5. Resistors
6. Capacitors
7. Transistors
8. Cables and Connecters
9. Diodes
10. PCB and Breadboard
11. LED
12. Transformer

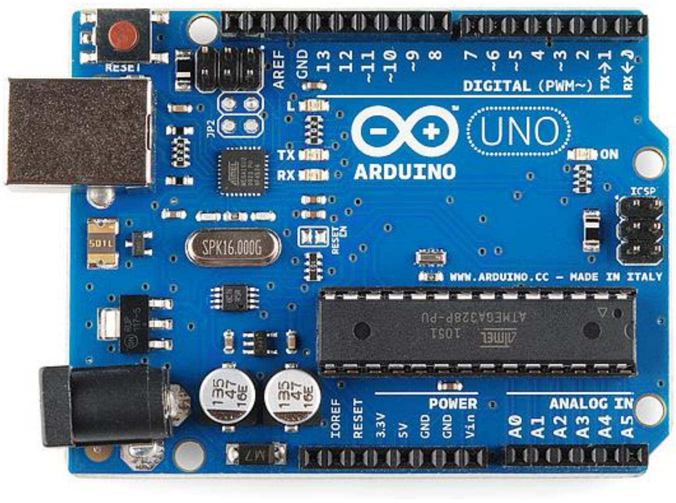


Figure 3.5.1

1. An Arduino is an open hardware development board that makers, hobbyists, and inventors can use to create and construct objects that interact with the physical environment.

Figure 3.5.2

2. The Arduino is most frequently used with buzzers. Due to its light weight, straightforward design, relatively low price, ability to produce a variety of musical tones at various frequencies, and lack of a separate oscillating circuit.

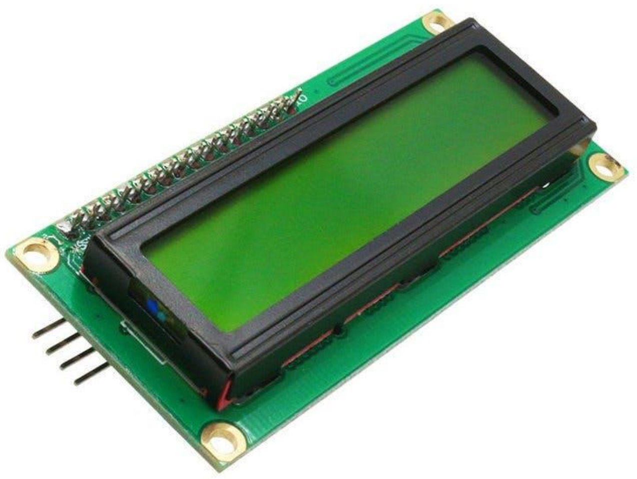


Figure 3.5.3

3. A user interface can be readily created by connecting an Arduino to a liquid crystal display (LCD). Data is frequently displayed on liquid crystal displays (LCDs) in gadgets like calculators.

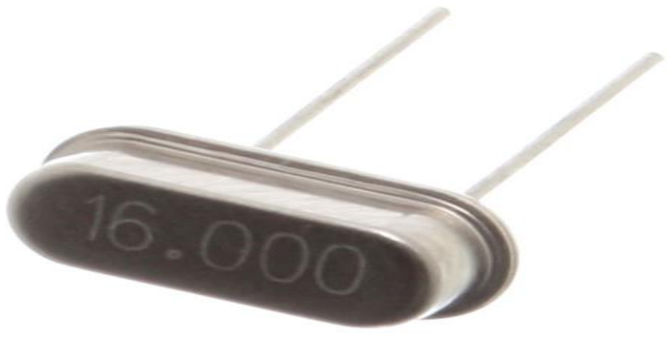


Figure 3.5.4

4. The microcontroller uses a crystal oscillator to determine time and synchronize internal processes. When transmitting and receiving signals to and from Arduino and its peripherals, time factors are extremely important. Microcontrollers have the ability to make quick judgments based on crystal oscillator frequency.

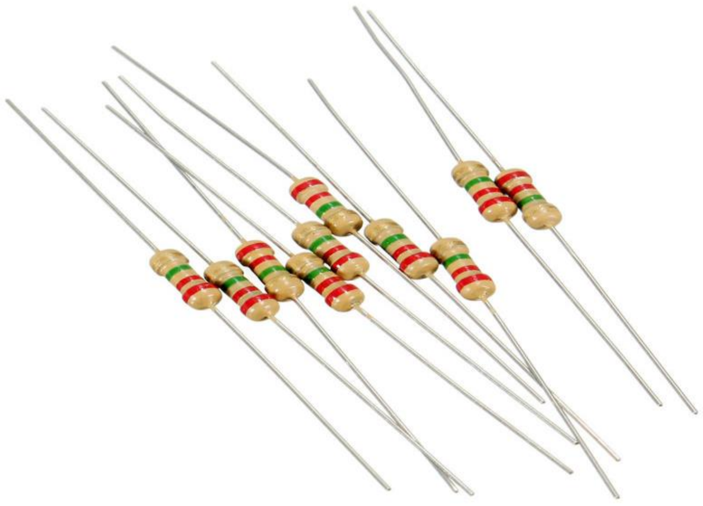


Figure 3.5.5

5. These are referred to as kilo- and mega-ohms. In this lesson, we'll employ resistors with four distinct values: 270 ohms, 470 ohms, 2.2 kohms, and 10 k ohms. Except for the various colored stripes that they have, these resistors all have the same appearance.

Figure 3.5.6

6. Electrical noise from power supply is filtered using capacitors. The capacitor helps to smooth out power dips caused by the servo's ability to draw extra current while it moves.

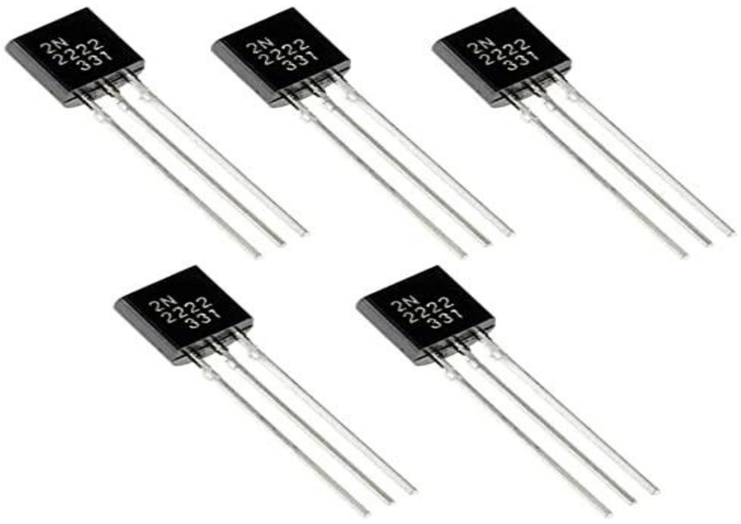


Figure 3.5.7

7. The Arduino can regulate loads with larger electrical demand thanks to the ability of a transistor to function as a digital switch.

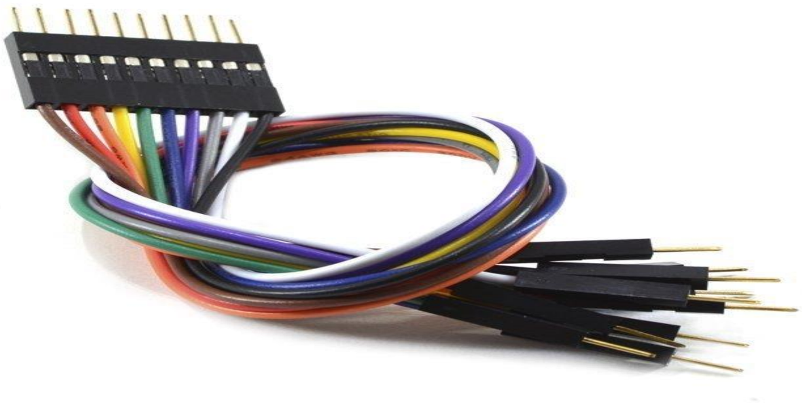


Figure 3.5.8

8. these cables used for connecting between the bread board and the Arduino to transfer connections.

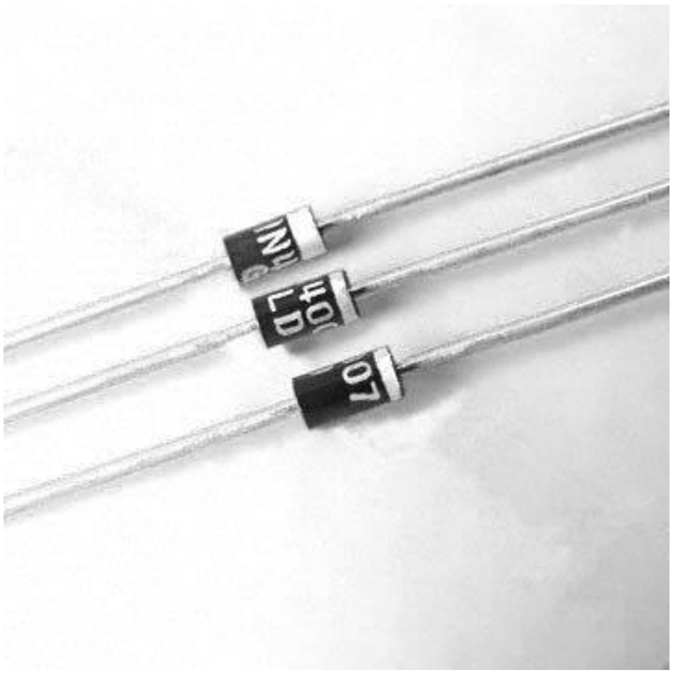


Figure 3.5.9

9. Diodes are passive electronic components with two terminals that only permit one direction of current flow.

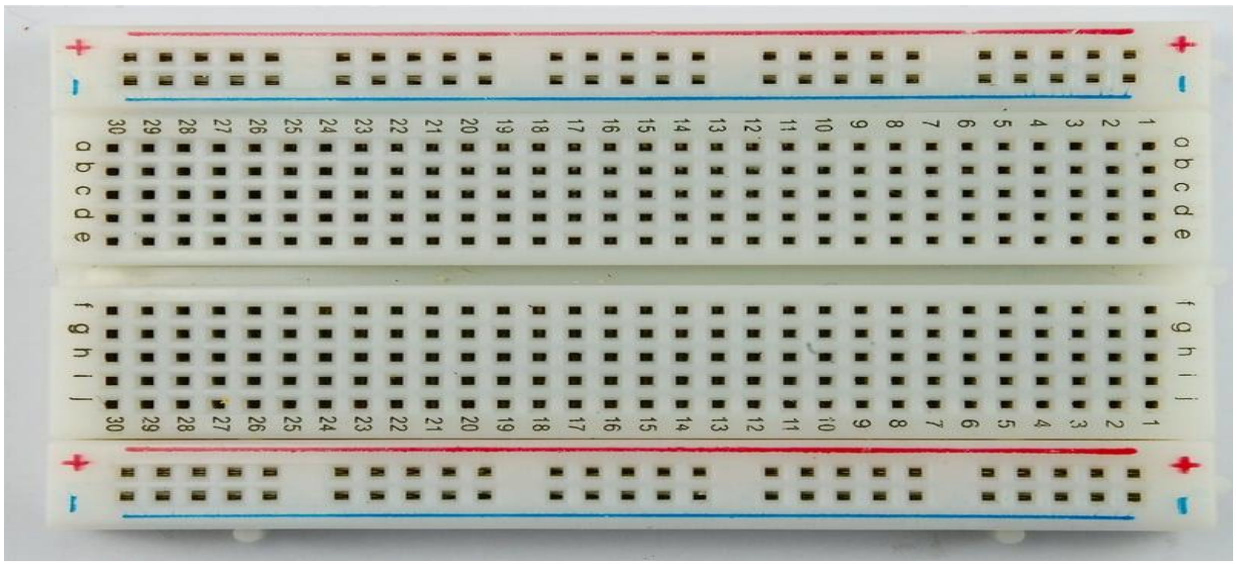


Figure 3.5.10

10. Electronics can be built using a breadboard instead of a soldering iron.

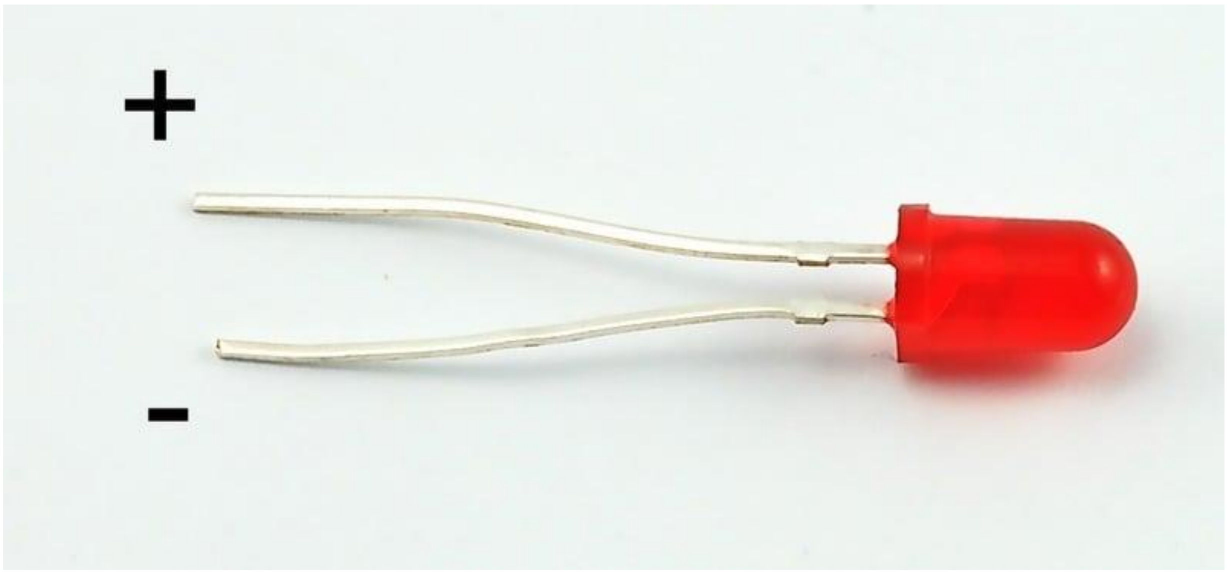


Figure 3.5.11

11. utilized in a variety of electronic gadgets as an ON/OFF indicator.

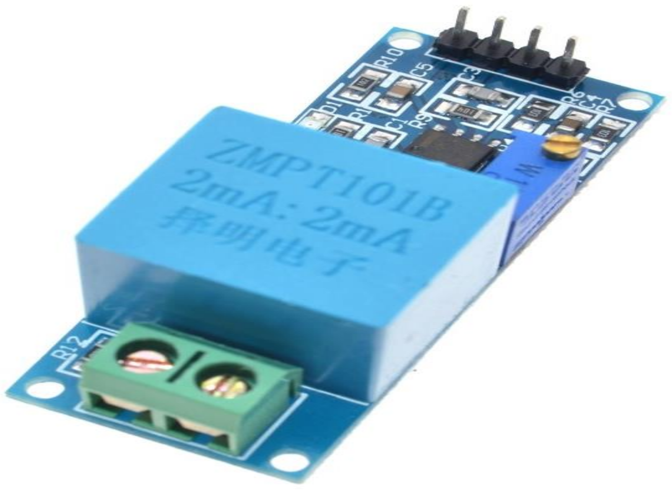


Figure 3.5.12

12. A transformer is an electrical energy transfer device that steps up or steps down the voltage from one alternating-current circuit to one or more other circuits.

3.5.2 System requirement

Software requirement is very crucial in developing a proposed system. Choosing the best software specification will lead to a smooth process of development. The minimum for software requirements is listed below.

13. Visual studio

14. React native

15. AWS

16. Flutter

3.6 Chapter Summary

Methodologies are frameworks that structures plan and control the process development of information each project can have different methodologies to implement for Tracking and safety Helmet application system agile is great and suitable agile is an approach for project management, in agile, there is a lot of frameworks to work on or you can don't use those frameworks usuallythose framework use for large or medium teams but still it can be used as an individual Choosing update to date technology is best practice to have new technology and mostof the new version technology are faster and have been more improved also those willbe the one that is implemented in the future for every project.

CHAPTER 4

REQUIREMENT ANALYSIS AND DESIGN

4.1 Introduction

In this chapter requirement and design will be discussed, translating all the theoretical ideas that have been shown in other chapters into diagrams that help to understand the more easily if the system, also shows who will use the system and why for which users this system has been designed also understand what user wants in the system those missing parts that need to be filled and to develop the right system for the right users, also it helps to test the system theoretically, in this chapter will view how the system reacts with users and how the process of the system is doing, also how many functionalities we have in the system and a bird's eye view of the whole system, designing the system will be based on all the requirement that has been gathered also from the survey that given to citizens.

4.2 Requirement Analysis

Use case diagram:

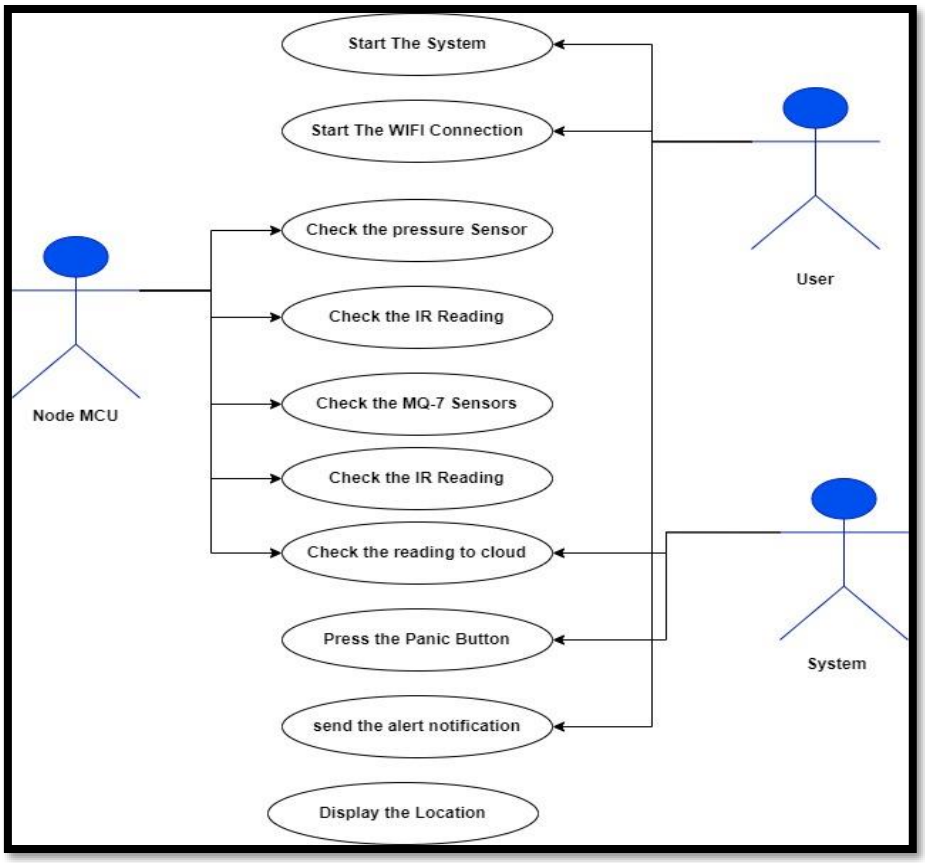


figure 4.2.1 use case diagram

The use case shows that each of the user functionality in the system 3 of the user has a shared functionality which Is login only all the users use the same method to login but each of the users has a different dashboard with different functionality they can update the information or manually set location, also on the dashboard they can see the status of there and press the panic button application that they send with those who have been rejected or accepted.

Sequence diagram:

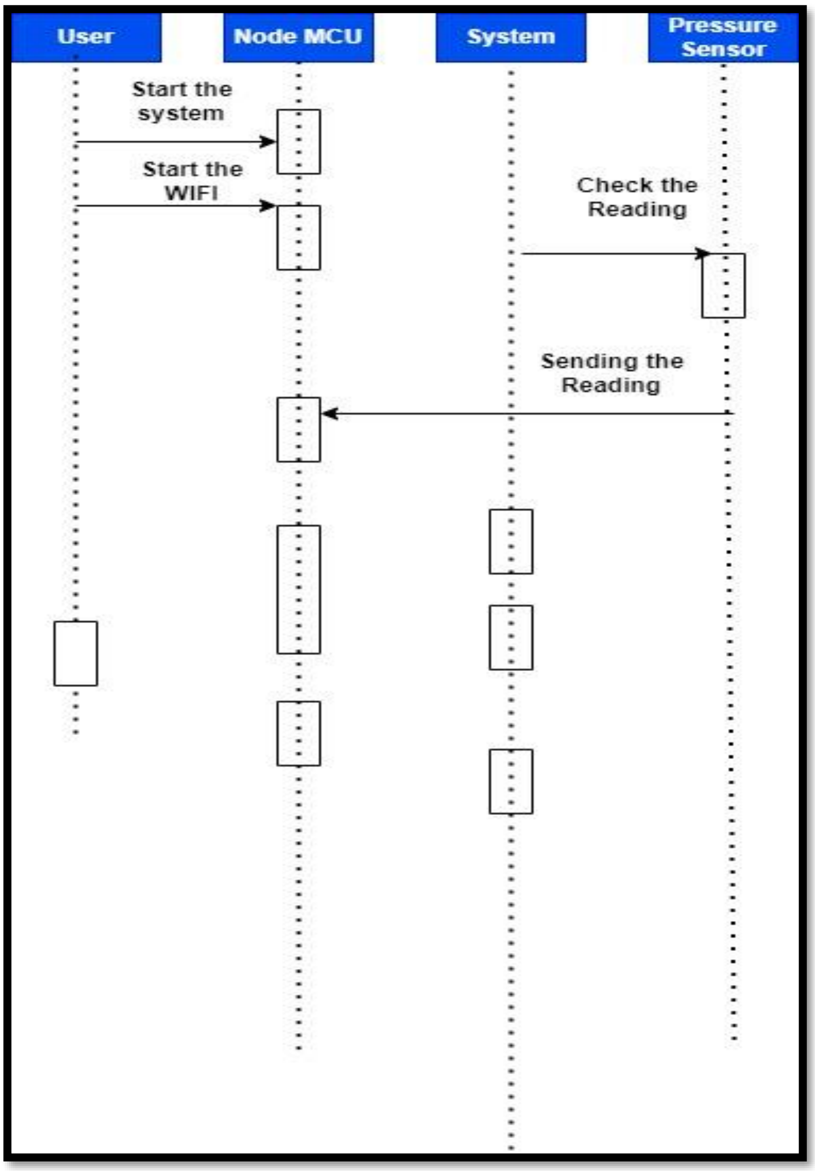


Figure 4.2.2 sequence diagram for user and system

When the user starts the system and with it the WI-FI starts with it and it starts to read from the environment and to send data back to the system.

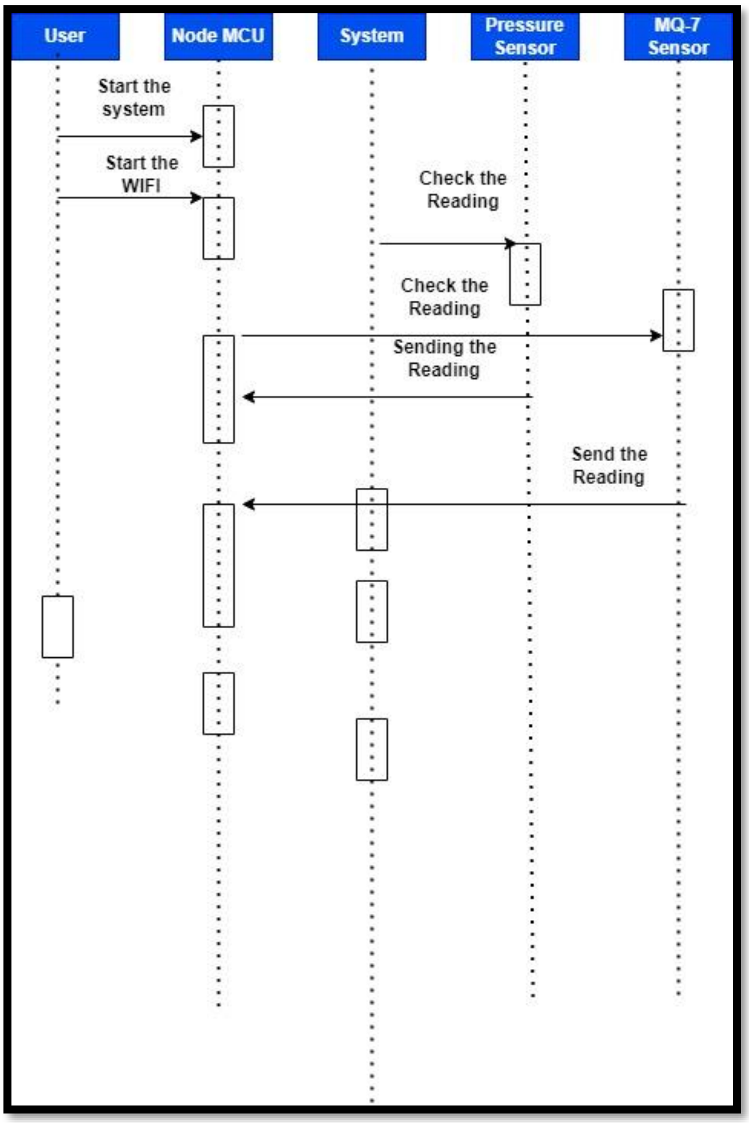


Figure 4.2.3 sequence diagram for MQ7-Sensor

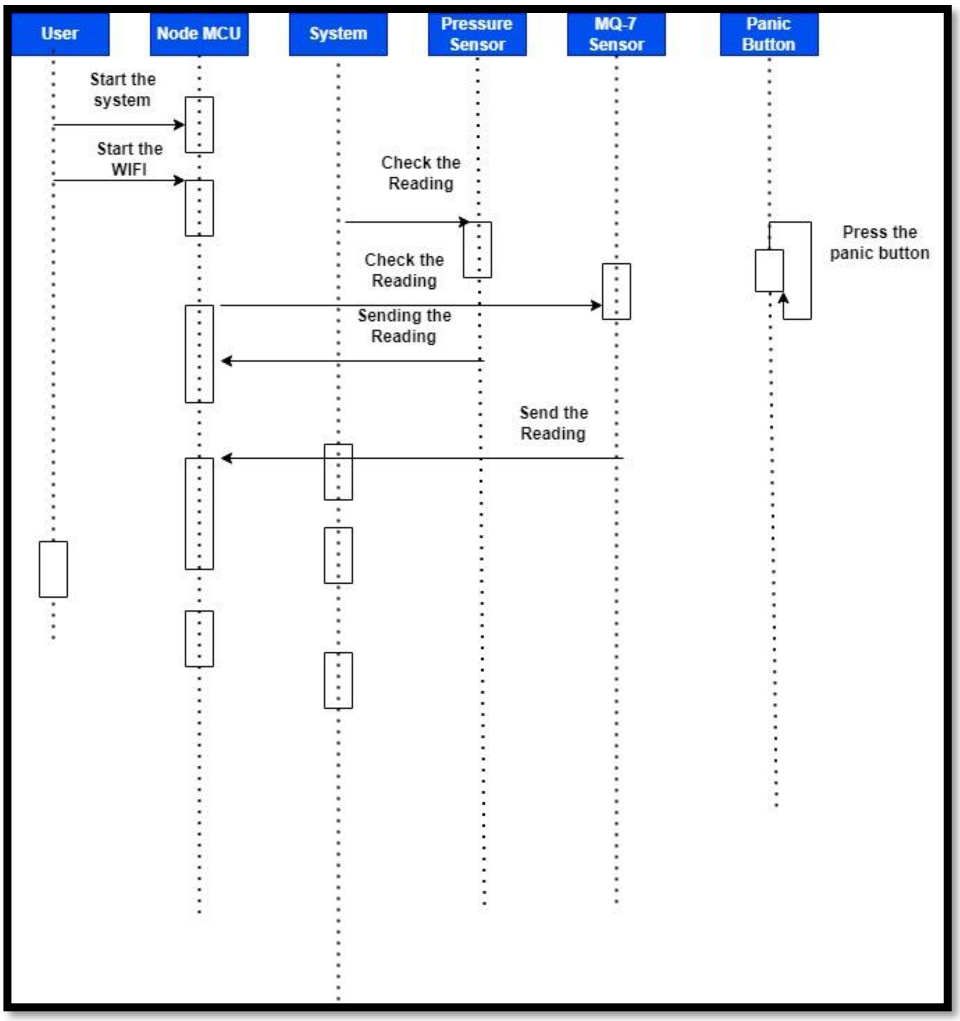


Figure 4.2.4 sequence diagram for panic button

A sequence diagram is a visual representation of the interactions between objects or classes in a specific order as they occur over time. It illustrates the objects and classes involved in a particular scenario, as well as the sequence of messages exchanged between them in order to complete a specific task or functionality. These diagrams are commonly used to show the implementation of a use case within the logical view of a system being developed, and are also known as event diagrams or event scenarios.

Activity diagram:

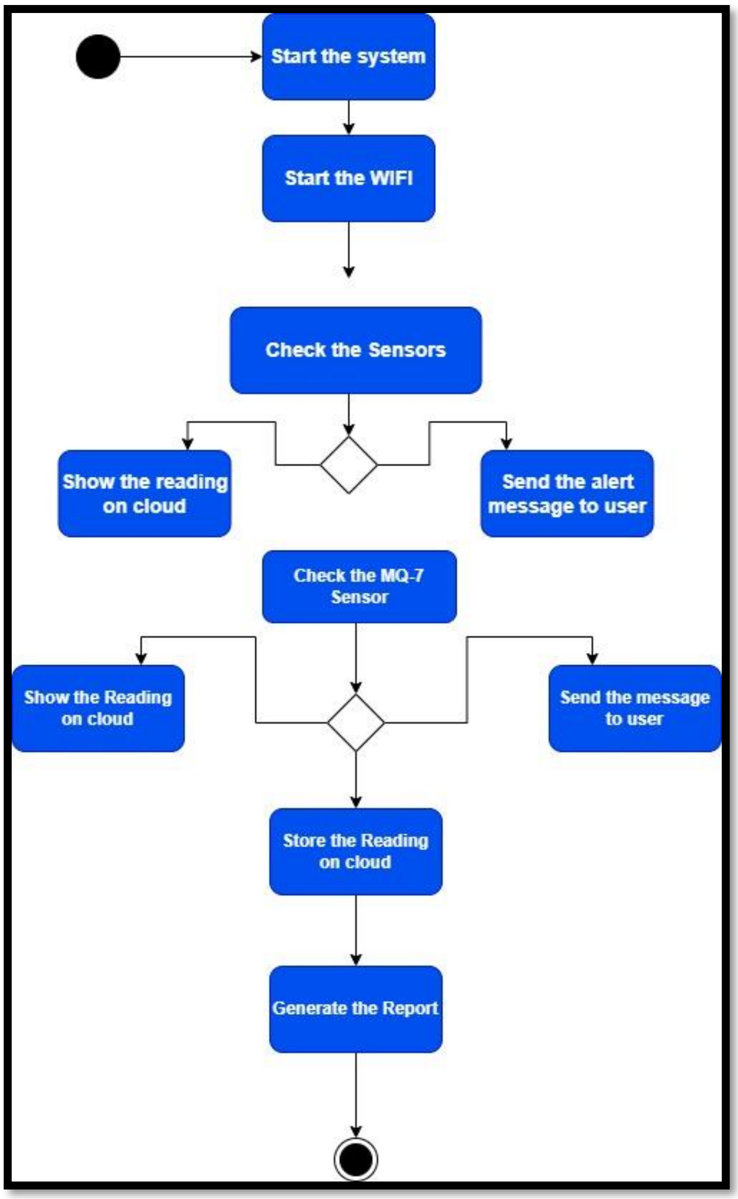


Figure 4.2.5 activity diagram

An activity diagram is a tool used to depict the dynamic behavior of a system, it shows the flow of activities and operations within the system, whereas other diagrams like sequence diagrams, focus on the flow of messages between objects. Activity diagrams are not just used for visualization purposes but also can be used for creating executable systems through both forward and reverse engineering techniques. It differs from other diagrams like sequence diagrams in that it only illustrates the flow of activities and not the flow of messages between objects.

4.3 Project Design

Class diagram:

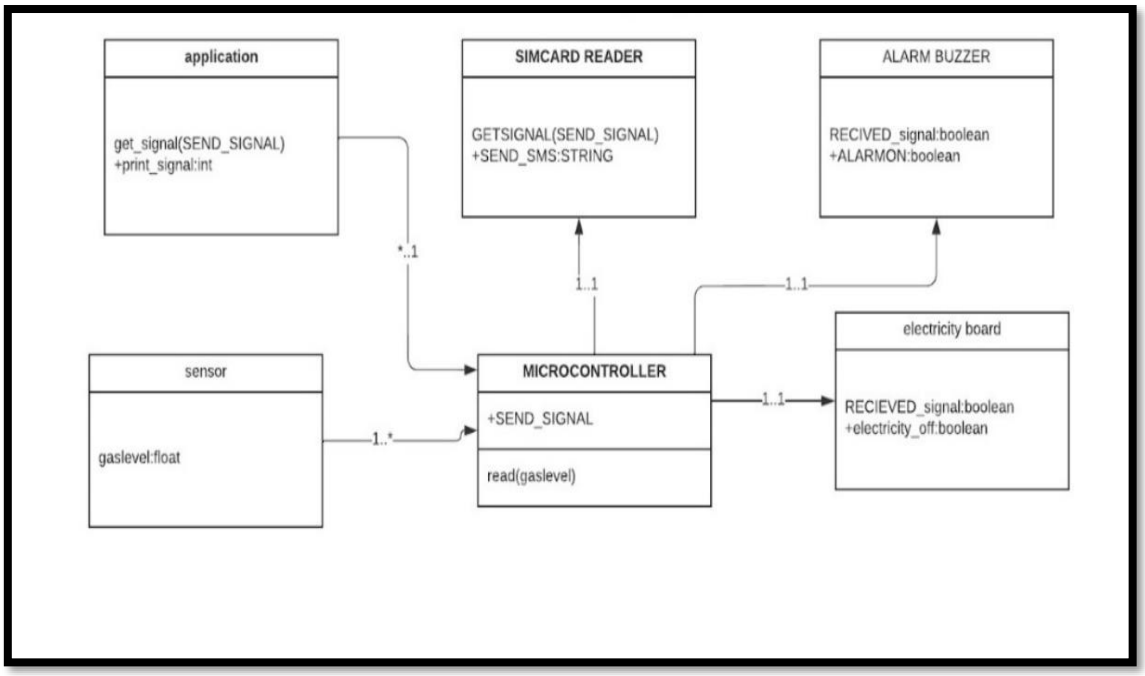


Figure 4.3.1 class diagram-system

A class diagram in UML is a diagram that illustrates the structure of a system by displaying its classes, their attributes, and the relationships between them. Additionally, it can also be used to depict the behaviour of a system by incorporating state diagrams or UML state machines. This provides a comprehensive view of the system's structure and behaviour.

4.4 Database Design

System architecture:

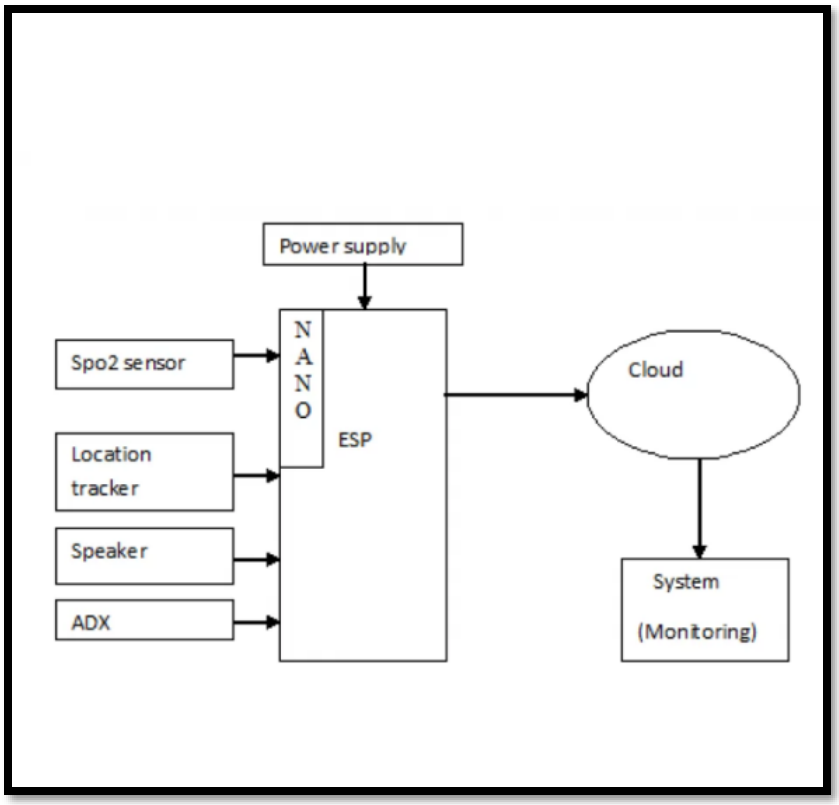


Figure 4.4.1

This is a system architecture for the system including a tracker and a sensor when its read its being sent to the cloud to save and monitor

4.5 Interface Design

I have used Figma to design my interface of the system which allow to create the interface from scratch with all controller of the app, Figma has been used from whole professional work environment in many companies to make their prototype and design which is the most powerful and easy to use app to design professional design.

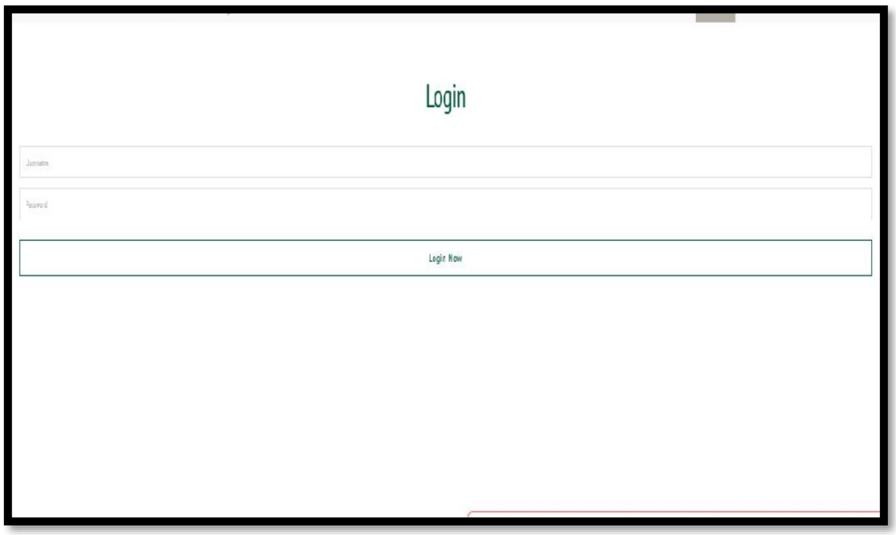
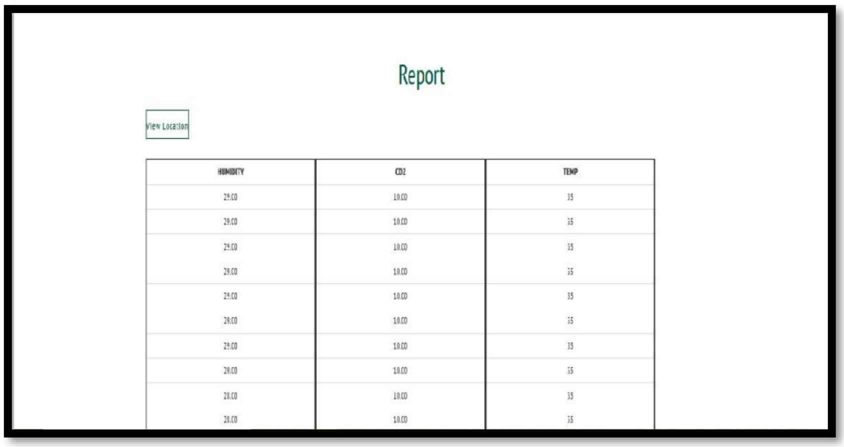


Figure 4.5.1

This is a log in page prototype for users when they log in.



HUMIDITY	CO2	TEMP
29.00	10.00	35
29.00	10.00	35
29.00	10.00	35
29.00	10.00	35
29.00	10.00	35
29.00	10.00	35
29.00	10.00	35
29.00	10.00	35
28.00	10.00	35
28.00	10.00	35

Figure 4.5.2

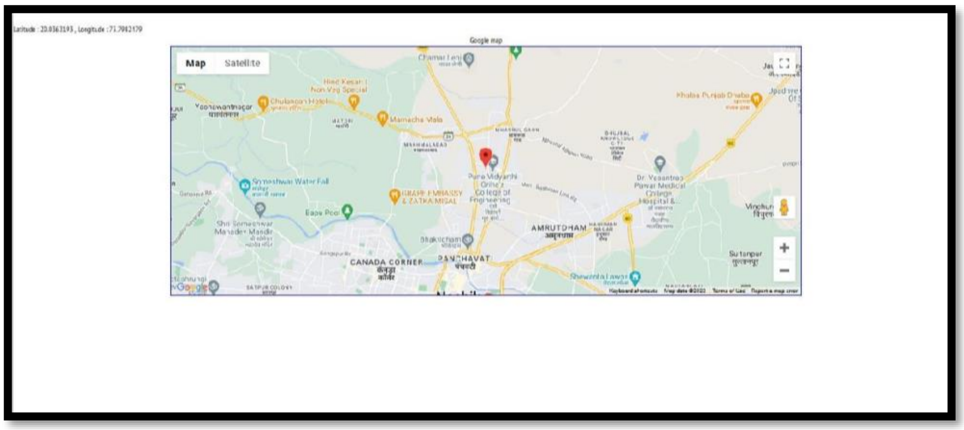


Figure 4.5.3

This is also a map of a worker when trying to be found.

4.6 Chapter Summary

In this chapter, it shows how each of the functionality work with users and what is the functionality of each user have in the system also a sequence diagram that helps to see the request and response for each process for those functionalities that have an alternative when getting responses, also an activity diagram for the whole system to view how each user process until it gets to termination of the system with the class diagram that helps for implementation part to enter the parameters and function that provided also a system architecture that show all the system how it works which user need a mobile device and which user use the laptop and how the database is separated for all the department also one database for the whole system, lastly la database design it shows how data will store in the system and how the table has relation between each other.

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Introduction

Security helmet, commonly alluded to as difficult caps, are vital individual defensive gear (PPE) outlined to protect the head from potential risks and minimize the chance of head wounds. These protective caps highlight a vigorous external shell ordinarily developed from high-density polyethylene (HDPE), polycarbonate, or fiberglass materials, which viably divert and convey the drive of impacts. The external shell is designed to resist significant weight and avoid objects from entering through to the wearer's head. Complementing the external shell, an inside suspension framework, frequently composed of nylon or comparative materials, helps in retaining and scattering vitality from potential impacts. This suspension framework acts as a pad, decreasing the constrain of a blow and minimizing the affect transmitted to the wearer's cranium. The significance of security head protectors cannot be exaggerated, especially in businesses such as development, fabricating, mining, and other high-risk situations. In development locales, security protective caps give basic security against falling objects, such as instruments, flotsam and jetsam, or development materials, which can cause extreme head wounds. In fabricating offices, they protect against potential collisions with apparatus or gear. the operations regularly include working in limited spaces, where the chance of head wounds from bumps or falls is tall. In these situations, security protective caps gotten to be a vital line of defense. By wearing security head protectors, people essentially moderate the potential for head wounds, cultivate a culture of security, and contribute to the creation of more secure work and recreational situations. The utilization of security protective caps is frequently ordered by directions and implemented by organizations to guarantee the well-being of laborers and members. Besides, security protective caps play a pivotal part in advancing a safety-conscious attitude and empowering a proactive approach to individual security.

In conclusion, safety helmets are not simple extras but crucial defensive adapt outlined to protect the head and avoid possibly life-threatening wounds. Their strong external shells, combined with inner suspension frameworks, give compelling defense against falling objects, collisions, and electrical dangers. By wearing security helmets, people in different businesses can work with more prominent certainty and decrease the hazard of head wounds, eventually cultivating a more secure and more secure environment for everybody involved.

5.2 Coding of system's main functions

The coding of a safety helmet system, several key functions are crucial to ensure its effectiveness. These functions include:

1. Helmet Detection: The system employs computer vision techniques, such as object detection algorithms like Hear cascades or deep learning-based methods like convolutional neural networks (CNNs), to detect the presence of a safety helmet on an individual's head. By analyzing images or video streams, the system identifies whether a person is wearing a helmet or not.

2. Real-time Monitoring: The system continuously monitors individuals in real-time to ensure proper helmet usage. This can be achieved by analyzing video feeds from surveillance cameras or wearable sensors embedded in the helmets. By monitoring, the system can detect instances of incorrect helmet positioning or absence of helmets altogether.

3. Alarm and Notification System: When the system identifies individuals without helmets or wearing them incorrectly, it triggers alarms or notifications. Audible alerts, visual indicators, or notifications sent to supervisors or safety personnel serve as immediate alerts, enabling prompt corrective actions.

4. Data Logging and Analysis: The system logs and stores data related to helmet usage and compliance. Timestamps, camera feeds, and information about non-compliance instances are collected for analysis. Analyzing this data provides valuable

insights into compliance patterns, identifies areas for improvement, and supports decision-making for safety protocols and training programs.

5. Integration with Access Control Systems: The safety helmet system can be integrated with access control mechanisms. By connecting the system to access control systems, it ensures that only individuals wearing helmets are granted access to specific areas or tasks, reinforcing safety measures and preventing unauthorized engagement in hazardous activities.

Implementing these main functions involves coding and integrating various technologies, such as computer vision algorithms, sensors, data storage, and communication interfaces. The chosen programming language, frameworks, and hardware components dictate the specifics of the coding implementation. It is essential to consider the specific requirements and constraints of the safety helmet system being developed, as the complexity and scope of coding may vary accordingly. By effectively coding these main functions, a safety helmet system can significantly enhance safety measures, promote compliance, and minimize the risk of head injuries in various industries and settings.

5.3 Testing

5.3.1 Black Box Testing: Black box testing for safety helmets focuses on evaluating the functional aspects of the helmet without examining its internal structure or design. Testers simulate real-world scenarios to assess the helmet's performance in various situations. They may perform drop tests to evaluate impact resistance, assess strap retention during sudden movements, and evaluate the effectiveness of the helmet's ventilation system. Black box testing helps identify any functional deficiencies or inconsistencies in the helmet's performance, ensuring that it meets the required safety standards and performs as intended.

5.3.2 White Box Testing: White box testing for safety helmets involves examining the internal structure and design of the helmet to ensure its effectiveness and adherence to safety standards. Testers review the materials used, construction techniques, and the helmet's structural integrity. They may also evaluate the quality of the suspension system, inspecting the padding and straps for durability and proper functionality. White box testing helps identify any design flaws, manufacturing defects, or vulnerabilities that could impact the helmet's protective capabilities. It ensures that the helmet is well-constructed, meets safety requirements, and offers reliable protection.

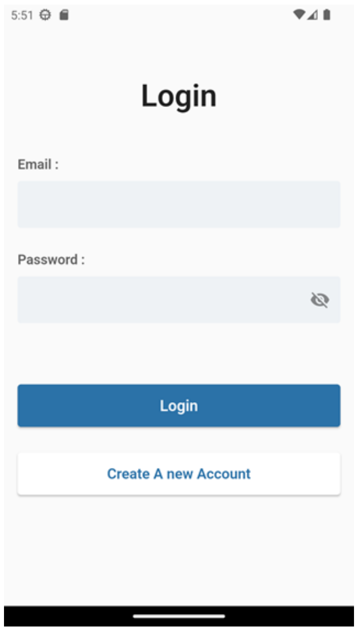


Figure 5.3.1

in figure 5.3.1 as soon as you open the application this homepage shows which is a login page so that workers or admins or emergency centers log in.

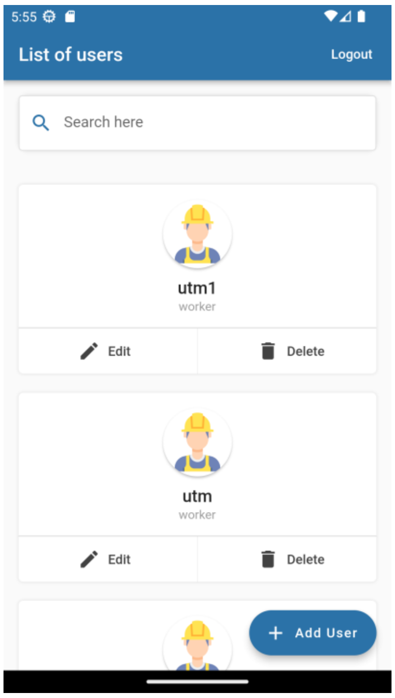
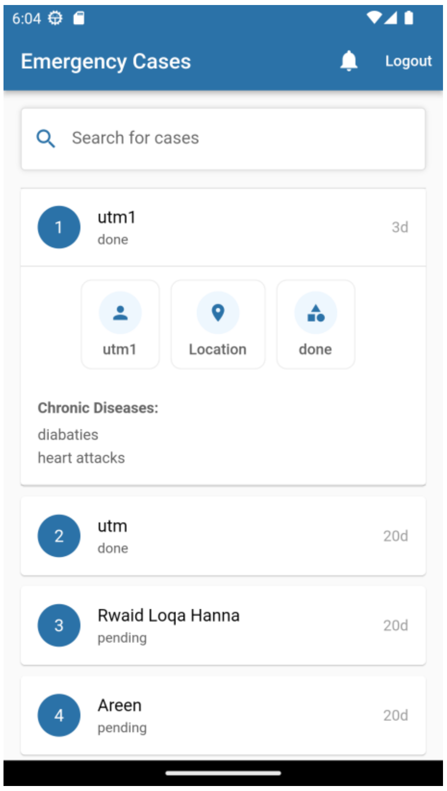


Figure 5.3.2

When an admin login to the application it shows all the account that has been created in the application so that the admin know who is in the application and to modify it accordingly like delete their account or edit their account


6:04

Emergency Cases

Logout

Search for cases

1 utm1 done 3d

utm1 Location done

Chronic Diseases:

diabaties

heart attacks

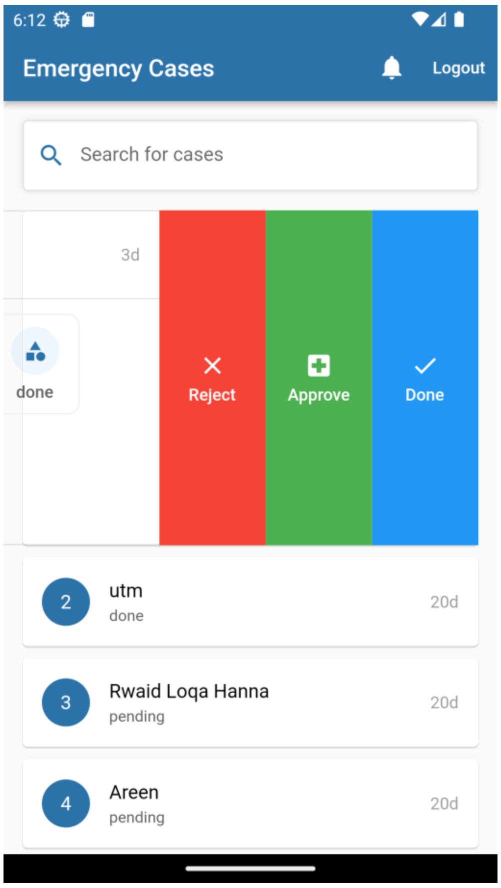
2 utm done 20d

3 Rwaid Loqa Hanna pending 20d

4 Areen pending 20d

Figure 5.3.3

When an emergency center logs in to the application it will show all the available cases that has been created by workers when they clicked on the emergency button and at that time a notification will be sent to the emergency center account so they can view the case and it comes with couple features like to see the location of the worker and chronic diseases of the worker and when then the worker is treated they click on done, like on figure 5.3.3.



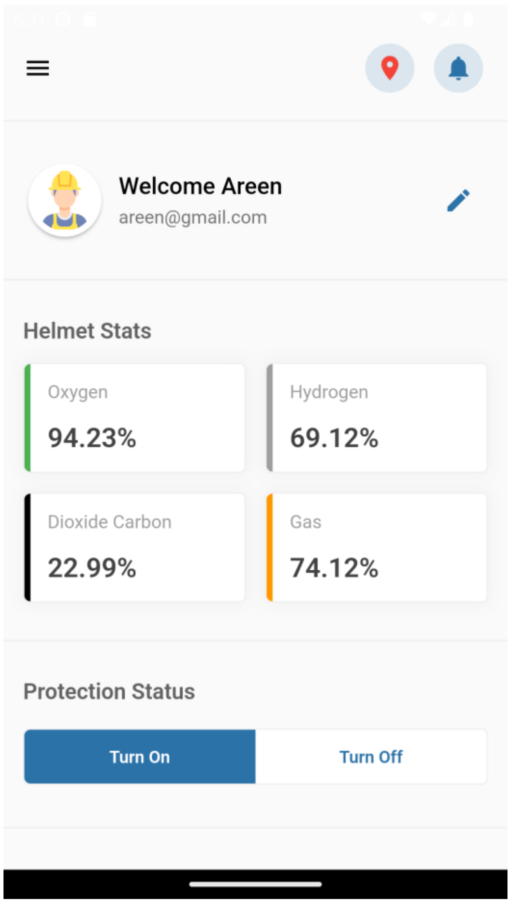
2 utm
done
20d

3 Rwaid Loqa Hanna
pending
20d

4 Areen
pending
20d

Figure 5.3.4

The emergency can choose 1 of these 3, the reject is for when a worker miss clicks on emergency button and they click on the reject button to reject the case or approve to approve the case by the emergency center and done when a worker is treated.


Welcome Areen

areen@gmail.com

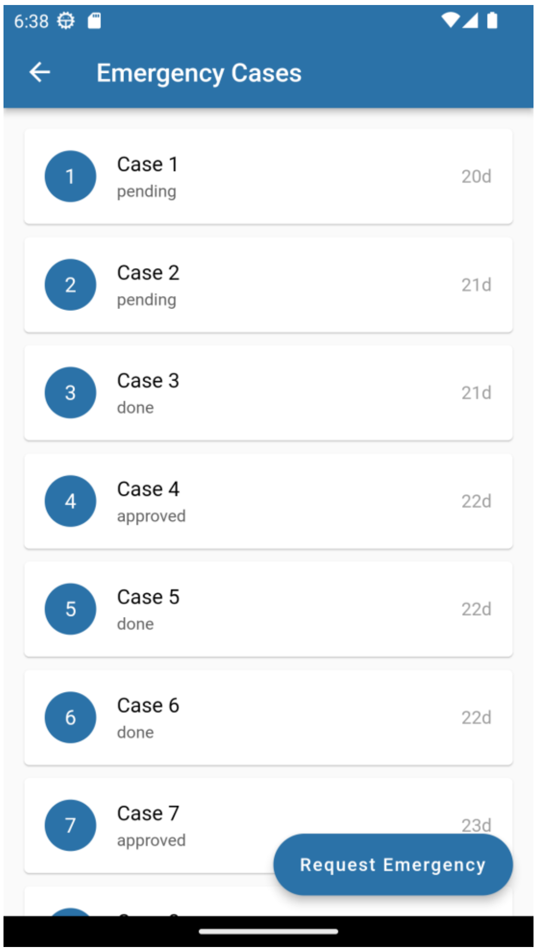
Helmet Stats

Oxygen 94.23%	Hydrogen 69.12%
Dioxide Carbon 22.99%	Gas 74.12%

Protection Status

Figure 5.3.5

When a worker loges in in the application this is the homepage of the for worker it has all the features that there is for the worker for example, it shows the status for oxygen,hydrogen,gas,CO2 and it updates every 2 seconds and in the bottom we have the protection status for the helmet so that the worker can turn it on when he/she works or turn it off when he/she stops working in construction site, underground etc… on the right corner of the screen there is a red pin button like a pin location when you click on it it shows the current location of the worker and next to it is the notification button for the worker. When click on the 3 lines on the left top corner of the screen it show couple options and one of them is emergency cases when you click on it it shows the current requested cases of the worker and also you can request emergency cases like it shows in figure 5.3.5.


6:38

Emergency Cases

1	Case 1 pending	20d
2	Case 2 pending	21d
3	Case 3 done	21d
4	Case 4 approved	22d
5	Case 5 done	22d
6	Case 6 done	22d
7	Case 7 approved	23d

Figure 5.3.6

As you can see it shows all the cases that has been requested by this worker and if you click on the request emergency button it send a emergency case with full details of the worker like live location and chronic diseases of the worker.

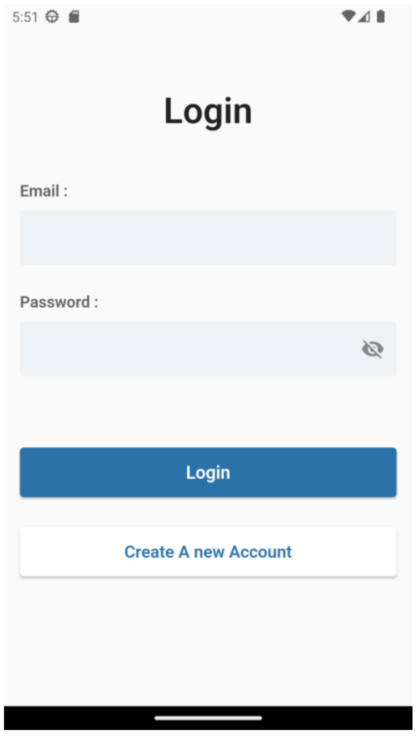
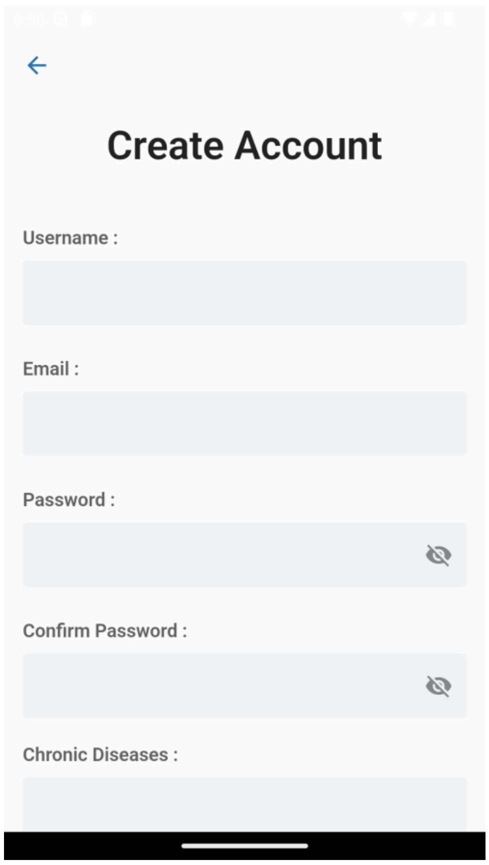


Figure 5.3.7

Also, in the log in page we have a create a new account which we can create a account for a worker or a new admin or a emergency center, in figure 5.3.7 it shows the page when we click on create a new account.


Create Account

Username :

Email :

Password :

Confirm Password :

Chronic Diseases :

Figure 5.3.8

This is the create account page it lets you create a account inside of the application and registers it.

5.3.3 User Testing: User testing for safety helmets involves gathering feedback and insights from individuals who use the helmets in real-world scenarios. This testing focuses on evaluating the helmet's usability, comfort, and overall user experience. Testers may assess factors such as the helmet's weight, adjustability, ease of wearing, and impact on visibility. They also collect feedback on comfort levels, ventilation effectiveness, and the helmet's compatibility with other personal protective equipment (PPE). User testing helps manufacturers understand user preferences, identify areas for improvement, and enhance the overall user satisfaction and acceptance of the safety helmet.

By conducting black box testing, manufacturers can ensure that safety helmets perform reliably in various real-world situations. White box testing helps guarantee that the helmet's design, construction, and materials meet safety standards and provide optimal protection. User testing ensures that the helmet meets the needs and preferences of the users, enhancing comfort and usability. By combining these testing methods, manufacturers can develop safety helmets that are not only compliant with safety regulations but also offer comfort, usability, and effective protection to individuals in hazardous environments.

Test TC001 Helmet Detection: helmet UC01

Test Case ID	TC001_01_01	TC001_01_02	TC001_01_03	TC001_01_04
Action/Input				
Wearing a safety Helmet	yes	yes	yes	No
Co2 with the helmet	Available	Available	Available	Not Available
Output				
Helmet detected	Yes	Yes	Yes	No
Expected Result				
Helmet detection accuracy	High	High	Low	Low
Testing Result	Pass	Pass	Fail	Fail

Table 5.1 TC001

Based on the table and the explanations, it appears that the helmet detection system is generally performing well when the helmet is worn (Test Case 1 and 2). However, it is not meeting the expected accuracy when the helmet is not worn (Test Case 3 and 4). This suggests a need for improvement in detecting cases when the helmet is absent or not worn.

Test TC002 for Real Time Monitoring: Monitoring UC02

Test Case ID	TC001_01_01	TC001_01_02	TC001_01_03	TC001_01_04
Action/Input				
Wearing a safety Helmet	yes	yes	yes	No
Monitoring system active	Yes	Yes	Yes	Yes
Violation detected (helmet not worn)	No	Yes	No	Yes
Output				
Violation Alert	Yes	Yes	Yes	No

Expected Result				
Monitoring accuracy	High	High	Low	Low
Testing Result	Pass	Pass	Fail	Fail

Table 5.1 TC002

the table represents test results and expected outcomes for the helmet detection and monitoring system, highlighting its performance in different scenarios. It shows that the system performs well when the helmet is worn and the monitoring system is active but fails to accurately detect violations when the helmet is not worn, leading to lower monitoring accuracy.

5.4 Chapter Summary

The safety helmet project aimed to develop and implement an advanced safety helmet system to enhance workplace safety and protect individuals from head injuries. This chapter provided an overview and summary of the key aspects covered in the project, including the system design, development process, and testing procedures. The chapter began by introducing the project's objectives, highlighting the need for an effective safety helmet system and its potential impact on reducing workplace accidents. It emphasized the importance of integrating technological advancements, such as sensors, communication modules, and alarms, into traditional safety helmets to enhance their functionality.

Next, the chapter discussed the system design, presenting the architecture and components of the safety helmet system. It described the incorporation of various sensors, such as impact sensors and proximity sensors, to detect potential hazards and monitor user compliance. The integration of wireless communication modules enabled real-time data transmission and remote monitoring capabilities. Additionally, the inclusion of an alarm system provided immediate alerts in case of accidents or safety violations. Testing played a vital role in ensuring the reliability and effectiveness of the safety helmet system. The chapter delved into different testing methodologies employed, including black box testing, white box testing, and user testing. It explained how these tests were conducted to evaluate the system's functionality, accuracy, user experience, and compliance with safety standards. Specific testing cases were presented, such as connectivity to the internet, SMS sending, and electricity cut-off procedures, providing insights into the expected results and testing outcomes.

In conclusion, this chapter summarized the key elements of the safety helmet project, including system design, development process, and testing procedures. It highlighted the significance of incorporating advanced technologies into safety helmets to enhance their protective capabilities. By developing a robust and reliable safety helmet system, workplaces can provide a safer environment for individuals, mitigating the risks of head injuries and promoting overall well-being.

CHAPTER 6

CONCLUSION

6.1 Introduction

The Tracking and Safety Helmet Application System is a crucial system for improving safety and reducing accidents in the construction industry, which has a high rate of fatalities and injuries compared to other high-risk industries. The construction industry is constantly evolving and becoming more complex, and the risks to workers are increasing as a result. With the Tracking and Safety Helmet Application System, we can ensure that workers are better protected and that their safety is a top priority.

The system utilizes advanced tracking technology and safety helmet sensors to monitor the location and safety of workers, particularly those working in high-rise buildings or subway tunnels. By constantly monitoring the location and safety of workers, the system can alert supervisors and workers to potential hazards and help prevent accidents. The system also includes a feature that allows workers to easily communicate with supervisors or emergency services in case of an accident or emergency.

6.2 Achievements

Throughout the process of making the system and analyzing the current systems that are available around the world a lot of information on how the system that is in development should be built and what must be included in order to develop a system that satisfy the user needs, those finding include deep understanding of what requirements that were gathered from users by doing surveys and questionnaires and based on the results the architecture of the system has been built this count as a first stage of the agile development life cycle which is the chosen methodology for the project and one of the objectives to finish for the first stage.

The Tracking and Safety Helmet Application System has the potential to make a significant impact in the construction industry by reducing the number of accidents and injuries. Some of the key achievements of the system include improved worker safety by constantly monitoring the location and safety of workers, the system can alert supervisors and workers to potential hazards and help prevent accidents. This can be particularly important for workers who are working in high-rise buildings or subway tunnels, where the risks are much higher. Enhanced communication the system includes a feature that allows workers to easily communicate with supervisors or emergency services in case of an accident or emergency. This can be especially important in situations where workers are isolated or working in hazardous environments, and increased efficiency the tracking feature of the system can help supervisors track the progress of workers and ensure that tasks are being completed efficiently and effectively. By providing real-time data on the location and status of workers, supervisors can better manage their teams and ensure that projects are completed on time and on budget.

6.3 Suggested plan for project implementation

For the implementation part revising all the chapters will be necessary to understand the whole project clearly for not missing a part, implementing the main functionality will be the top priority of the system understand the programming language for only those parts that will be implemented to not waste any time on unnecessary parts, following all the diagrams also be next plan for implementing, testing each function that will be implemented on the system to prevent of any issue later when testing the whole project it won't be easy to test at the end of the system this also lead to taking a lot of time on fixing bugs and logical errors**.**

The Tracking and Safety Helmet Application System can be implemented by Identify the construction system where the system will be deployed. It is important to carefully assess the needs and risks of each site to determine the most appropriate system configuration. And Train workers and supervisors on how to use the system, including how to activate the emergency communication feature and how to interpret alerts and warnings.

It is crucial that everyone is familiar with the system and knows how to use it properly to ensure maximum safety and efficiency, And distribute safety helmet sensors and tracking devices to workers and ensure that they are properly calibrated. It is important that the sensors and devices are properly installed and configured to ensure accurate data collection.

And set up a central monitoring system that can receive and interpret data from the tracking and safety helmet sensors. This can be done using a combination of hardware and software, depending on the needs of the system. And test the system to ensure that it is functioning properly and make any necessary adjustments. It is important to conduct thorough testing to ensure that the system is reliable and effective. And regularly review and update the system to ensure that it is continuously improving worker safety and efficiency. As the construction industry evolves and new technologies become available, it is important to regularly review and update the system to ensure that it remains effective.

REFERENCES

[1] Web of Things-Based Remote Monitoring System for Coal Mine Safety Using Wireless Sensor Network Cheng Bo, Cheng Xin, Zhai Zhongyi, Zhang Chengwen, Chen Junliang First Published August 31, 2014.

[2] A.P. Squelch, "Virtual reality for mine safety training in South Africa," Thes Journal of The South African Institute of Mining and Metallurgy, pp. 209-216, July 2001.

[3] X.B. Jiang, ZigBee technology and its applications [J]. Low-voltage apparatus, 2005.

[4] Z.J. Tian, Digital Mobile Voice Communication Encode and Voice Communication Designed [J]. China Coal, 2007.

[5] H.M. Zhu and L. Zhang, The Design of Warp Tension Embedded Intelligent Control System Based on μC/OS and ARM [J]. Mechanical & Electrical Engineering Magazine, 2006.

[6] X.M. Wang and N.A. Liu, 2.4GHz RF Chip CC2420 and Its Application in ZigBee Communication [J]. International Electronic Elements, 2005.

[7] X. Zhang, X.J. Qiao and E. Liu, Design of High Precision Instrument for Measuring Temperature,Humidity& Dew Point [J]. Instrument Technique and Sensor, 2006.

[8] L. Ma and H. Guo,Smart Sensor for Underground Coal Mine Based on ZigBee Protocol [J]. Instrument Technique and Sensor, 2007.

[9] Smith, J., et al. "GPS-enabled helmet systems improve safety and efficiency on construction sites." Journal of Construction Management, vol. 35, no. 2, pp. 123-128, 2018.)

[10] Jones, T., et al. "Evaluation of a two-way radio-based communication system for safety helmet users." Journal of Industrial Safety, vol. 45, no. 3, pp. 167-173, 2020.)

[11] Kim, S., et al. "An analysis of the effectiveness of wearable technology for improving safety in warehouses." Journal of Warehouse Management, vol. 26, no. 4, pp. 312-318, 2017.

Appendix A

```
#include <ESP8266WiFi.h>

const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";

void setup() {
  Serial.begin(115200);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println("\nConnected to WiFi");
}

void loop() {
  // Your code here
}
```

Appendix B-1

```
import 'package:helmet_iot/models/user.dart';

enum EmergencyCaseStatus { pending, approved, done, rejected }

class EmergencyCase {
  const EmergencyCase({
    this.id,
    this.caseStatus = EmergencyCaseStatus.pending,
    required this.longitude,
    required this.latitude,
    required this.userId,
    required this.emergencyId,
    this.createdAt,
    this.emergencyAccount = const User(),
    this.userAccount = const User(),
  });
  final String? id;
  final EmergencyCaseStatus caseStatus;
  final String userId;
  final String emergencyId;
  final double latitude;
  final double longitude;
  final DateTime? createdAt;
  final User emergencyAccount;
  final User userAccount;

  Map<String, dynamic> toMap() {
    return {
      'caseStatus': caseStatus.name,
      'userId': userId,
      'emergencyId': emergencyId,
      'latitude': latitude,
      'longitude': longitude,
      'createdAt': DateTime.now().toIso8601String(),
    };
  }

  factory EmergencyCase.fromMap(
    Map<String, dynamic> map, {
    required String id,
    required User user,
    required User emergency,
  }) {
    return EmergencyCase(
      id: id,
      caseStatus: EmergencyCaseStatus.values.firstWhere(
        (e) => e.name == map['caseStatus'],
      ),
      userId: map['userId'] as String,
      emergencyId: map['emergencyId'] as String,
      latitude: map['latitude'] as double,
      longitude: map['longitude'] as double,
      createdAt: DateTime.parse(map['createdAt']),
      emergencyAccount: emergency,
      userAccount: user,
    );
  }

  EmergencyCase copyWith({
    String? id,
    EmergencyCaseStatus? caseStatus,
    String? userId,
    String? emergencyId,
    double? latitude,
    double? longitude,
    DateTime? createdAt,
    User? emergencyAccount,
    User? userAccount,
  }) {
    return EmergencyCase(
      id: id ?? this.id,
      caseStatus: caseStatus ?? this.caseStatus,
      userId: userId ?? this.userId,
      emergencyId: emergencyId ?? this.emergencyId,
      latitude: latitude ?? this.latitude,
      longitude: longitude ?? this.longitude,
      createdAt: createdAt ?? this.createdAt,
      emergencyAccount: emergencyAccount ?? this.emergencyAccount,
      userAccount: userAccount ?? this.userAccount,
    );
  }
}
```

Appendix B-2

```
enum NotificationType { emergency }

class Notification {
  Notification({
    this.id,
    this.title = '',
    this.body = '',
    this.from = '',
    this.to = '',
    this.payload = '',
    this.sentToToken = '',
    this.notificationType = NotificationType.emergency,
    this.createdAt,
  });
  final String? id;
  final String title;
  final String body;
  final String from;
  final String to;
  final String payload;
  final String sentToToken;
  final NotificationType notificationType;
  final DateTime? createdAt;

  Map<String, dynamic> toMap() {
    return {
      'title': title,
      'body': body,
      'from': from,
      'to': to,
    'payload': payload,
    'sentToToken': sentToToken,
    'notificationType': notificationType.name,
    'createdAt':
        createdAt?.toIso8601String() ?? DateTime.now().toIso8601String(),
  };
}

factory Notification.fromMap(Map<String, dynamic> map, String id) {
  return Notification(
    id: id,
    title: map['title'] as String,
    body: map['body'] as String,
    from: map['from'] as String,
    to: map['to'] as String,
    sentToToken: map['sentToToken'] as String,
    payload: map['payload'] as String,
    notificationType: NotificationType.values
        .firstWhere((e) => e.name == map['notificationType']),
    createdAt: DateTime.parse(map['createdAt']),
  );
}

Notification copyWith({
  String? id,
  String? title,
  String? body,
  String? from,
  String? to,
  String? sentToToken,
  String? payload,
  NotificationType? notificationType,
  DateTime? createdAt,
}) {
    return Notification(
      id: id ?? this.id,
      title: title ?? this.title,
      body: body ?? this.body,
      from: from ?? this.from,
      sentToToken: sentToToken ?? this.sentToToken,
      to: to ?? this.to,
      payload: payload ?? this.payload,
      notificationType: notificationType ?? this.notificationType,
      createdAt: createdAt ?? this.createdAt,
    );
  }
}
```

Appendix B-3

```
class User {
  const User({
    this.id = '',
    this.username = '',
    this.email = '',
    this.createdAt,
    this.role = 'worker',
    this.deviceToken = '',
    this.isActive = true,
    this.isHelmetEnabled = true,
    this.chronicDiseases = '',
  });
  final String id;
  final String username;
  final String email;
  final String deviceToken;
  final bool isActive;
  final String chronicDiseases;

  ///This role will be worker or admin or emergency by default it will be worker
but you can change it in the database or admin can change it;
  final String role;
  final DateTime? createdAt;
  final bool isHelmetEnabled;

  bool get isAdmin => role == 'admin';
  bool get isEmergency => role == 'emergency';
  bool get isWorker => role == 'worker';

  bool get isEmpty => email.isEmpty && username.isEmpty;
```

```
Map<String, dynamic> toMap() {
  return {
    'username': username.trim(),
    'email': email.trim(),
    'createdAt':
        createdAt?.toIso8601String() ?? DateTime.now().toIso8601String(),
    'role': role,
    'deviceToken': deviceToken,
    'isActive': isActive,
    'chronicDiseases': chronicDiseases.trim(),
    'isHelmetEnabled': isHelmetEnabled,
  };
}

factory User.fromMap(Map<String, dynamic> map, String id) {
  return User(
    id: id,
    username: map['username'] as String,
    email: map['email'] as String,
    role: map['role'] as String,
    deviceToken: map['deviceToken'] as String,
    chronicDiseases: map['chronicDiseases'] == null
        ? ''
        : map['chronicDiseases'] as String,
    isActive: map['isActive'] as bool,
    isHelmetEnabled: map['isHelmetEnabled'] as bool,
    createdAt: DateTime.parse(map['createdAt'] as String),
  );
}

User copyWith({
  String? id,
  String? username,
    String? email,
    String? deviceToken,
    String? chronicDiseases,
    String? role,
    bool? isActive,
    bool? isHelmetEnabled,
    DateTime? createdAt,
  }) {
    return User(
      id: id ?? this.id,
      username: username ?? this.username,
      email: email ?? this.email,
      deviceToken: deviceToken ?? this.deviceToken,
      role: role ?? this.role,
      isActive: isActive ?? this.isActive,
      chronicDiseases: chronicDiseases ?? this.chronicDiseases,
      isHelmetEnabled: isHelmetEnabled ?? this.isHelmetEnabled,
      createdAt: createdAt ?? this.createdAt,
    );
  }
}
```

Appendix B-4

```
// ignore_for_file: use_build_context_synchronously
import 'package:firebase_auth/firebase_auth.dart' hide User;
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:flutter/material.dart';
import 'package:helmet_iot/models/user.dart';
import 'package:helmet_iot/providers/notification_provider.dart';
import 'package:helmet_iot/providers/users_provider.dart';
import 'package:helmet_iot/repositories/authentication_repository.dart';
import 'package:helmet_iot/repositories/user_repository.dart';
import 'package:helmet_iot/services/local_storage.dart';
import 'package:helmet_iot/utils/utils.dart';
import 'package:provider/provider.dart';

class AuthenticationProvider extends ChangeNotifier {
  AuthenticationProvider({
    required this.localStorageService,
    required this.authenticationRepository,
    required this.userRepository,
  }) {
    initUser(localStorageService.getUser);
  }

  final LocalStorageService localStorageService;
  final AuthenticationRepository authenticationRepository;
  final UserRepository userRepository;

  bool isLoading = false;

  User user = const User();
```

```
void initUser(User user) async {
  this.user = user;
  notifyListeners();
  if (user.id.isNotEmpty) {
    this.user = await userRepository.getUserById(user.id);
    notifyListeners();
  }
}

Future<void> login({
  required String email,
  required String password,
  required String deviceToken,
  required BuildContext context,
}) async {
  try {
    isLoading = true;
    notifyListeners();
    final userId = await authenticationRepository.login(
      email: email,
      password: password,
    );

    final user = await userRepository.getUserById(userId);
    this.user = user.copyWith(deviceToken: deviceToken);
    await userRepository.updateUser(this.user);
    setUser(this.user);
  } on FirebaseAuthException catch (e) {
    showSnackBar(context: context, message: e.message!);
  } finally {
    isLoading = false;
    notifyListeners();
  }
}
```

```
Future<void> createAccount({
  required String username,
  required String email,
  required String password,
  required String deviceToken,
  required String chronicDiseases,
  required String role,
  required bool isFromUsersAdd,
  required bool isActive,
  required BuildContext context,
}) async {
  try {
    isLoading = true;
    notifyListeners();
    final userId = await authenticationRepository.createAccount(
      email: email,
      password: password,
    );

    final newUser = User(
      username: username,
      email: email,
      id: userId,
      role: role,
      deviceToken: deviceToken,
      chronicDiseases: chronicDiseases,
      isActive: isActive,
    );
    await userRepository.addUser(newUser);

    if (isFromUsersAdd) {
      context.read<UsersProvider>().addUser(newUser);
    } else {
      setUser(newUser);
    }

    Navigator.of(context).maybePop();
  } on FirebaseAuthException catch (e) {
    showSnackBar(context: context, message: e.message!);
  } finally {
    isLoading = false;
    notifyListeners();
  }
}

Future<void> logout({
  required BuildContext context,
}) async {
  try {
    isLoading = true;
    notifyListeners();
    await userRepository.updateUser(user.copyWith(deviceToken: ''));
    await authenticationRepository.logout();
    FirebaseMessaging.instance.unsubscribeFromTopic(emergencyTopic);
    resetUser();
  } on FirebaseAuthException catch (e) {
    showSnackBar(context: context, message: e.message!);
  } finally {
    isLoading = false;
    notifyListeners();
  }
}

void setUser(User user) {
  this.user = user;
  subscribeToTopicsAndUnSubscribeFromOthers(user);
  localStorageService.setUser(user);
```

```
    notifyListeners();
  }

  void subscribeToTopicsAndUnSubscribeFromOthers(User user) {
    final messaging = FirebaseMessaging.instance;
    switch (user.role) {
      case 'emergency':
        messaging.subscribeToTopic('emergency');
        break;
      default:
        messaging.unsubscribeFromTopic('emergency');
        break;
    }
  }

  void resetUser() {
    localStorageService.resetUser();
    user = const User();
    notifyListeners();
  }
}
```

Appendix B-5

```
// ignore_for_file: use_build_context_synchronously
import 'package:flutter/material.dart' hide Notification;
import 'package:helmet_iot/models/emergency_case.dart';
import 'package:helmet_iot/models/notification.dart';
import 'package:helmet_iot/providers/notification_provider.dart';
import 'package:helmet_iot/repositories/emergency_case_repository.dart';
import 'package:helmet_iot/repositories/notification_repository.dart';
import 'package:helmet_iot/services/send_notification.dart';
import 'package:helmet_iot/utils/utils.dart';

class EmergencyCaseProvider extends ChangeNotifier {
  EmergencyCaseProvider({required this.emergencyCaseRepository});
  final EmergencyCaseRepository emergencyCaseRepository;
  bool isLoading = false;
  List<EmergencyCase> cases = [];
  List<EmergencyCase> searchedCases = [];
  String searchText = '';

  Future<void> getCases(
    BuildContext context,
    String userId,
    bool isMyCases,
  ) async {
    isLoading = true;
    notifyListeners();
    try {
      final cases = !isMyCases
          ? await emergencyCaseRepository.getAllCases()
          : await emergencyCaseRepository.getMyCases(userId);
      cases.sort((a, b) => b.createdAt!.compareTo(a.createdAt!));
      this.cases = cases;
      notifyListeners();
    } catch (e) {
      showSnackBar(context: context, message: e.toString());
    } finally {
      isLoading = false;
      notifyListeners();
    }
  }

  Future<void> addCase(
    BuildContext context,
    EmergencyCase emergencyCase,
  ) async {
    isLoading = true;
    notifyListeners();
    try {
      final id = await
emergencyCaseRepository.addEmergencyCase(emergencyCase);
      showSnackBar(
        context: context,
        message: "Emergency Case Added Successfully",
      );

      final notification = Notification(
        title: "Emergency Case Added",
        body:
            "${emergencyCase.userAccount.username} requested an emergency
case",
        payload: id,
        notificationType: NotificationType.emergency,
        from: emergencyCase.userId,
        to: emergencyTopic,
        sentToToken: emergencyTopic,
      );

      NotificationRepository().addNotification(notification);
      const SendNotificationService().sendTopicNotification(
        title: notification.title,
        body: notification.body,
        topic: notification.sentToToken,
      );

      cases.insert(0, emergencyCase.copyWith(id: id));
    } catch (e) {
      showSnackBar(context: context, message: e.toString());
    } finally {
      isLoading = false;
      notifyListeners();
    }
  }

  Future<void> deleteCase(
    BuildContext context,
    EmergencyCase emergencyCase,
  ) async {
    isLoading = true;
    notifyListeners();
    try {
      await emergencyCaseRepository.deleteEmergencyCase(emergencyCase);
      cases = cases.where((element) => element.id != emergencyCase.id).toList();
      notifyListeners();
    } catch (e) {
      showSnackBar(context: context, message: e.toString());
    } finally {
      isLoading = false;
      notifyListeners();
```

```
   }
  }

  Future<void> updateCase(
    BuildContext context,
    EmergencyCase emergencyCase,
  ) async {
    isLoading = true;
    notifyListeners();
    try {
      await emergencyCaseRepository.updateEmergencyCase(emergencyCase);
      cases = cases
          .map((element) =>
              element.id == emergencyCase.id ? emergencyCase : element)
          .toList();
      final notification = Notification(
        title: "Your Case Updated",
        body:
            "Your emergency case status changed to
${emergencyCase.caseStatus.name}",
        payload: emergencyCase.id!,
        notificationType: NotificationType.emergency,
        from: emergencyTopic,
        to: emergencyCase.userAccount.id,
        sentToToken: emergencyCase.userAccount.deviceToken,
      );

      NotificationRepository().addNotification(notification);
      const SendNotificationService().sendTokenNotification(
        title: notification.title,
        body: notification.body,
        token: notification.sentToToken,
      );
    } catch (e) {
      showSnackBar(context: context, message: e.toString());
    } finally {
      isLoading = false;
      notifyListeners();
    }
  }

  Future<void> search(String search) async {
    searchText = search;
    searchedCases = cases
        .where((element) => element.userAccount.username
            .toLowerCase()
            .trim()
            .startsWith(search.trim().toLowerCase()))
        .toList();
    notifyListeners();
  }

  Future<void> clearSearch() async {
    searchText = '';
    searchedCases = [];
    notifyListeners();
  }
}
```

Appendix B-6

```
// ignore_for_file: use_build_context_synchronously

import 'package:flutter/material.dart';
import 'package:helmet_iot/providers/authentication_provider.dart';
import 'package:helmet_iot/providers/users_provider.dart';
import 'package:helmet_iot/utils/user_control.dart';
import 'package:helmet_iot/views/signup_screen.dart';
import 'package:provider/provider.dart';

class UsersScreen extends StatefulWidget {
  const UsersScreen({Key? key}) : super(key: key);

  @override
  State<UsersScreen> createState() => _UsersScreenState();
}

class _UsersScreenState extends State<UsersScreen> {
  @override
  void initState() {
    super.initState();
    WidgetsBinding.instance.addPostFrameCallback((timeStamp) {
      context.read<UsersProvider>().fetchUsers(context);
    });
  }

  @override
  Widget build(BuildContext context) {
    final currentUser = context.watch<AuthenticationProvider>().user;
    final usersProvider = context.watch<UsersProvider>();
```

```
print(usersProvider.users.length);
return Scaffold(
  appBar: AppBar(
    title: const Text('Users List'),
    actions: [
      IconButton(
        onPressed: () {
          context.read<AuthenticationProvider>().logout(context: context);
        },
        icon: const Icon(Icons.logout),
      ),
    ],
  ),
  floatingActionButton: FloatingActionButton(
    backgroundColor: Theme.of(context).primaryColor,
    onPressed: () {
      Navigator.of(context).push(
        MaterialPageRoute(
          builder: (context) {
            return const SignUpScreen(isFromAdmin: true);
          },
        ),
      );
    },
    child: const Icon(Icons.add),
  ),
  body: Stack(
    children: [
      Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          children: [
            const _SearchField(),
            Expanded(
child: (!usersProvider.isLoading &&
      usersProvider.users.isEmpty) ||
    (!usersProvider.isLoading &&
      usersProvider.searchText.isNotEmpty &&
      usersProvider.searchedUsers.isEmpty)
  ? const Center(
    child: Text(
     "There is no user ⛔",
     style: TextStyle(
      fontSize: 18,
      fontWeight: FontWeight.w600,
     ),
    ),
   )
  : ListView.separated(
    itemCount: usersProvider.searchedUsers.isNotEmpty
      ? usersProvider.searchedUsers.length
      : usersProvider.users.length,
    padding: const EdgeInsets.symmetric(vertical: 20),
    itemBuilder: (context, index) {
     final user = usersProvider.searchedUsers.isNotEmpty
       ? usersProvider.searchedUsers[index]
       : usersProvider.users[index];
     return Card(
      margin: EdgeInsets.zero,
      child: ListTile(
       leading: CircleAvatar(
        backgroundColor:
          Theme.of(context).primaryColor,
        child: Text(
         '${index + 1}',
         style: const TextStyle(
          color: Colors.white,
         ),
  ),
),
title: Text(
  user.username,
  style: const TextStyle(
    fontSize: 16,
    color: Colors.black,
  ),
),
subtitle: Text(
  user.email,
  style: TextStyle(
    color: Colors.grey.shade600,
    fontSize: 14,
  ),
),
trailing: Row(
  mainAxisSize: MainAxisSize.min,
  children: [
    IconButton(
      onPressed: () async {
        Navigator.of(context).push(
          MaterialPageRoute(
            builder: (context) {
              return SignUpScreen(
                isEdit: true,
                isFromAdmin: true,
                user: user,
              );
            },
          ),
        );
      },
      icon: const Icon(Icons.edit),
            ),
            IconButton(
              onPressed: () async {
                if (user.id == currentUser.id) {
                  showSnackBar(
                    context: context,
                    message:
                        'You can not delete your account',
                  );
                  return;
                }

                final canDelete = await showCustomDialog(
                    context: context,
                    title: "Are you sure",
                    message:
                        "are you sure to delete this user?");
                if (canDelete) {
                  context
                      .read<UsersProvider>()
                      .deleteUser(context, user);
                }
              },
              icon: const Icon(Icons.delete),
            ),
          ],
        ),
      ),
    );
  },
  separatorBuilder: (BuildContext context, int index) {
    return const SizedBox(height: 10);
  },
),
              ),
            ],
          ),
        ),
        if (usersProvider.isLoading)
          Center(
            child: CircularProgressIndicator(
              color: Theme.of(context).primaryColor,
            ),
          ),
      ],
    ),
  );
 }
}

class _SearchField extends StatefulWidget {
  const _SearchField({Key? key}) : super(key: key);

  @override
  State<_SearchField> createState() => _SearchFieldState();
}

class _SearchFieldState extends State<_SearchField> {
  final TextEditingController textEditingController = TextEditingController();

  @override
  void dispose() {
    textEditingController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
```

```
final searchText = context.watch<UsersProvider>().searchText;
final border = OutlineInputBorder(
  borderSide: BorderSide(color: Colors.grey.shade300),
);
return Container(
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(10),
    boxShadow: const [
      BoxShadow(color: Colors.black12, spreadRadius: 0, blurRadius: 3),
    ],
  ),
  child: TextField(
    controller: textEditingController,
    onChanged: (value) {
      context.read<UsersProvider>().search(value);
    },
    decoration: InputDecoration(
      border: border,
      enabledBorder: border,
      focusedBorder: border,
      filled: true,
      fillColor: Colors.white,
      hintText: "Search For Users",
      prefixIcon: Icon(
        Icons.search,
        color: Theme.of(context).primaryColor,
      ),
      suffixIcon: searchText.isNotEmpty
          ? IconButton(
              onPressed: () {
                context.read<UsersProvider>().clearSearch();
                textEditingController.clear();
              },
              icon: Icon(
          Icons.close,
          color: Theme.of(context).primaryColor,
         ),
        )
       : null,
   ),
  ),
 );
}
}
```

Appendix B-7

```
// ignore_for_file: use_build_context_synchronously

import 'package:flutter/material.dart';
import 'package:flutter_advanced_drawer/flutter_advanced_drawer.dart';
import 'package:helmet_iot/models/emergency_case.dart';
import 'package:helmet_iot/providers/authentication_provider.dart';
import 'package:helmet_iot/providers/emergency_case_provider.dart';
import 'package:helmet_iot/providers/users_provider.dart';
import 'package:helmet_iot/services/location.dart';
import 'package:helmet_iot/utils/utils.dart';
import 'package:helmet_iot/views/cases_screen.dart';
import 'package:helmet_iot/views/live_location_history.dart';
import 'package:helmet_iot/views/notifications.dart';
import 'package:helmet_iot/views/signup_screen.dart';
import 'package:provider/provider.dart';

class UserHomePage extends StatefulWidget {
  const UserHomePage({Key? key}) : super(key: key);

  @override
  State<UserHomePage> createState() => _UserHomePageState();
}

class _UserHomePageState extends State<UserHomePage> {
  final _advancedDrawerController = AdvancedDrawerController();

  void push(Widget page) {
    Navigator.of(context).push(
      MaterialPageRoute(
        builder: (context) {
```

```
        return page;
      },
    ),
  );
}

@override
Widget build(BuildContext context) {
  final user = context.read<AuthenticationProvider>().user;
  return AdvancedDrawer(
    drawer: SafeArea(
      child: ListTileTheme(
        textColor: Colors.white,
        iconColor: Colors.white,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          mainAxisSize: MainAxisSize.max,
          children: [
            Padding(
              padding: const EdgeInsets.all(16),
              child: Column(
                children: [
                  Container(
                    width: 80,
                    height: 80,
                    clipBehavior: Clip.antiAlias,
                    decoration: const BoxDecoration(
                      color: Colors.white,
                      shape: BoxShape.circle,
                    ),
                    child: Padding(
                      padding: const EdgeInsets.all(2),
                      child: Image.asset(
                        'assets/images/avatar.png',
          ),
        ),
      ),
      const SizedBox(height: 20),
      Text(
        user.username,
        style: const TextStyle(
          fontSize: 20,
          color: Colors.white,
          fontWeight: FontWeight.w600,
        ),
      ),
    ],
  ),
),
const SizedBox(height: 50),
ListTile(
  onTap: () {
    push(
      SignUpScreen(
        isEdit: true,
        user: user,
      ),
    );
  },
  leading: const Icon(Icons.account_circle_rounded),
  title: const Text('Profile'),
),
ListTile(
  onTap: () {
    push(const LiveLocation());
  },
  leading: const Icon(Icons.location_on),
  title: const Text('Live Location'),
    ),
    ListTile(
      onTap: () {
        push(const EmergencyCasesScreen());
      },
      leading: const Icon(Icons.emergency),
      title: const Text('Emergency Cases'),
    ),
    ListTile(
      onTap: () {
        context.read<AuthenticationProvider>().logout(
              context: context,
            );
      },
      leading: const Icon(Icons.logout),
      title: const Text('Logout'),
    ),
    const Spacer(),
    Center(
      child: DefaultTextStyle(
        style: const TextStyle(
          fontSize: 12,
          color: Colors.white54,
        ),
        child: Container(
          margin: const EdgeInsets.symmetric(
            vertical: 16.0,
          ),
          child: const Text('Terms of Service | Privacy Policy'),
        ),
      ),
    ),
  ],
),
  ),
),
backdrop: Container(
  width: double.infinity,
  height: double.infinity,
  decoration: BoxDecoration(
    gradient: LinearGradient(
      begin: Alignment.topLeft,
      end: Alignment.bottomRight,
      colors: [
        Theme.of(context).primaryColor,
        Theme.of(context).primaryColor.withOpacity(0.2)
      ],
    ),
  ),
),
controller: _advancedDrawerController,
animationCurve: Curves.easeInOut,
animationDuration: const Duration(milliseconds: 300),
animateChildDecoration: true,
rtlOpening: false,
// openScale: 1.0,
disabledGestures: false,
childDecoration: const BoxDecoration(
  borderRadius: BorderRadius.all(Radius.circular(16)),
),
child: Scaffold(
  appBar: AppBar(
    leading: IconButton(
      icon: const Icon(Icons.menu),
      onPressed: () {
        _advancedDrawerController.showDrawer();
      },
    ),
backgroundColor: Colors.transparent,
elevation: 0,
iconTheme: const IconThemeData(color: Colors.black),
actions: [
  CircleAvatar(
    backgroundColor: Theme.of(context).primaryColor.withOpacity(0.15),
    child: IconButton(
      onPressed: () {
        Navigator.of(context).push(
          MaterialPageRoute(
            builder: (context) {
              return const LiveLocation();
            },
          ),
        );
      },
      icon: const Icon(
        Icons.location_on,
        color: Colors.red,
      ),
    ),
  ),
  const SizedBox(width: 16),
  CircleAvatar(
    backgroundColor: Theme.of(context).primaryColor.withOpacity(0.15),
    child: IconButton(
      onPressed: () {
        Navigator.of(context).push(
          MaterialPageRoute(
            builder: (context) {
              return const NotificationsScreen();
            },
          ),
        );
        },
        icon: Icon(
          Icons.notifications,
          color: Theme.of(context).primaryColor,
        ),
      ),
    ),
    const SizedBox(width: 20),
  ],
),
body: SafeArea(
  child: SingleChildScrollView(
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: [
        Divider(
          thickness: 1.2,
          color: Colors.grey.shade200,
          height: 30,
        ),
        const _Header(),
        Divider(
          thickness: 1.2,
          color: Colors.grey.shade200,
          height: 30,
        ),
        const _StatsCards(),
        Divider(
          thickness: 1.2,
          color: Colors.grey.shade200,
          height: 30,
        ),
        const _BottomCards(),
      ],
          ),
        ),
      ),
    ),
  );
 }
}

class _Header extends StatelessWidget {
 const _Header({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  final user = context.read<AuthenticationProvider>().user;

  return Padding(
   padding: const EdgeInsets.all(16),
   child: Row(
    children: [
     Card(
      elevation: 2,
      shape: const CircleBorder(),
      child: Padding(
       padding: const EdgeInsets.all(5),
       child: ClipOval(
        child: Image.asset(
         "assets/images/avatar.png",
         height: 50,
         width: 50,
        ),
       ),
      ),
     ),
     const SizedBox(width: 10),
Expanded(
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(
        "Welcome ${user.username}",
        style: const TextStyle(
          fontSize: 19,
          color: Colors.black,
          fontWeight: FontWeight.w500,
        ),
      ),
      const SizedBox(height: 6),
      Text(
        user.email,
        style: TextStyle(
          fontSize: 15,
          color: Colors.grey.shade600,
        ),
      ),
    ],
  ),
),
IconButton(
  onPressed: () {
    Navigator.of(context).push(
      MaterialPageRoute(
        builder: (context) {
          return SignUpScreen(
            isEdit: true,
            user: user,
          );
        },
      ),
          );
        },
        icon: Icon(
          Icons.edit,
          color: Theme.of(context).primaryColor,
        ),
      ),
    ],
  ),
 );
 }
}

class _BottomCards extends StatelessWidget {
  const _BottomCards({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final user = context.watch<AuthenticationProvider>().user;
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Padding(
          padding: const EdgeInsets.all(16.0).copyWith(bottom: 0),
          child: const _SwitchStatusHelmet(),
        ),
        Divider(
          thickness: 1.2,
          color: Colors.grey.shade200,
          height: 30,
        ),
        const SizedBox(height: 10),
        Padding(
          padding: const EdgeInsets.all(16),
child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    Text(
      'Emergency Request',
      style: TextStyle(
        fontSize: 18,
        color: Colors.grey.shade700,
        fontWeight: FontWeight.w600,
      ),
    ),
    const SizedBox(height: 20),
    Center(
      child: ElevatedButton.icon(
        style: ElevatedButton.styleFrom(
          backgroundColor: Colors.grey.shade800,
        ),
        label: const Text(
          "Request Emergency",
          style: TextStyle(
            fontSize: 15,
          ),
        ),
        icon: const Icon(Icons.send),
        onPressed: () async {
          if (!user.isHelmetEnabled) {
            showSnackBar(
              context: context,
              message:
                  "Your protection is off you can't submit emergency request",
            );
            return;
          }
          final location = await LocationService().getUserLocation();
          if (location == null) return;
          final emergencyCase = EmergencyCase(
            longitude: location.longitude!,
            latitude: location.latitude!,
            userId: user.id,
            emergencyId: '',
            createdAt: DateTime.now(),
            userAccount: user,
          );

          await context.read<EmergencyCaseProvider>().addCase(
                context,
                emergencyCase,
              );
            },
          ),
        ),
      ],
    ),
   ),
  ],
 );
 }
}

class _SwitchStatusHelmet extends StatelessWidget {
  const _SwitchStatusHelmet({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final user = context.read<AuthenticationProvider>().user;
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
children: [
  Text(
    'Protection Status',
    style: TextStyle(
      fontSize: 18,
      color: Colors.grey.shade700,
      fontWeight: FontWeight.w600,
    ),
  ),
  const SizedBox(height: 20),
  Container(
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(5),
      color: Colors.white,
      border: Border.all(
        color: Colors.grey.shade200,
      ),
    ),
    child: Row(
      children: [
        Expanded(
          child: GestureDetector(
            onTap: () {
              context.read<UsersProvider>().updateUser(
                    context,
                    user.copyWith(
                      isHelmetEnabled: true,
                    ),
                    false,
                    true,
                  );
            },
            child: AnimatedContainer(
              duration: const Duration(milliseconds: 400),
        decoration: BoxDecoration(
          borderRadius: const BorderRadius.horizontal(
            left: Radius.circular(5),
          ),
          color: user.isHelmetEnabled
              ? Theme.of(context).primaryColor
              : Colors.white,
        ),
        padding: const EdgeInsets.all(14),
        alignment: Alignment.center,
        child: Text(
          "Turn On",
          style: TextStyle(
            fontSize: 14,
            color: !user.isHelmetEnabled
                ? Theme.of(context).primaryColor
                : Colors.white,
            fontWeight: FontWeight.w600,
          ),
        ),
      ),
    ),
  ),
Expanded(
  child: GestureDetector(
    onTap: () {
      context.read<UsersProvider>().updateUser(
            context,
            user.copyWith(
              isHelmetEnabled: false,
            ),
            false,
            true,
          );
```

```
                  },
                  child: AnimatedContainer(
                    duration: const Duration(milliseconds: 400),
                    decoration: BoxDecoration(
                      borderRadius: const BorderRadius.horizontal(
                        right: Radius.circular(5),
                      ),
                      color: !user.isHelmetEnabled
                          ? Theme.of(context).primaryColor
                          : Colors.white,
                    ),
                    padding: const EdgeInsets.all(14),
                    alignment: Alignment.center,
                    child: Text(
                      "Turn Off",
                      style: TextStyle(
                        fontSize: 14,
                        color: user.isHelmetEnabled
                            ? Theme.of(context).primaryColor
                            : Colors.white,
                        fontWeight: FontWeight.w600,
                      ),
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
        const SizedBox(height: 20)
      ],
    );
  }
}
```

```
class _StatsCards extends StatelessWidget {
  const _StatsCards({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(16),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            'Helmet Stats',
            style: TextStyle(
              fontSize: 18,
              color: Colors.grey.shade700,
              fontWeight: FontWeight.w600,
            ),
          ),
          const SizedBox(height: 16),
          Row(
            children: const [
              Expanded(
                child: _StatsCard(
                  title: 'Oxygen',
                  value: '94.23%',
                  color: Colors.green,
                ),
              ),
              SizedBox(width: 16),
              Expanded(
                child: _StatsCard(
                  title: 'Hydrogen',
                  value: '69.12%',
                  color: Colors.grey,
                ),
              ),
            ],
          ),
          const SizedBox(height: 16),
          Row(
            children: const [
              Expanded(
                child: _StatsCard(
                  title: 'Dioxide Carbon',
                  value: '22.99%',
                  color: Colors.black,
                ),
              ),
              SizedBox(width: 16),
              Expanded(
                child: _StatsCard(
                  title: 'Gas',
                  value: '74.12%',
                  color: Colors.orange,
                ),
              ),
            ],
          ),
        ],
      ),
    );
  }
}

class _StatsCard extends StatelessWidget {
  const _StatsCard({
    Key? key,
    required this.title,
    required this.value,
    required this.color,
  }) : super(key: key);

  final String title;
  final String value;
  final Color color;

  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: BoxDecoration(
        border: Border.all(
          color: Colors.grey.shade200,
        ),
        color: Colors.white,
        boxShadow: [
          BoxShadow(
            color: Colors.grey.shade200,
            blurRadius: 10,
          ),
        ],
        borderRadius: BorderRadius.circular(5),
      ),
      child: ClipRRect(
        borderRadius: BorderRadius.circular(5),
        child: IntrinsicHeight(
          child: Row(
            children: [
              Container(
                width: 5,
                color: color,
              ),
              Padding(
                padding: const EdgeInsets.all(14),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      title,
                      style: const TextStyle(
                        fontSize: 15,
                        color: Colors.grey,
                      ),
                    ),
                    const SizedBox(height: 16),
                    Text(
                      value,
                      style: TextStyle(
                        fontSize: 22,
                        color: Colors.grey.shade800,
                        fontWeight: FontWeight.w600,
                      ),
                    ),
                  ],
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}

class _HeaderItem extends StatelessWidget {
  const _HeaderItem({
```

```
  Key? key,
  required this.title,
  required this.value,
  required this.color,
}) : super(key: key);
final String title;
final String value;
final Color color;

@override
Widget build(BuildContext context) {
  return Container(
    color: Colors.white,
    padding: const EdgeInsets.all(8.0),
    child: Column(
      children: [
        Text(
          title,
          style: const TextStyle(
            fontSize: 13,
            color: Colors.grey,
          ),
        ),
        const SizedBox(height: 8),
        Text(
          value,
          style: const TextStyle(
            fontSize: 15,
            color: Colors.black,
          ),
        ),
        const SizedBox(height: 8),
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 8.0),
            child: Container(
              height: 4,
              decoration: BoxDecoration(
                color: color,
                borderRadius: BorderRadius.circular(10),
              ),
              width: double.infinity,
            ),
          ),
        ],
      ),
    );
  }
}
```

Appendix B-8

```
import 'package:flutter/material.dart';

class TextFieldWidget extends StatelessWidget {
  const TextFieldWidget({
    super.key,
    required this.onChanged,
    required this.label,
    this.initialValue = '',
    this.suffix,
    this.isPassword = false,
    this.errorText = '',
    this.maxLines = 1,
    this.isMultiLine = false,
  });

  final ValueChanged<String> onChanged;
  final String label;
  final bool isPassword;
  final Widget? suffix;
  final String initialValue;
  final String errorText;
  final int maxLines;
  final bool isMultiLine;

  @override
  Widget build(BuildContext context) {
    const border = OutlineInputBorder(
      borderSide: BorderSide(
        color: Colors.transparent,
      ),
);
return Padding(
  padding: const EdgeInsets.only(bottom: 22),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(
        '$label :',
        style: TextStyle(
          fontSize: 16,
          color: Colors.grey.shade700,
          fontWeight: FontWeight.w600,
        ),
      ),
      const SizedBox(height: 10),
      SizedBox(
        height: isMultiLine
            ? null
            : maxLines == 1
                ? 55
                : null,
        child: TextFormField(
          textInputAction:
              isMultiLine ? TextInputAction.newline : TextInputAction.next,
          maxLines: maxLines,
          initialValue: initialValue,
          onChanged: onChanged,
          obscureText: isPassword,
          style: const TextStyle(
            color: Colors.black,
            fontWeight: FontWeight.w500,
          ),
          decoration: InputDecoration(
            border: border,
      enabledBorder: border,
      focusedBorder: border,
      filled: true,
      fillColor: Theme.of(context).primaryColor.withOpacity(0.06),
      hintStyle: TextStyle(
        fontSize: 14,
        color: Colors.grey.shade600,
      ),
      suffixIcon: suffix,
    ),
  ),
),
AnimatedSwitcher(
  duration: const Duration(milliseconds: 400),
  transitionBuilder: (child, anim) {
    return SizeTransition(
      sizeFactor: anim,
      axisAlignment: -1,
      child: FadeTransition(
        opacity: anim,
        child: child,
      ),
    );
  },
  child: errorText.isNotEmpty
      ? Padding(
          padding: const EdgeInsets.only(
            left: 2,
            right: 2,
            top: 2,
            bottom: 6,
          ),
          child: Text(
            errorText,
                      style: const TextStyle(
                        color: Colors.redAccent,
                        fontSize: 13,
                      ),
                    ),
                  )
                : const SizedBox.shrink(),
          ),
        ],
      ),
    );
  }
}
```

Appendix B-9

```
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:flutter/material.dart';
import 'package:helmet_iot/models/user.dart';
import 'package:helmet_iot/providers/authentication_provider.dart';
import 'package:helmet_iot/providers/users_provider.dart';
import 'package:helmet_iot/views/text_field.dart';
import 'package:provider/provider.dart';

class SignUpScreen extends StatefulWidget {
  const SignUpScreen(
      {Key? key, this.isFromAdmin = false, this.isEdit = false, this.user})
      : super(key: key);
  final bool isFromAdmin;
  final bool isEdit;
  final User? user;

  @override
  State<SignUpScreen> createState() => _SignUpScreenState();
}

class _SignUpScreenState extends State<SignUpScreen> {
  String username = '';
  bool showUsernameErrorMessage = false;
  String email = '';
  bool showEmailErrorMessage = false;
  String password = '';
  bool showPasswordErrorMessage = false;
  bool showPassword = false;
  String confirmPassword = '';
  bool showConfirmPasswordErrorMessage = false;
bool showConfirmPassword = false;
String chronicDiseases = '';

String role = 'worker';
bool isActive = true;

String validateUsername(String value) {
  final sanetizeValue = value.trim();
  if (sanetizeValue.isEmpty) return "username is required";
  return '';
}

String validatePasswordText(String value) {
  final sanetizeValue = value.trim();
  if (sanetizeValue.isEmpty) return "password is required";
  if (sanetizeValue.length < 6) {
    return "password must be 6 characters at least";
  }
  return '';
}

String validateConfirmPasswordText(String value) {
  final sanetizeValue = value.trim();
  if (sanetizeValue.isEmpty) return "password is required";
  if (sanetizeValue.length < 6) {
    return "password must be 6 characters at least";
  }
  if (password != value) {
    return "passwords doesn't match";
  }
  return '';
}

String validateEmailText(String value) {
    final sanetizeValue = value.trim();
    if (sanetizeValue.isEmpty) return "email is required";
    final bool emailValid = RegExp(
            r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-
Z]+")
        .hasMatch(email);
    if (!emailValid) return "please enter a valid email";

    return '';
  }

  bool get isUsernameValid => validateUsername(username).isEmpty;
  bool get isEmailValid => validateEmailText(email).isEmpty;
  bool get isPasswordValid =>
      validatePasswordText(password).isEmpty &&
      validateConfirmPasswordText(confirmPassword).isEmpty;

  String deviceToken = '';

  @override
  void initState() {
    super.initState();
    if (!widget.isEdit && !widget.isFromAdmin) {
      FirebaseMessaging.instance.getToken().then((value) {
        deviceToken = value ?? '';
      });
    }

    if (widget.isEdit) {
      username = widget.user!.username;
      email = widget.user!.email;
      role = widget.user!.role;
      deviceToken = widget.user!.deviceToken;
      chronicDiseases = widget.user!.chronicDiseases;
      isActive = widget.user!.isActive;
    }
  }

  @override
  Widget build(BuildContext context) {
    final authenticationProvider = context.watch<AuthenticationProvider>();
    final usersProvider = context.watch<UsersProvider>();
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.transparent,
        elevation: 0,
        iconTheme: IconThemeData(color: Theme.of(context).primaryColor),
      ),
      body: Align(
        alignment: const Alignment(0, -1 / 4),
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.symmetric(
              horizontal: 16,
              vertical: 20,
            ),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                Text(
                  widget.isEdit
                      ? "Update Profile"
                      : widget.isFromAdmin
                          ? 'Add User'
                          : 'Create Account',
                  style: const TextStyle(
                    fontSize: 34,
                    fontWeight: FontWeight.w600,
                  ),
                ),
                const SizedBox(height: 50),
                TextFieldWidget(
                  initialValue: username,
                  onChanged: (value) {
                    setState(() {
                      showUsernameErrorMessage = true;
                      username = value;
                    });
                  },
                  label: "Username",
                  errorText: showUsernameErrorMessage
                      ? validateUsername(username)
                      : "",
                ),
                const SizedBox(height: 6),
                TextFieldWidget(
                  initialValue: email,
                  onChanged: (value) {
                    setState(() {
                      showEmailErrorMessage = true;
                      email = value;
                    });
                  },
                  label: "Email",
                  errorText:
                      showEmailErrorMessage ? validateEmailText(email) : "",
                ),
                if (!widget.isEdit) ...[
                  const SizedBox(height: 6),
                  TextFieldWidget(
                    onChanged: (value) {
                      setState(() {
```

```
      showPasswordErrorMessage = true;
      password = value;
    });
  },
  label: "Password",
  errorText: showPasswordErrorMessage
      ? validatePasswordText(password)
      : "",
  isPassword: !showPassword,
  suffix: IconButton(
    onPressed: () {
      setState(() {
        showPassword = !showPassword;
      });
    },
    icon: Icon(
      showPassword ? Icons.visibility : Icons.visibility_off,
    ),
  ),
),
const SizedBox(height: 6),
TextFieldWidget(
  onChanged: (value) {
    setState(() {
      showConfirmPasswordErrorMessage = true;
      confirmPassword = value;
    });
  },
  label: "Confirm Password",
  errorText: showConfirmPasswordErrorMessage
      ? validateConfirmPasswordText(confirmPassword)
      : "",
  isPassword: !showConfirmPassword,
  suffix: IconButton(
      onPressed: () {
        setState(() {
          showPassword = !showPassword;
        });
      },
      icon: Icon(
        showConfirmPassword
            ? Icons.visibility
            : Icons.visibility_off,
      ),
    ),
  ),
],
if (role == 'worker') ...[
  TextFieldWidget(
    maxLines: 4,
    isMultiLine: true,
    initialValue: chronicDiseases,
    onChanged: (value) {
      setState(() {
        chronicDiseases = value;
      });
    },
    label: "Chronic Diseases",
  ),
  const SizedBox(height: 6),
],
if (widget.isFromAdmin) ...[
  const SizedBox(height: 10),
  Card(
    margin: EdgeInsets.zero,
    child: Padding(
      padding: const EdgeInsets.symmetric(horizontal: 16),
      child: DropdownButton(
      value: role,
      isExpanded: true,
      underline: Container(),
      items: const [
        DropdownMenuItem(
          value: 'worker',
          child: Text('Worker'),
        ),
        DropdownMenuItem(
          value: 'emergency',
          child: Text('Emergency'),
        ),
        DropdownMenuItem(
          value: 'admin',
          child: Text('Admin'),
        ),
      ],
      onChanged: (value) {
        setState(() {
          role = value!;
        });
      },
    ),
  ),
),
const SizedBox(height: 20),
Card(
  margin: EdgeInsets.zero.copyWith(bottom: 10),
  child: CheckboxListTile(
    value: isActive,
    title: const Text("Is Active"),
    onChanged: (value) {
      setState(() {
        isActive = value!;
        });
      },
    ),
  ),
],
const SizedBox(height: 10),
ElevatedButton(
  onPressed: () {
    if (isEmailValid &&
        (widget.isEdit || isPasswordValid) &&
        isUsernameValid) {
      if (widget.isEdit) {
        final userProvider =
            context.read<AuthenticationProvider>();
        context.read<UsersProvider>().updateUser(
              context,
              widget.user!.copyWith(
                deviceToken: deviceToken,
                email: email,
                username: username,
                role: role,
                isActive: isActive,
                chronicDiseases: chronicDiseases,
              ),
              true,
              widget.user!.id == userProvider.user.id,
            );
        context.read<AuthenticationProvider>();
      } else {
        context.read<AuthenticationProvider>().createAccount(
              username: username,
              email: email,
              password: password,
              role: role,
                    context: context,
                    deviceToken: deviceToken,
                    chronicDiseases: chronicDiseases,
                    isActive: isActive,
                    isFromUsersAdd: widget.isFromAdmin,
                  );
              }
            }
          },
          child: ((widget.isEdit)
                  ? usersProvider.isLoading
                  : authenticationProvider.isLoading)
              ? const Center(
                  child: CircularProgressIndicator(
                    color: Colors.white,
                  ),
                )
              : Text(
                  widget.isEdit
                      ? "Update"
                      : widget.isFromAdmin
                          ? "Add"
                          : "Create Account",
                  style: const TextStyle(
                    fontSize: 18,
                    color: Colors.white,
                    fontWeight: FontWeight.w600,
                  ),
                ),
        ),
      ],
    ),
  ),
),
      ),
    );
  }
}
```

Appendix B-10

```
import 'package:flutter/material.dart' hide Notification;
import 'package:helmet_iot/models/notification.dart';
import 'package:helmet_iot/providers/authentication_provider.dart';
import 'package:helmet_iot/providers/notification_provider.dart';
import 'package:provider/provider.dart';
import 'package:timeago/timeago.dart';

class NotificationsScreen extends StatefulWidget {
  const NotificationsScreen({Key? key}) : super(key: key);

  @override
  State<NotificationsScreen> createState() => _NotificationsScreenState();
}

class _NotificationsScreenState extends State<NotificationsScreen> {
  @override
  void initState() {
    super.initState();
    WidgetsBinding.instance.addPostFrameCallback((timeStamp) {
      context.read<NotificationProvider>().getNotifications(
            context,
            context.read<AuthenticationProvider>().user,
          );
    });
  }

  @override
  Widget build(BuildContext context) {
    final notificationProvider = context.watch<NotificationProvider>();
    return Scaffold(
      appBar: AppBar(
        title: const Text("Notifications"),
      ),
      body: notificationProvider.isLoading
          ? const Center(
              child: CircularProgressIndicator(),
            )
          : !notificationProvider.isLoading &&
                  notificationProvider.notifications.isEmpty
              ? const Center(
                  child: Text("There is no notifications"),
                )
              : ListView.builder(
                  itemCount: notificationProvider.notifications.length,
                  padding: const EdgeInsets.all(14),
                  itemBuilder: (context, index) {
                    final notification =
                        notificationProvider.notifications[index];
                    return _NotificationCard(notification: notification);
                  },
                ),
    );
  }
}

class _NotificationCard extends StatelessWidget {
  const _NotificationCard({
    super.key,
    required this.notification,
  });
  final Notification notification;

  @override
  Widget build(BuildContext context) {
```

```
return GestureDetector(
  onTap: () {},
  child: Card(
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(5),
    ),
    margin: const EdgeInsets.only(bottom: 20),
    elevation: 4,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Padding(
          padding: const EdgeInsets.all(12),
          child: Row(
            children: [
              Container(
                padding: const EdgeInsets.all(8),
                decoration: BoxDecoration(
                  color: Theme.of(context).primaryColor.withOpacity(0.15),
                  borderRadius: BorderRadius.circular(5),
                ),
                child: Icon(
                  Icons.notifications,
                  color: Theme.of(context).primaryColor,
                  size: 25,
                ),
              ),
              const SizedBox(width: 10),
              Expanded(
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
        notification.title,
        style: const TextStyle(
          fontSize: 16,
          fontWeight: FontWeight.w500,
        ),
      ),
      const SizedBox(height: 6),
      Row(
        children: [
          Text(
            notification.notificationType.name,
            style: TextStyle(
              color: Colors.grey.shade700,
            ),
          ),
          const Text('   '),
          Text(
            '#${notification.id!.substring(0, 15)}',
            style: TextStyle(
              color: Colors.grey.shade700,
              fontSize: 13,
            ),
          ),
        ],
      ),
    ],
  ),
),
Text(
  format(
    notification.createdAt!,
    locale: 'en_short',
  ),
  style: const TextStyle(
                      color: Colors.grey,
                    ),
                  ),
                ],
              ),
            ),
            Divider(
              thickness: 1.2,
              height: 16,
              color: Colors.grey.shade300,
            ),
            Padding(
              padding: const EdgeInsets.all(14).copyWith(top: 4),
              child: Text(
                notification.body,
                style: TextStyle(
                  color: Colors.black.withOpacity(0.7),
                  fontWeight: FontWeight.w400,
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Appendix B-11

```
import 'package:flutter/material.dart';
import 'package:helmet_iot/providers/authentication_provider.dart';
import 'package:provider/provider.dart';

class NotActiveScreen extends StatelessWidget {
  const NotActiveScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Padding(
        padding: const EdgeInsets.all(20),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            CircleAvatar(
              radius: 60,
              backgroundColor: Theme.of(context).primaryColor,
              child: const Icon(
                Icons.person_off,
                color: Colors.white,
                size: 50,
              ),
            ),
            const SizedBox(height: 20),
            const Text(
              "Not Activated",
              style: TextStyle(
                fontSize: 24,
                fontWeight: FontWeight.w600,
              ),
            ),
            const SizedBox(height: 10),
            Text(
              "your account is not activated you can communicate our team to enable
your account",
              style: TextStyle(
                fontSize: 16,
                color: Colors.grey.shade700,
              ),
              textAlign: TextAlign.center,
            ),
            const SizedBox(height: 40),
            ElevatedButton(
              onPressed: () {
                context.read<AuthenticationProvider>().logout(context: context);
              },
              child: const Text(
                "Logout",
                style: TextStyle(
                  fontSize: 18,
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Appendix B-12

```
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:flutter/material.dart';
import 'package:helmet_iot/providers/authentication_provider.dart';
import 'package:helmet_iot/views/signup_screen.dart';
import 'package:helmet_iot/views/text_field.dart';
import 'package:provider/provider.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({Key? key}) : super(key: key);

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  String email = '';
  bool showEmailErrorMessage = false;
  String password = '';
  bool showPasswordErrorMessage = false;
  bool showPassword = false;

  String validatePasswordText(String value) {
    final sanetizeValue = value.trim();
    if (sanetizeValue.isEmpty) return "password is required";
    if (sanetizeValue.length < 6) {
      return "password must be 6 characters at least";
    }
    return '';
  }
  String validateEmailText(String value) {
    final sanetizeValue = value.trim();
    if (sanetizeValue.isEmpty) return "email is required";
    final bool emailValid = RegExp(
            r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-
Z]+")
        .hasMatch(email);
    if (!emailValid) return "please enter a valid email";

    return '';
  }

  bool get isEmailValid => validateEmailText(email).isEmpty;
  bool get isPasswordValid => validatePasswordText(password).isEmpty;

  String deviceToken = '';

  @override
  void initState() {
    super.initState();
    FirebaseMessaging.instance.getToken().then((value) {
      deviceToken = value ?? '';
    });
  }

  @override
  Widget build(BuildContext context) {
    final authenticationProvider = context.watch<AuthenticationProvider>();
    return Scaffold(
      body: Align(
        alignment: const Alignment(0, -1 / 4),
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.symmetric(
  horizontal: 16,
  vertical: 20,
),
child: Column(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    const Text(
      'Login',
      style: TextStyle(
        fontSize: 36,
        fontWeight: FontWeight.w600,
      ),
    ),
    const SizedBox(height: 50),
    TextFieldWidget(
      onChanged: (value) {
        setState(() {
          showEmailErrorMessage = true;
          email = value;
        });
      },
      label: "Email",
      errorText:
          showEmailErrorMessage ? validateEmailText(email) : "",
    ),
    const SizedBox(height: 6),
    TextFieldWidget(
      onChanged: (value) {
        setState(() {
          showPasswordErrorMessage = true;
          password = value;
        });
      },
      label: "Password",
  errorText: showPasswordErrorMessage
      ? validatePasswordText(password)
      : "",
  isPassword: !showPassword,
  suffix: IconButton(
    onPressed: () {
      setState(() {
        showPassword = !showPassword;
      });
    },
    icon: Icon(
      showPassword ? Icons.visibility : Icons.visibility_off,
    ),
  ),
),
const SizedBox(height: 50),
ElevatedButton(
  onPressed: () {
    if (isEmailValid && isPasswordValid) {
      context.read<AuthenticationProvider>().login(
            email: email,
            password: password,
            deviceToken: deviceToken,
            context: context,
          );
    }
    FocusScope.of(context).unfocus();
  },
  child: authenticationProvider.isLoading
      ? const Center(
          child: CircularProgressIndicator(
            color: Colors.white,
          ),
        )
```

```
          : const Text(
              "Login",
              style: TextStyle(
                fontSize: 18,
                color: Colors.white,
                fontWeight: FontWeight.w600,
              ),
            ),
    ),
    const SizedBox(height: 30),
    ElevatedButton(
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.white,
      ),
      onPressed: () {
        Navigator.of(context).push(
          MaterialPageRoute(
            builder: (context) {
              return const SignUpScreen();
            },
          ),
        );
      },
      child: Text(
        "Create A new Account",
        style: TextStyle(
          fontSize: 17,
          color: Theme.of(context).primaryColor,
          fontWeight: FontWeight.w600,
        ),
      ),
    ),
  ],
),
          ),
        ),
      ),
    );
  }
}
```

Appendix B-13

```
// ignore_for_file: avoid_renaming_method_parameters

import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';
import 'package:flutter_map/flutter_map.dart';
import 'package:latlong2/latlong.dart';

final erbilLocation = LatLng(36.198572, 43.971241);

class MapWidget extends StatefulWidget {
  const MapWidget({
    Key? key,
    required this.onLocationSelected,
    this.isSmall = false,
    this.initialLocation,
  }) : super(key: key);

  final ValueChanged<LatLng> onLocationSelected;
  final bool isSmall;
  final LatLng? initialLocation;

  @override
  State<MapWidget> createState() => _MapWidgetState();
}

class _MapWidgetState extends State<MapWidget> {
  LatLng? currentLocation;

  @override
  void initState() {
```

```
  super.initState();
  if (widget.initialLocation != null) {
    setState(() {
      if (widget.initialLocation!.latitude == 0 &&
          widget.initialLocation!.longitude == 0) {
        currentLocation = erbilLocation;
      } else {
        currentLocation = LatLng(
          widget.initialLocation!.latitude,
          widget.initialLocation!.longitude,
        );
      }
    });
  }
}

@override
Widget build(BuildContext context) {
  final size = MediaQuery.of(context).size;
  return SafeArea(
    child: Material(
      type: MaterialType.transparency,
      child: Stack(
        children: [
          SizedBox(
            height: widget.isSmall ? size.height * 0.25 : size.height,
            child: ClipRRect(
              borderRadius: BorderRadius.circular(5),
              child: IgnorePointer(
                ignoring: widget.isSmall,
                child: FlutterMap(
                  options: MapOptions(
                    maxZoom: 18,
                    zoom: widget.isSmall ? 11 : 15,
          center: currentLocation ?? erbilLocation,
          onTap: (tapPosition, latLng) {
            setState(() {
              currentLocation = latLng;
            });
          },
        ),
        layers: [
          TileLayerOptions(
            urlTemplate:
                'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
            subdomains: ['a', 'b', 'c'],
            tileProvider: const CachedTileProvider(),
          ),
          MarkerLayerOptions(
            rotate: true,
            markers: [
              Marker(
                height: 60,
                width: 60,
                point: currentLocation ?? erbilLocation,
                builder: (ctx) => const Icon(
                  Icons.location_on,
                  size: 30,
                  color: Colors.red,
                ),
              ),
            ],
          ),
        ],
      ),
    ),
  ),
),
if (!widget.isSmall)
  Positioned(
    top: 10,
    right: 10,
    child: FloatingActionButton(
      heroTag: 'back_button',
      backgroundColor: Theme.of(context).primaryColor,
      onPressed: () {
        if (currentLocation != null) {
          widget.onLocationSelected(
            LatLng(
              currentLocation!.latitude,
              currentLocation!.longitude,
            ),
          );
        }
        Navigator.of(context).pop();
      },
      child: const Icon(
        Icons.done,
        color: Colors.white,
      ),
    ),
  ),
if (!widget.isSmall)
  Positioned(
    top: 10,
    left: 10,
    child: FloatingActionButton(
      heroTag: 'done_button',
      backgroundColor: Theme.of(context).primaryColor,
      onPressed: () {
        Navigator.of(context).pop();
      },
                  child: const Icon(
                    Icons.arrow_back,
                    color: Colors.white,
                  ),
                ),
              ),
          ],
        ),
      ),
    );
  }
}

class CachedTileProvider extends TileProvider {
  const CachedTileProvider();
  @override
  ImageProvider getImage(Coords<num> cords, TileLayerOptions options) {
    return CachedNetworkImageProvider(
      getTileUrl(cords, options),
    );
  }
}
```

Appendix B-14

```
// ignore_for_file: avoid_renaming_method_parameters

import 'dart:io';

import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';
import 'package:flutter_map/flutter_map.dart';
import 'package:latlong2/latlong.dart';
import 'package:location/location.dart';
import 'package:permission_handler/permission_handler.dart'
    hide PermissionStatus;

final erbilLocation = LatLng(36.198572, 43.971241);

class LiveLocation extends StatefulWidget {
  const LiveLocation({
    Key? key,
    this.initialLatLng,
  }) : super(key: key);

  final LatLng? initialLatLng;

  @override
  State<LiveLocation> createState() => _LiveLocationState();
}

class _LiveLocationState extends State<LiveLocation> {
  LatLng currentLocation = LatLng(0, 0);
  int currentIndex = 0;
```

```
Future<void> _getUserLocation() async {
  final location = Location();
  var permissionGranted = await location.hasPermission();
  if (permissionGranted == PermissionStatus.denied) {
    permissionGranted = await location.requestPermission();
    if (permissionGranted != PermissionStatus.granted) {
      return;
    }
  }
  var serviceEnabled = await location.serviceEnabled();
  if (!(serviceEnabled)) {
    serviceEnabled = await location.requestService();
    if (!(serviceEnabled)) {
      return;
    }
  }

  final locationData = await location.getLocation();
  if (mounted) {
    setState(() {
      currentLocation = LatLng(
        locationData.latitude!,
        locationData.longitude!,
      );
    });
  }
}

void _requestPermission() async {
  final location = Platform.isIOS
      ? await Permission.locationWhenInUse.request()
      : await Permission.location.request();

  if (location.isGranted) {
    _getUserLocation();
  }
}

@override
void initState() {
  super.initState();
  if (widget.initialLatLng != null) {
    setState(() {
      currentLocation = widget.initialLatLng!;
    });
  } else {
    _requestPermission();
  }
}

final MapController mapController = MapController();

@override
Widget build(BuildContext context) {
  final size = MediaQuery.of(context).size;
  return Scaffold(
    appBar: AppBar(
      title: const Text("Live Location"),
    ),
    body: SizedBox(
      height: size.height,
      child: currentLocation.latitude == 0 && currentLocation.longitude == 0
          ? const Center(
              child: CircularProgressIndicator(),
            )
          : ClipRRect(
              borderRadius: BorderRadius.circular(5),
              child: FlutterMap(
mapController: mapController,
options: MapOptions(
  maxZoom: 18,
  zoom: 15,
  center: currentLocation,
),
layers: [
  TileLayerOptions(
    urlTemplate:
        'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
    subdomains: ['a', 'b', 'c'],
    tileProvider: const CachedTileProvider(),
  ),
  MarkerLayerOptions(
    rotate: true,
    markers: [
      ...List.generate(1, (index) {
        return Marker(
          height: 80,
          width: 80,
          point: currentLocation,
          builder: (ctx) => Container(
            height: double.infinity,
            width: double.infinity,
            decoration: BoxDecoration(
              color: Theme.of(context)
                  .primaryColor
                  .withOpacity(0.15),
              shape: BoxShape.circle,
              border: Border.all(
                color: Theme.of(context).primaryColor,
              ),
            ),
            child: const Icon(
                                Icons.location_on,
                                color: Colors.red,
                                size: 30,
                              ),
                            ),
                          );
                        }),
                      ],
                    ),
                  ],
                ),
              ),
        ),
      );
    }
  }

  class CachedTileProvider extends TileProvider {
    const CachedTileProvider();
    @override
    ImageProvider getImage(Coords<num> cords, TileLayerOptions options) {
      return CachedNetworkImageProvider(
        getTileUrl(cords, options),
      );
    }
  }
```

Appendix B-15

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

class EmergencyScreen extends StatefulWidget {
  const EmergencyScreen({Key? key}) : super(key: key);

  @override
  State<EmergencyScreen> createState() => _EmergencyScreenState();
}

class _EmergencyScreenState extends State<EmergencyScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
    );
  }
}
```

Appendix B-16

```
// ignore_for_file: use_build_context_synchronously

import 'package:flutter/material.dart';
import 'package:flutter_slidable/flutter_slidable.dart';
import 'package:helmet_iot/models/emergency_case.dart';
import 'package:helmet_iot/models/user.dart';
import 'package:helmet_iot/providers/authentication_provider.dart';
import 'package:helmet_iot/providers/emergency_case_provider.dart';
import 'package:helmet_iot/services/location.dart';
import 'package:helmet_iot/utils/utils.dart';
import 'package:helmet_iot/views/live_location_history.dart';
import 'package:helmet_iot/views/notifications.dart';
import 'package:intl/intl.dart';
import 'package:latlong2/latlong.dart';
import 'package:provider/provider.dart';
import 'package:timeago/timeago.dart' as timeago;

class EmergencyCasesScreen extends StatefulWidget {
  const EmergencyCasesScreen({Key? key, this.isAdmin = false})
      : super(key: key);
  final bool isAdmin;

  @override
  State<EmergencyCasesScreen> createState() =>
_EmergencyCasesScreenState();
}

class _EmergencyCasesScreenState extends State<EmergencyCasesScreen> {
  @override
  void initState() {
```

```
  super.initState();
  WidgetsBinding.instance.addPostFrameCallback((timeStamp) {
    final userId = context.read<AuthenticationProvider>().user.id;
    if (widget.isAdmin) {
      context.read<EmergencyCaseProvider>().getCases(context, userId, false);
    } else {
      context.read<EmergencyCaseProvider>().getCases(context, userId, true);
    }
  });
}

@override
Widget build(BuildContext context) {
  final casesProvider = context.watch<EmergencyCaseProvider>();
  final currentUser = context.read<AuthenticationProvider>().user;
  return Scaffold(
    floatingActionButton: currentUser.isWorker
        ? FloatingActionButton.extended(
            backgroundColor: Theme.of(context).primaryColor,
            onPressed: () async {
              if (!currentUser.isHelmetEnabled) {
                showSnackBar(
                  context: context,
                  message:
                      "Your protection is off you can't submit emergency request",
                );
                return;
              }

              final location = await LocationService().getUserLocation();
              if (location == null) return;
              final emergencyCase = EmergencyCase(
                longitude: location.longitude!,
                latitude: location.latitude!,
          userId: currentUser.id,
          emergencyId: '',
          createdAt: DateTime.now(),
          userAccount: currentUser,
        );

        await context.read<EmergencyCaseProvider>().addCase(
             context,
             emergencyCase,
           );
      },
      label: const Text('Request Emergency'),
    )
  : null,
appBar: AppBar(
  title: const Text("Emergency Cases"),
  actions: [
    if (widget.isAdmin) ...[
      IconButton(
        onPressed: () {
          Navigator.of(context).push(
            MaterialPageRoute(
              builder: (context) {
                return const NotificationsScreen();
              },
            ),
          );
        },
        icon: const Icon(Icons.notifications),
      ),
      TextButton(
        style: TextButton.styleFrom(
          foregroundColor: Colors.white,
        ),
        onPressed: () {
          context.read<AuthenticationProvider>().logout(context: context);
        },
        child: const Text('Logout'),
      ),
    ]
  ],
),
body: Stack(
  children: [
    !casesProvider.isLoading && casesProvider.cases.isEmpty
        ? const Center(
            child: Text("There is no cases submitted yet"),
          )
        : Column(
            children: [
              if (widget.isAdmin) const _SearchField(),
              Expanded(
                child: ListView.builder(
                  padding: const EdgeInsets.all(16),
                  itemCount: casesProvider.searchText.isNotEmpty
                      ? casesProvider.searchedCases.length
                      : casesProvider.cases.length,
                  itemBuilder: (context, index) {
                    final emergencyCase =
                        casesProvider.searchText.isNotEmpty
                            ? casesProvider.searchedCases[index]
                            : casesProvider.cases[index];

                    final user = emergencyCase.userAccount;
                    final emergency = emergencyCase.emergencyAccount;
                    return _EmergencyCaseCard(
                      emergency: emergency,
                      user: user,
                            emergencyCase: emergencyCase,
                            index: index,
                            isAdmin: widget.isAdmin,
                          );
                        },
                      ),
                    ),
                  ],
                ),
          if (casesProvider.isLoading)
            const Center(
              child: CircularProgressIndicator(),
            ),
        ],
      ),
    );
  }
}

class _EmergencyCaseCard extends StatefulWidget {
  final User emergency;
  final User user;
  final EmergencyCase emergencyCase;
  final int index;
  final bool isAdmin;
  const _EmergencyCaseCard({
    Key? key,
    required this.emergency,
    required this.user,
    required this.emergencyCase,
    required this.index,
    required this.isAdmin,
  }) : super(key: key);
```

```
 @override
 State<_EmergencyCaseCard> createState() => _EmergencyCaseCardState();
}

class _EmergencyCaseCardState extends State<_EmergencyCaseCard> {
 void updateCase(EmergencyCase emergencyCase, EmergencyCaseStatus
status) {
  final currentUser = context.read<AuthenticationProvider>().user;

  context.read<EmergencyCaseProvider>().updateCase(
     context,
     emergencyCase.copyWith(
      caseStatus: status,
      emergencyId: currentUser.id,
     ),
    );
 }

 @override
 Widget build(BuildContext context) {
  final currentUser = context.read<AuthenticationProvider>().user;

  return Card(
   margin: const EdgeInsets.only(bottom: 10),
   child: Slidable(
    key: ValueKey(widget.index),
    endActionPane: !widget.isAdmin
      ? null
      : ActionPane(
        extentRatio: 0.7,
        motion: const ScrollMotion(),
        children: [
         SlidableAction(
          onPressed: (_) {
```

```
    updateCase(
      widget.emergencyCase,
      EmergencyCaseStatus.rejected,
    );
  },
  backgroundColor: Colors.red,
  foregroundColor: Colors.white,
  icon: Icons.close,
  label: 'Reject',
),
SlidableAction(
  onPressed: (_) {
    updateCase(
      widget.emergencyCase,
      EmergencyCaseStatus.approved,
    );
  },
  backgroundColor: Colors.green,
  foregroundColor: Colors.white,
  icon: Icons.local_hospital,
  label: 'Approve',
),
SlidableAction(
  onPressed: (_) {
    updateCase(
      widget.emergencyCase,
      EmergencyCaseStatus.done,
    );
  },
  backgroundColor: Colors.blue,
  foregroundColor: Colors.white,
  icon: Icons.done,
  label: 'Done',
),
    ],
   ),
child: Column(
 crossAxisAlignment: CrossAxisAlignment.start,
 children: [
  ExpansionTile(
   leading: CircleAvatar(
    backgroundColor: Theme.of(context).primaryColor,
    child: Text(
     "${widget.index + 1}",
     style: const TextStyle(
      color: Colors.white,
     ),
    ),
   ),
   title: Text(
    currentUser.isWorker
      ? "Case ${widget.index + 1}"
      : widget.user.username,
    style: const TextStyle(
     color: Colors.black,
    ),
   ),
   trailing: Text(
    timeago.format(
     widget.emergencyCase.createdAt!,
     locale: 'en_short',
    ),
    style: const TextStyle(
     color: Colors.grey,
     fontSize: 14,
    ),
   ),
   subtitle: Text(
  widget.emergencyCase.caseStatus.name,
  style: TextStyle(
    color: Colors.grey.shade700,
    fontSize: 13,
  ),
),
children: [
  const Divider(
    thickness: 1.1,
    height: 1,
  ),
  Padding(
    padding: const EdgeInsets.all(12),
    child: SingleChildScrollView(
      scrollDirection: Axis.horizontal,
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          if (widget.isAdmin) ...[
            _ItemTile(
              iconData: Icons.person,
              label: widget.user.username,
            ),
            const SizedBox(width: 10),
          ],
          _ItemTile(
            onTap: () {
              Navigator.of(context).push(
                MaterialPageRoute(
                  builder: (context) {
                    return LiveLocation(
                      initialLatLng: LatLng(
                        widget.emergencyCase.latitude,
                        widget.emergencyCase.longitude,
                  ),
                );
              },
            ),
          );
        },
        iconData: Icons.location_on,
        label: 'Location',
      ),
      const SizedBox(width: 10),
      if (widget.emergency.username.isNotEmpty &&
          !widget.isAdmin) ...[
        _ItemTile(
          iconData: Icons.emergency,
          label: widget.emergency.username,
        ),
        const SizedBox(width: 10),
      ],
      _ItemTile(
        iconData: Icons.category,
        label: widget.emergencyCase.caseStatus.name,
      ),
      const SizedBox(width: 10),
      if (!widget.isAdmin)
        _ItemTile(
          iconData: Icons.timer,
          label: DateFormat.yMMMd()
              .format(widget.emergencyCase.createdAt!),
        ),
    ],
  ),
 ),
),
if (widget.isAdmin && widget.user.chronicDiseases.isNotEmpty)
```

```
        Padding(
          padding: const EdgeInsets.all(16),
          child: Align(
            alignment: Alignment.centerLeft,
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text.rich(
                  TextSpan(
                    text: 'Chronic Diseases: ',
                    style: TextStyle(
                      fontSize: 14,
                      color: Colors.grey.shade700,
                      fontWeight: FontWeight.w700,
                    ),
                  ),
                ),
                const SizedBox(height: 4),
                Text(
                  widget.user.chronicDiseases,
                  style: TextStyle(
                    fontSize: 14,
                    color: Colors.grey.shade700,
                    fontWeight: FontWeight.w400,
                    height: 1.6,
                  ),
                )
              ],
            ),
          ),
        ),
      ],
  ),
],
        ),
      ),
    );
  }
}

class _ItemTile extends StatelessWidget {
  const _ItemTile({
    Key? key,
    required this.iconData,
    required this.label,
    this.onTap,
  }) : super(key: key);
  final IconData iconData;
  final String label;
  final VoidCallback? onTap;

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: onTap,
      child: SizedBox(
        child: Container(
          decoration: BoxDecoration(
            border: Border.all(
              color: Colors.grey.shade200,
            ),
            borderRadius: BorderRadius.circular(10),
          ),
          padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 10),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              CircleAvatar(
                radius: 20,
                backgroundColor:
                    Theme.of(context).backgroundColor.withOpacity(0.15),
                child: Icon(
                  iconData,
                  size: 20,
                  color: Theme.of(context).primaryColor,
                ),
              ),
              const SizedBox(height: 6),
              Text(
                label,
                style: TextStyle(
                  fontSize: 14,
                  color: Colors.grey.shade700,
                  fontWeight: FontWeight.w500,
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}

class _SearchField extends StatefulWidget {
  const _SearchField({Key? key}) : super(key: key);

  @override
  State<_SearchField> createState() => _SearchFieldState();
}

class _SearchFieldState extends State<_SearchField> {
```

```
final TextEditingController textEditingController = TextEditingController();

@override
void dispose() {
  textEditingController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  final searchText = context.watch<EmergencyCaseProvider>().searchText;
  final border = OutlineInputBorder(
    borderSide: BorderSide(
      color: Colors.grey.shade300,
    ),
  );
  return Container(
    margin: const EdgeInsets.all(16).copyWith(bottom: 0),
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(5),
      boxShadow: const [
        BoxShadow(color: Colors.black12, spreadRadius: 0, blurRadius: 3),
      ],
    ),
    child: TextField(
      controller: textEditingController,
      onChanged: (value) {
        context.read<EmergencyCaseProvider>().search(value);
      },
      decoration: InputDecoration(
        border: border,
        enabledBorder: border,
        focusedBorder: border,
        filled: true,
          fillColor: Colors.white,
          hintText: "Search for cases",
          prefixIcon: Icon(
            Icons.search,
            color: Theme.of(context).primaryColor,
          ),
          suffixIcon: searchText.isNotEmpty
              ? IconButton(
                  onPressed: () {
                    context.read<EmergencyCaseProvider>().clearSearch();
                    textEditingController.clear();
                  },
                  icon: Icon(
                    Icons.close,
                    color: Theme.of(context).primaryColor,
                  ),
                )
              : null,
        ),
      ),
    );
  }
}
```

Appendix B-17

```
import 'package:flutter/material.dart';

class TextFieldWidget extends StatelessWidget {
  const TextFieldWidget({
    super.key,
    required this.onChanged,
    required this.label,
    this.initialValue = '',
    this.suffix,
    this.isPassword = false,
    this.errorText = '',
    this.maxLines = 1,
  });

  final ValueChanged<String> onChanged;
  final String label;
  final bool isPassword;
  final Widget? suffix;
  final String initialValue;
  final String errorText;
  final int maxLines;

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.only(bottom: 20),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          SizedBox(
  height: maxLines == 1 ? 55 : null,
  child: TextFormField(
    maxLines: maxLines,
    initialValue: initialValue,
    onChanged: onChanged,
    obscureText: isPassword,
    style: const TextStyle(
      color: Colors.black,
      fontWeight: FontWeight.w500,
    ),
    decoration: InputDecoration(
      hintText: label,
      hintStyle: TextStyle(
        fontSize: 16,
        color: Colors.grey.shade800,
      ),
      suffixIcon: suffix,
    ),
  ),
),
AnimatedSwitcher(
  duration: const Duration(milliseconds: 400),
  transitionBuilder: (child, anim) {
    return SizeTransition(
      sizeFactor: anim,
      axisAlignment: -1,
      child: FadeTransition(
        opacity: anim,
        child: child,
      ),
    );
  },
  child: errorText.isNotEmpty
      ? Padding(
          padding: const EdgeInsets.only(
            left: 2,
            right: 2,
            top: 2,
            bottom: 6,
          ),
          child: Text(
            errorText,
            style: const TextStyle(
              color: Colors.redAccent,
              fontSize: 13,
            ),
          ),
        )
      : const SizedBox.shrink(),
  ),
 ],
),
);
}
}
```

Appendix B-18

```
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars,
avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
///
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for web - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
    case TargetPlatform.macOS:
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for macos - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    case TargetPlatform.windows:
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for windows - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    case TargetPlatform.linux:
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for linux - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    default:
      throw UnsupportedError(
        'DefaultFirebaseOptions are not supported for this platform.',
      );
  }
}

static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyDhFiV8YlAGAifAeygH7myf0H2_ArBb74k',
  appId: '1:1036999106467:android:ef11bb42d2e49f32dd8416',
  messagingSenderId: '1036999106467',
  projectId: 'helmet-iot-692f4',
  storageBucket: 'helmet-iot-692f4.appspot.com',
);

static const FirebaseOptions ios = FirebaseOptions(
  apiKey: 'AIzaSyCu_XbGT83FvmULtNSUtcNlIv_Op1R7yog',
  appId: '1:1036999106467:ios:42eeee6617fbaf45dd8416',
  messagingSenderId: '1036999106467',
    projectId: 'helmet-iot-692f4',
    storageBucket: 'helmet-iot-692f4.appspot.com',
    iosBundleId: 'com.example.helmetIot',
  );
}
```

Appendix B-19

```
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:helmet_iot/firebase_options.dart';
import 'package:helmet_iot/providers/authentication_provider.dart';
import 'package:helmet_iot/providers/emergency_case_provider.dart';
import 'package:helmet_iot/providers/notification_provider.dart';
import 'package:helmet_iot/providers/users_provider.dart';
import 'package:helmet_iot/repositories/authentication_repository.dart';
import 'package:helmet_iot/repositories/emergency_case_repository.dart';
import 'package:helmet_iot/repositories/notification_repository.dart';
import 'package:helmet_iot/repositories/user_repository.dart';
import 'package:helmet_iot/services/local_notification.dart';
import 'package:helmet_iot/services/local_storage.dart';
import 'package:helmet_iot/services/send_notification.dart';
import 'package:helmet_iot/views/cases_screen.dart';
import 'package:helmet_iot/views/login_screen.dart';
import 'package:helmet_iot/views/not_active.dart';
import 'package:helmet_iot/views/user_home_page.dart';
import 'package:helmet_iot/views/users_page.dart';
import 'package:provider/provider.dart';

void main() async {
WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);

await LocalStorageService().init();
await LocalNotificationService().init();
```

```
await
SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);

SystemChrome.setSystemUIOverlayStyle(
const SystemUiOverlayStyle(
statusBarColor: Colors.transparent,
statusBarBrightness: Brightness.light,
statusBarIconBrightness: Brightness.dark,
),
);

return runApp(const App());
}

class App extends StatefulWidget {
const App({Key? key}) : super(key: key);

@override
State<App> createState() => _AppState();
}

class _AppState extends State<App> {
final FirebaseMessaging messaging = FirebaseMessaging.instance;

Future<void> requestPermissionAndRegister() async {
final settings = await messaging.requestPermission();
if (settings.authorizationStatus == AuthorizationStatus.authorized) {
FirebaseMessaging.onMessage.listen(
(RemoteMessage message) async {
if (message.notification != null) {
await LocalNotificationService().showNotification(
<String, dynamic>{
'title': message.notification!.title,
'body': message.notification!.body,
},
);
}
},
);
}
}

@override
void initState() {
super.initState();
requestPermissionAndRegister();
}

@override
Widget build(BuildContext context) {
return MultiProvider(
providers: [
ChangeNotifierProvider(create: (context) {
return AuthenticationProvider(
localStorageService: LocalStorageService(),
authenticationRepository: AuthenticationRepository(),
userRepository: UserRepository(),
);
}),
],
child: Builder(
builder: (context) {
final user = context.watch<AuthenticationProvider>().user;

return MultiProvider(
providers: [
ChangeNotifierProvider(create: (context) {
```

```
return UsersProvider(
userRepository: UserRepository(),
);
}),
ChangeNotifierProvider(create: (context) {
return NotificationProvider(
notificationsRepository: NotificationRepository(),
sendNotificationService: const SendNotificationService(),
);
}),
ChangeNotifierProvider(create: (context) {
return EmergencyCaseProvider(
emergencyCaseRepository: EmergencyCaseRepository(),
);
}),
],
child: MaterialApp(
debugShowCheckedModeBanner: false,
title: 'Helmet App',
theme: ThemeData.light().copyWith(
primaryColor: const Color(0xff2B72A8),
appBarTheme: const AppBarTheme(
systemOverlayStyle: SystemUiOverlayStyle(
statusBarIconBrightness: Brightness.light,
),
iconTheme: IconThemeData(color: Colors.white),
backgroundColor: Color(0xff2B72A8),
),
elevatedButtonTheme: ElevatedButtonThemeData(
style: ButtonStyle(
minimumSize: MaterialStateProperty.all(
const Size(double.infinity, 50),
),
backgroundColor:
```

```
MaterialStateProperty.all(const Color(0xff2B72A8)),
),
),
),
home: Builder(
builder: (context) {
bool isUserAuthenticates = !user.isEmpty;

return isUserAuthenticates
? !user.isActive
? const NotActiveScreen()
: user.isAdmin
? const UsersScreen()
: user.isWorker
? const UserHomePage()
: EmergencyCasesScreen(
isAdmin: user.isEmergency,
)
: const LoginScreen();
},
),
),
);
},
),
);
}
}
```

Appendix B-20

```
// This is a basic Flutter widget test.
//
// To perform an interaction with a widget in your test, use the WidgetTester
// utility in the flutter_test package. For example, you can send tap and scroll
// gestures. You can also use WidgetTester to find child widgets in the widget
// tree, read text, and verify that the values of widget properties are correct.

import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

import 'package:helmet_iot/main.dart';

void main() {
  testWidgets('Counter increments smoke test', (WidgetTester tester) async {
    // Build our app and trigger a frame.
    await tester.pumpWidget(const MyApp());

    // Verify that our counter starts at 0.
    expect(find.text('0'), findsOneWidget);
    expect(find.text('1'), findsNothing);

    // Tap the '+' icon and trigger a frame.
    await tester.tap(find.byIcon(Icons.add));
    await tester.pump();

    // Verify that our counter has incremented.
    expect(find.text('0'), findsNothing);
    expect(find.text('1'), findsOneWidget);
  });
}
```